Università degli Studi di Salerno
Corso di Ingegneria del Software

EduCat
System Design Document
Versione 1.3

Data: 20/01/2026

Coordinatore del progetto:

Nome	Matricola
Isabella Dima	0512119719

Partecipanti:

Nome	Matricola
Isabella Dima	0512119719
Anna Chiara Esposito	0512119143

Scritto da:	Isabella Dima

Revision History

Data	Versione	Descrizione	Autore
25/11/2025	1.0	System Design Document	Isabella Dima
13/01/2026	1.1	Modifica alla Decomposizione dei sottosistemi, servizi dei sottosistemi e Boundary Conditions	Anna Chiara Esposito
13/01/2026	1.2	Aggiornamento Glossario, acronimi	Anna Chiara Esposito
20/01/2026	1.3	Rivisitazione ai BDUC e la divisione dei servizi in sottosistemi	Anna Chiara Esposito

Indice

1. Introduzione

1.1 Scopo del Sistema

EduCat è una piattaforma web di prenotazione e acquisto di lezioni private rivolta a tutor di diverse discipline, studenti di qualsiasi grado di istruzione e genitori.
L'obiettivo del sistema è offrire a studenti e genitori (in caso di studenti minorenni) un modo semplice per confrontare i tutor in base a modalità di insegnamento, fascia di prezzo e disponibilità, così da prenotare lezioni secondo le proprie esigenze e possibilità economiche. I tutor potranno registrarsi sulla piattaforma e offrire ripetizioni a pagamento.
Moduli accessori come il sistema di Recensioni sono stati modellati a livello di database e interfaccia, ma la loro logica di business è esclusa dal perimetro di questa release.

1.2 Obiettivi di Design

Performance	Il sistema deve garantire tempi di risposta sotto i 2 secondi per il 90% delle richieste utente al fine di evitare frustrazione da parte dell'utente e l'abbandono del sito.
Affidabilità	Il sistema deve garantire l'integrità dei dati durante le transazioni, in maniera che esse vengano annullate se non sono completate.
Sicurezza	La piattaforma adotta misure di sicurezza quali query parametrizzate e hashing di password.
Portabilità	Il sistema deve essere accessibile su diverse piattaforme utilizzando interfacce e protocolli standardizzati e utilizzando la programmazione modulare.
Usabilità	Il sistema deve essere mobile friendly e offrire un'interfaccia responsive. L'utente medio deve essere in grado di effettuare la registrazione, cercare una lezione e completare l'acquisto in meno di 5 minuti.
Modularità e Manutenibilità	Il sistema adotta pattern MVC per separare la logica di business, la presentazione e l'accesso ai dati persistenti per migliore manutenibilità e maggiore facilità nel debugging.

1.3 Definizioni, acronimi e abbreviazioni

Nel seguente documento sono presenti diversi acronimi, qui riportiamo il significato di ciascuno di essi:

- CSS: Cascading Style Sheets
- DBMS: DataBase Management System
- FK: Foreign Key
- HTML: HyperText Markup Language
- HTTP: HyperText Transfer Protocol

- JDBC: Java Database Connectivity
- JSP: JavaServer Pages
- MVC: Model-View-Controller
- PK: Primary Key
- RAD: Requirements Analysis Document
- SDD: System Design Document
- UCBC: Use Case Boundary Condition
- UID: User Identifier
- UML: Unified Modeling Language

1.4 Riferimenti

- ProblemStatement_EduCat v1.1
- RAD_EduCat v1.5
- Bernd Bruegge & Allen Dutoit - Object-Oriented Software Engineering: Using UML, Patterns, and Java

1.5 Overview

Il documento è diviso in quattro sezioni:

- **Software Attuale**: illustrazione di architetture di sistemi simili a quello proposto.
- **Architettura Software Proposta**: modello di system design per il nuovo sistema.
- **Servizi dei Sottosistemi**: descrizione di start-up, shutdown e comportamento in caso di errori del sistema.
- **Glossario**: lista di termini utilizzati nel documento con relative definizioni.

2. Software attuale

Attualmente i sistemi con funzionalità simili a EduCat sono applicazioni web realizzate con pattern MVC e con modello Client/Server.
Caratteristiche comuni sono l'utilizzo di database relazionali, interfacce web responsive e mobile friendly e un sistema di recensioni.
Tuttavia alcuni sistemi mostrano limiti quali scarsa ottimizzazione della ricerca delle lezioni e mancanza di trasparenza nei processi di segnalazione utenti.

