Università degli Studi di Salerno
Corso di Ingegneria del Software

UniClass
System Design Document
Versione 1.0

Data: 23/11/2024

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:	Giuseppe Sabetta (GS), Sara Gallo (SG), Saverio D'Avanzo (SD), Gerardo Antonio Cetrulo (AC)

Revision History

Data	Versione	Descrizione	Autore
22/11/2024	1.0	System Design	GS, SG, SD, AC

Sommario

1. Introduzione

1.1. Scopo del Sistema

Lo scopo di UniClass è di migliorare l'esperienza degli studenti universitari, semplifica*ndo* le interazioni dei docenti con il sistema e con gli studenti stessi, permette*re* 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 tenendo conto del documento OWASP Top Ten, come standard di riferimento. Inoltre, per garantire la resilienza, la piattaforma avrà un file di backup per un corretto funzionamento del sistema in caso di guasti, malfunzionamenti o perdite di informazioni.

- **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 nei giorni d'esame o orari con più affluenza in Ateneo.

- **Supportability**: Proprio per i picchi di utenza presenti poco prima di ogni esame per localizzare aule, aule libere etc., il sistema deve presentare scalabilità.

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, con l'aggiunta di un'agenda personale per studenti.

- **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 a pieno verranno ereditate 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.

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:

    - **OrariManagement**: gestisce la visualizzazione, creazione e modifica degli orari settimanali, in base alle informazioni esplicitate nel Requirements Analysis Document

    - **AgendaManagement**: gestisce la creazione, visualizzazione e modifica dell'agenda di ogni utente di tipo "Studente".

    - **AuleManagement**: gestisce la visualizzazione delle aule libere e si basa sugli orari delle lezioni esistenti e sugli esami.

    - **UserManagement**: gestisce la creazione, visualizzazione, modifica degli utenti e delle loro informazioni.

    - **LezioniManagement**: gestisce la creazione, visualizzazione, modifica delle lezioni, delle aule di riferimento e la locazione temporale all'interno degli orari settimanali.

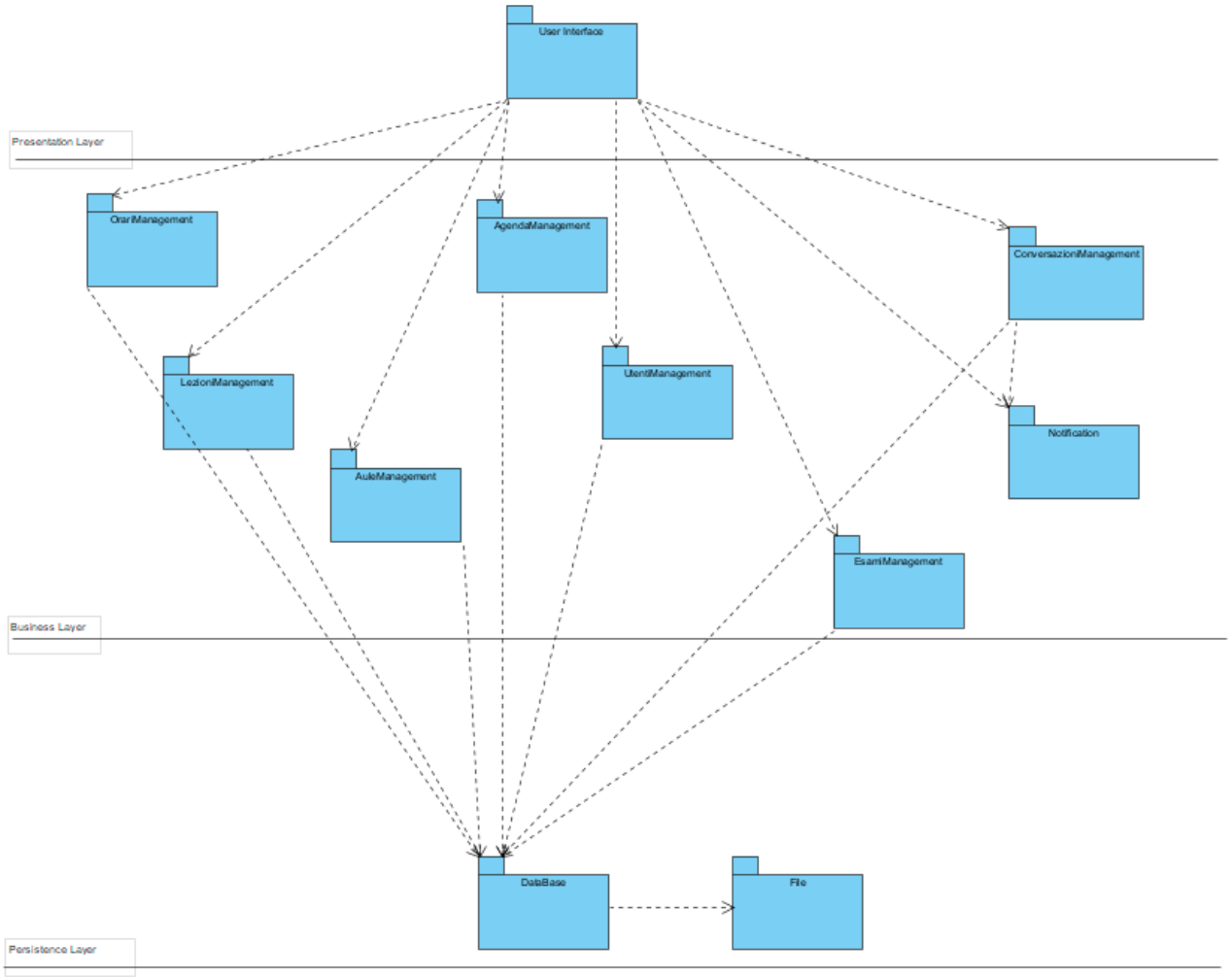
    - **EsamiManagement:** gestisce la creazione, visualizzazione, modifica degli esami, dei partecipanti, tra cui studenti e docenti.