3. Architettura Software Proposta

3.1 Overview

La piattaforma "EduCat" è un'applicazione web basata su architettura Client/Server e design pattern MVC. Così facendo la modularità del sistema permette una maggiore leggibilità e manutenibilità.
Il Web Server e Container scelto per il sistema è Apache Tomcat.

Sviluppo del back-end:

- Java23
- Eclipse

Sviluppo del front-end:

- HTML5
- CSS3

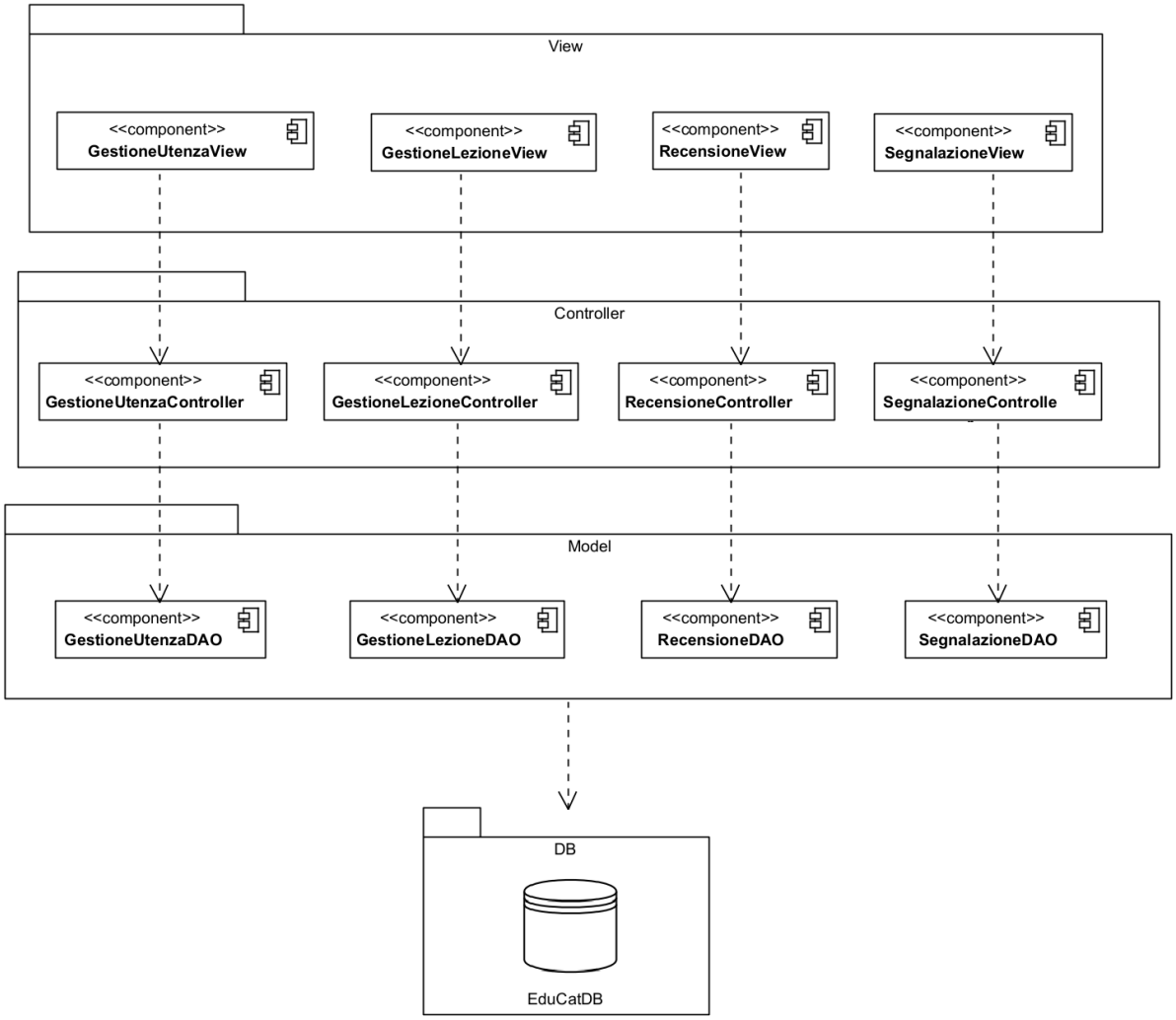
- JavaScript
- JSP

Persistenza dei dati:
- JDBC
- MySQL

3.2 Decomposizione in sottosistemi

Il sistema viene quindi suddiviso nei seguenti sottosistemi:
- **Gestione Utenza**: si occupa di gestire le funzioni di login, logout e di registrazione per gli utenti di tipo "Studente", "Genitore" e "Tutor", visualizzazione, modifica e cancellazione account. Per gli admin sono rese disponibili le medesime funzioni, con l'aggiunta della funzione di cancellazione degli utenti.
- **Gestione Lezioni**: è responsabile della ricerca, visualizzazione delle informazioni, prenotazione e acquisto delle lezioni. Permette anche di visualizzare le lezioni acquistate (sia da parte dello studente/genitore che tutor) ed eventualmente annullare la prenotazione.
- **Gestione Recensioni**: si occupa di inserire e visualizzare le recensioni e calcolare una media delle valutazioni relative al tutor *(Componente presente in stato di Mock-up funzionale a fini dimostrativi)*.
- **Gestione Segnalazioni**: è responsabile della lettura e gestione delle segnalazioni.

Di seguito una vista dettagliata di ciascun sottosistema evidenziando le componenti principali:
- **View**: è responsabile di mostrare gli oggetti del dominio applicativo agli utenti sotto forma di pagine web.
- **Controller**: si occupa della logica per il controllo e business del sistema, è responsabile della sequenza di interazioni con l'utente e di notificare il View dei cambiamenti nel model.
- **Model**: è responsabile della conoscenza del dominio applicativo. Si occupa della gestione dei dati, quindi della memorizzazione dei dati sul database e dell'interazione con il DB stesso.
- **Database:** Rappresenta il livello di memorizzazione fisica dei dati persistenti. È il componente esterno con cui il Model interagisce per salvare e recuperare le informazioni.



3.3 Mapping Hardware/Software

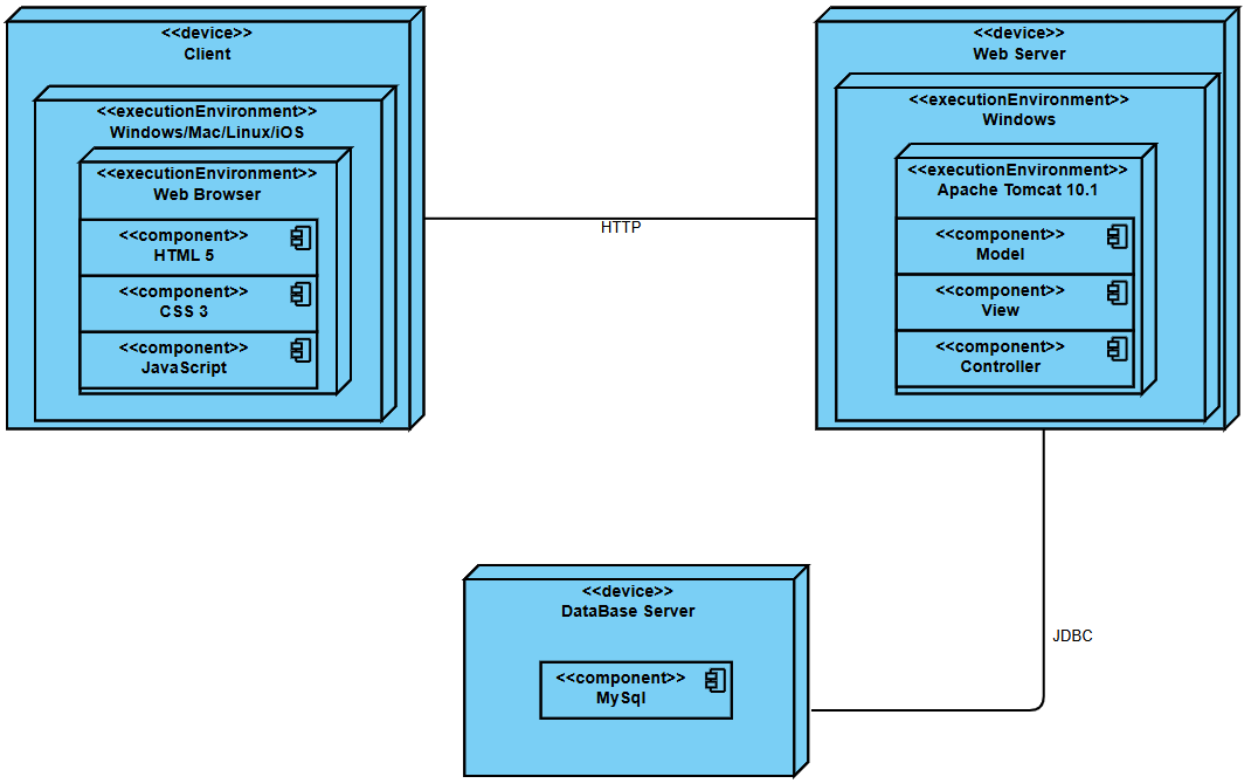
L'architettura del sistema EduCat è basata su un modello Client-Server a tre livelli (Three-Tier), configurata secondo un approccio Thin Client. In questa configurazione, la logica di business risiede principalmente sul server, alleggerendo il carico computazionale sui dispositivi degli utenti e centralizzando la gestione delle risorse.