    - **ConversazioniManagement:** 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

    - **Notification:** gestione delle notifiche presenti e generate per le

conversazioni degli utenti.

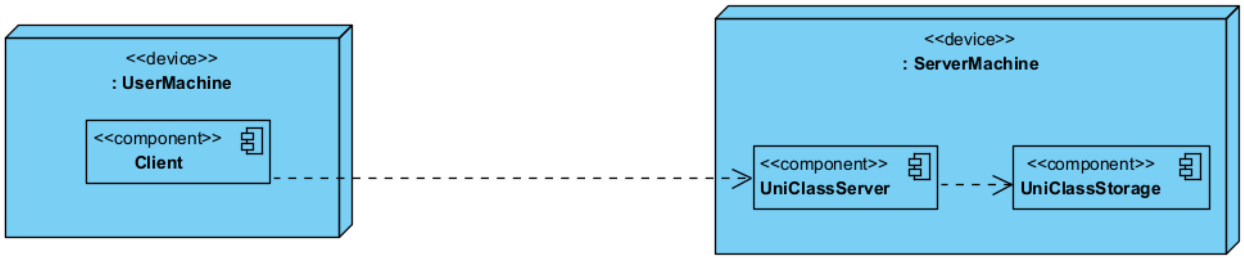
- **Persistence Layer:** per il Persistence Layer, si è deciso di utilizzare un Database in grado di memorizzare e rendere i dati richiesti persistenti. Per rendere il sistema resiliente, si è deciso di utilizzare un file di backup, in grado di contenere le informazioni che potrebbero essere perse a causa di malfunzionamenti.



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.



3.4 Gestione dei Dati Persistenti

Per la gestione delle operazioni all'interno del sistema UniClass, c'è la necessità di rendere diversi tipi di dati persistenti. Questi dati saranno acceduti diverse volte per diversi contesti e da diversi tipi di utenti:

- Utenti
- Lezioni
- Orari
- Aule Libere, Edifici, Strutture di UniClass
- Conversazioni
- Agende
- Esami

La strategia d'archiviazione di questa piattaforma si basa sul reperire, inserire o modificare dati correlati tra loro, quindi sulle transazioni in un Database relazionale, per cui useremo il linguaggio SQL (per usare direttamente il database MySQL) e la Java Persistence API (JPA) per effettuare le stesse transazioni nell'ambiente Java utilizzato. Inoltre, per le query in java, si userà l'API JPQL (Java Persistence Query Language). Inoltre, la piattaforma avrà un file di backup per resilienza del sistema in caso di malfunzionamenti.