Il sistema è strutturato nei seguenti nodi logici e fisici:

- **Client (Presentation Tier):** Ogni Client connesso a Internet (PC, Smartphone o Tablet) invia richieste al Web Server attraverso il proprio Web Browser. Tutta la presentazione dei dati è gestita tramite standard web (HTML5, CSS3, JavaScript), mentre la comunicazione con il server avviene tramite protocollo HTTP.

- **Web Server (Application Tier):** Il Web Server, eseguito su un ambiente Windows, utilizza il container Apache Tomcat 10.1 per l'esecuzione dell'applicazione. La logica interna segue il pattern architetturale MVC, mappato sulle tecnologie Java SE.

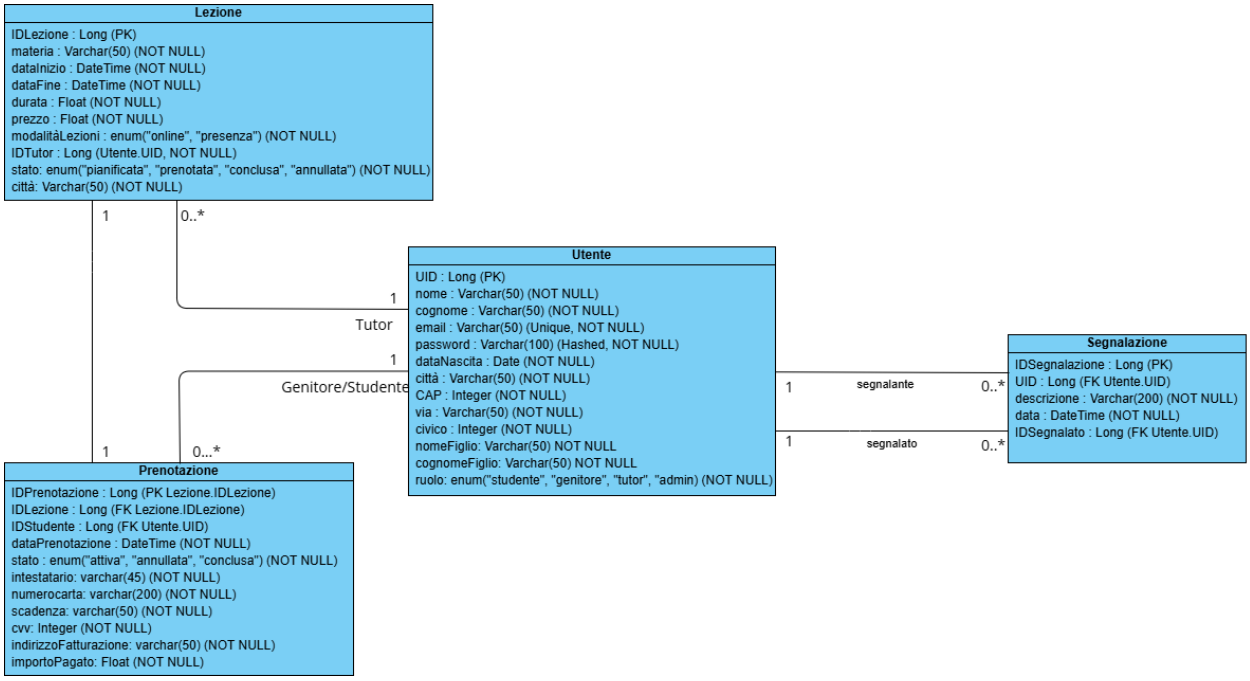
- **Database Server (Data Tier):** La gestione dei dati persistenti è affidata al DBMS MySQL, scelto per ragioni di sicurezza, affidabilità e performance. Il Web Server comunica con il Database per interrogare, aggiungere o rimuovere dati utilizzando driver JDBC (Java Database Connectivity), garantendo una separazione efficace tra logica applicativa e memorizzazione dei dati.



3.4 Gestione dei Dati Persistenti

Per la gestione dei dati persistenti è stato selezionato MySQL, un database relazionale. I motivi di questa scelta sono la facilità di utilizzo, l'integrità e accuratezza dei dati, sicurezza e performance.

Il grafico sottostante rappresenta ciascuna entità all'interno del database con relativi attributi, Primary Key (PK), Foreign Key (FK), vincoli di integrità (NOT NULL) e relazioni tra le varie entità con le relative cardinalità.



3.5 Access Control and Security

Operazione\Attore	***Studente/Genitore***	***Tutor***	***Amministratore Utenti***
Gestione Utenza	• Login • Logout • Iscrizione • Visualizzazione Profilo • Modifica informazioni personali • Modifica metodo di pagamento • Modifica password • Cancellazione account	• Login • Logout • Iscrizione • Visualizzazione Profilo • Modifica informazioni personali • Modifica metodo di pagamento • Modifica password • Cancellazione account	• Login • Logout • Visualizzazione Profilo • Modifica password • Visualizza utenti registrati • Ricerca per nome • Cancella utenti
Gestione Lezioni	• Visualizza informazioni lezione • Ricerca lezioni • Acquisto lezioni • Annullamento lezione • Visualizza	• Modifica lezioni offerte • Visualizza lezioni • Riscossione pagamento • Cancella lezioni previste	-

	lezioni acquistate		
Recensioni (Mock)	• Scrivi recensione	• Visualizza tutte le recensioni	-
Segnalazioni	• Segnalazione utenti	• Segnalazione utenti	• Lettura segnalazioni • Risposta segnalazioni

3.6 Global Software Control

Il flusso di controllo del sistema EduCat segue un modello Event-Driven (guidato dagli eventi). Le operazioni vengono avviate attraverso eventi innescati dall'interazione dell'utente con l'interfaccia grafica (View), implementata tramite pagine JSP. Tali eventi generano poi richieste HTTP che vengono gestite dal Web Browser e smistate alle classi Servlet dedicate (Controller). Queste ultime, dopo l'interazione con il Model e l'elaborazione dei dati, inviato una risposta al client sotto forma di pagina HTML (generata dinamicamente dalla JSP) visualizzabile dal browser e al Client.

3.7 Boundary Conditions

In questa sezione vengono descritti i comportamenti del sistema nei casi limite: avvio (Start-up), spegnimento (Shut Down) e gestione dei fallimenti (Failures).

3.7.1. Start-up

All'avvio, il sistema Educat inizializza i servizi di connessione al database e rende disponibile una pagina di login, la quale permetterà all'utente di poter accedere alla Home Page e a tutte le funzionalità offerte dal sito.

UCBC1: Avvio del Sistema

- *Descrizione*: L'amministratore avvia il server e sistema.
- *Attore principale*: Amministratore.
- *Entry condition*: L'amministratore accede al server.
- *Exit condition* - *On success*: Il sistema è attivo e le funzionalità sono disponibili agli utenti.
- *Exit condition* - *On failure*: Il sistema non si avvia (es: errore di configurazione).
- *Flusso di Eventi*:
  1. Amministratore: Esegue il comando per inizializzare il Web Server (startup di Tomcat).
  2. Sistema: Carica il contesto dell'applicazione, stabilisce la connessione con il Database (via JDBC) e rende le Servlet pronte a ricevere richieste.

3.7.2. Shut down

Il sistema EduCat deve essere spento in modo controllato per garantire che i dati persistenti siano salvati e le sessioni attive gestite correttamente.

UCBC2: Spegnimento del sistema

- *Descrizione*: L'amministratore esegue la procedura di arresto del sistema.
- *Attore principale*: Amministratore.

- *Entry condition*: Il sistema è attivo.
- *Exit condition* - *On success*: Il sistema è spento correttamente e le risorse rilasciate.
- *Exit condition* -*On failure*: Il sistema forza la chiusura o rimane in uno stato inconsistente.
- *Flusso di eventi principale*:
    1. Amministratore: Esegue il comando di arresto del sistema.
    2. Sistema: Verifica la presenza di client attivi e salva eventuali dati in sospeso. Se non ci sono operazioni critiche in corso, il servizio viene interrotto.
- *Flusso alternativo a (Client connessi)*:
    1. Sistema: Rileva sessioni attive. Notifica l'amministratore o attende il time-out delle sessioni prima di procedere allo spegnimento forzato.
- *Flusso alternativo b (Errore salvataggio)*:
    1. Sistema: Il salvataggio dei dati persistenti fallisce. L'errore viene loggato e notificato all'amministratore prima della chiusura definitiva.

3.7.3. Failures

Rischi di perdita dati dovuti a fallimenti software o hardware.

UCBC3: Fallimento del Sistema (Crash)

- *Descrizione*: Gestione del riavvio dopo un arresto anomalo.
- *Attore principale*: Amministratore.
- *Entry condition*: Il sistema ha smesso di funzionare inaspettatamente.
- *Exit condition* - *On success*: Il sistema è ripristinato ed è nuovamente operativo con il backup più recente.
- *Exit condition* - *On failure*: Il sistema non viene riavviato correttamente.
- *Flusso di eventi principale*:
    1. Amministratore: Esegue il comando per il ripristino del sistema.
    2. Sistema: Esegue la procedura di Start up (Include UCBC1). Se necessario, ripristina lo stato precedente tramite log delle transazioni.
- *Flusso alternativo a*:
    1. Sistema: Il sistema non viene avviato correttamente e le funzionalità non sono disponibili agli utenti.
    2. Amministratore: Analizza i log di errore del server per identificare la causa del guasto

Il sistema EduCat potrebbe avere problemi di accesso ai dati persistenti, oppure che quest'ultimi siano essere corrotti.

UCBC4: Errore di accesso ai dati persistenti

- *Descrizione*: Comportamento in caso di indisponibilità del Database (es. MySQL down).
- *Attore principale*: Amministratore
- *Entry condition*: Il sistema non riesce ad accedere ai dati oppure i dati risultano corrotti.
- *Exit condition* - *On success*: La connessione al DB è ristabilita.
- *Exit condition* -*On failure*: Il Sistema non riprende il normale funzionamento.
- *Flusso di eventi principale*:
    1. Sistema: Rileva l'eccezione di connessione (es. SQLException) Il Web Container restituisce all'utente un messaggio di errore standard HTTP 500 (Internal Server

Error), interrompendo l'elaborazione della richiesta.
2. Amministratore: Verifica lo stato del Database Server. Se necessario, riavvia il servizio MySQL o ripristina un backup dei dati.
3. Amministratore: Esegue il riavvio del sistema (UCBC1).

4. Servizi dei sottosistemi

Questa sezione definisce le operazioni pubbliche offerte dai sottosistemi identificati nell'architettura logica**.** *Nota di Mappatura*: Ogni componente denominato "*Controller* rappresenta un raggruppamento logico (package) che aggrega l'insieme delle Servlet responsabili per quello specifico dominio funzionale. Non esiste una corrispondenza 1-a-1 con una singola classe Java, ma una mappatura 1-a-n verso le Servlet del package corrispondente (es. it.unisa.educat.controller.gestioneutenza).

4.1 Gestione Utenza

Componente: GestioneUtenzaController

Operazione	***Descrizione***	***Mappatura***
login(email, password)	Verifica le credenziali nel database. Restituisce un token di sessione o un'eccezione in caso di fallimento.	LoginServlet
registraUtente(datiUtente)	Riceve un oggetto con i dati dell'utente (Studente, Genitore o Tutor) e crea la persistenza del nuovo utente.	RegistrazioneServlet
logout(session)	Invalida la sessione corrente dell'utente loggato.	LogoutServlet
eliminaAccount(idUtente)	Rimuove permanentemente il record dell'utente e i dati correlati dal database.	ElininaAccountServlet
modificaProfilo(idUtente, nuoviDati)	Aggiorna i campi modificabili del profilo (es. indirizzo) per l'utente specificato.	/

4.2 Gestione Lezione

Componente: GestioneLezioneController

Operazione	***Descrizione***	***Mappatura***
cercaLezioni(criteriRicerca)	Restituisce una lista di lezioni/tutor filtrata per materia, prezzo, città e disponibilità.	CercaLezioneServlet

pubblicaAnnuncio(datiLezione)	Permette al Tutor di salvare nel sistema una nuova proposta di lezione.	PubblicaAnnuncioServlet
prenotaLezione(idLezione, idStudente)	Crea una nuova prenotazione associando lo studente alla lezione.	PrenotaLezioneServlet
annullaPrenotazione(idPrenotazione)	Se i vincoli temporali lo consentono, cancella una prenotazione esistente e avvia il rimborso.	AnnullaPrenotazioneServlet
getStoricoLezioni(idUtente)	Recupera la lista delle lezioni (svolte, future, cancellate) associate all'utente.	StoricoLezioniServlet

4.3 Recensione

Componente: RecensioneController

Nota: Questo sottosistema è definito nell'architettura logica ma, nella versione attuale, non è implementato a livello di codice backend. Le interfacce utente mostrano dati simulati.

Operazione	***Descrizione***	***Mappatura***
aggiungiRecensione(idLezione, voto, testo)	Salva una nuova recensione collegandola alla lezione conclusa e all'autore.	/
getRecensioniTutor(idTutor)	Restituisce la lista di tutte le recensioni ricevute da un determinato Tutor.	/
calcolaMediaVoto(idTutor)	Calcola e restituisce il punteggio medio delle recensioni di un Tutor.	/

4.4 Segnalazione

Componente: SegnalazioneController

Operazione	***Descrizione***	***Mappatura***
inviaSegnalazione(idSegnalante, idSegnalato, motivo)	Crea un nuovo record di segnalazione nel sistema.	InviaSegnalazioneServlet
getListaSegnalazioni()	(Admin) Restituisce tutte le segnalazioni aperte che necessitano di moderazione.	ListaSegnalazioniServlet

risolviSegnalazione(idSegnalazione, esito)	(Admin) Chiude la segnalazione registrando l'azione intrapresa (es. ban utente o archiviazione).	RisolviSegnalazioneServlet

5. Glossario

Termine	*Significato*
Apache Tomcat	Web server e servlet container open source. Implementa le specifiche Java Servlet e JavaServer Pages.
Boundary Condition	Condizione limite o caso eccezionale nel funzionamento del sistema, come l'avvio (start-up), lo spegnimento (shut-down) o la gestione degli errori (failures).
Controller	Nel pattern MVC, è il componente che riceve l'input dell'utente (tramite la View), lo elabora (spesso interagendo con il Model) e determina la risposta o la vista successiva. In EduCat è implementato tramite Java Servlet.
Event-Driven	Paradigma di programmazione in cui il flusso del programma è determinato da eventi come le azioni dell'utente (click del mouse, invio di moduli) o messaggi da altri programmi.
Java Servlet	Componente software lato server che estende le capacità di un server web per gestire il modello richiesta-risposta.
Model	Nel pattern MVC, rappresenta la struttura dei dati e la logica di business dell'applicazione, indipendente dall'interfaccia utente. Gestisce l'interazione diretta con il database.
Thin Client	Architettura client-server in cui il client (in questo caso il browser) ha funzionalità ridotte e dipende principalmente dal server centrale per l'elaborazione dei dati e la logica applicativa.
Three-Tier Architecture	Architettura software multi-tier in cui i processi logici sono divisi in tre livelli fisici separati: Presentazione (Client), Logica Applicativa (Web Server) e Dati (Database Server).
View	Nel pattern MVC, è il componente responsabile della visualizzazione dei dati all'utente. In EduCat è implementato tramite pagine JSP.