3.5 Controllo degli Accessi e Sicurezza

Actor/Object	Lezioni	Orari	Aule	Agenda	Esami	Utenti	Conversazioni
Ospite	X	viewOrari	viewAule	X	X	X	X
Studente	X	viewOrari	viewAule	getAgenda setAgenda	signUp getEsami	X	createChat sendMessage
Docente	X	viewOrari	viewAule	X	viewStudenti viewEsami	X	sendResponse sendBroadcast
Coordinatore	createLezione setLezione removeLezione	viewOrari	viewAule	X	createExam modifyExam removeExam	X	sendBroadcast sendResponse
Personale TA	X	X	X	X	X	createUtente modificaUtente rimuoviUtente	X

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:

- **OrariManagement**: questo sottosistema permette la visualizzazione delle varie lezioni per

ogni dipartimento, anno, resto all'interno di una tabella raffigurante l'orario settimanale. Come servizi avrà *visualizzaOrario*

- **AuleManagement**: questo sottosistema permette la visualizzazione delle aule libere presenti per ogni edificio all'interno dell'Ateneo. Per questo motivo, l'unico servizio accessibile sarà *visualizzaAuleLibere*, per informare l'utente dell'orario di disponibilità e occupazione dell'aula desiderata nell'edificio di rilevanza

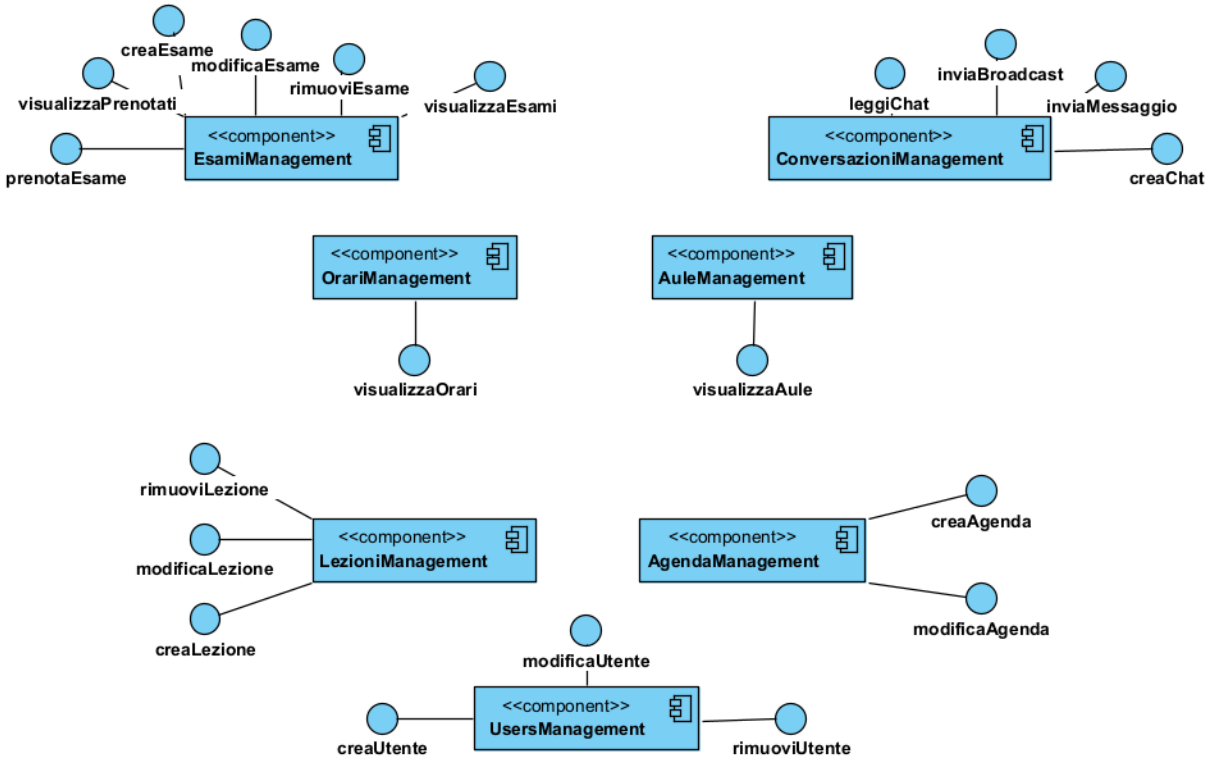
- **LezioniManagement**: questo sottosistema permette la creazione, modifica e rimozione delle lezioni che si presenteranno poi negli orari settimanali. I servizi sono diversi: *creaLezione, modificaLezione, eliminaLezione*.

- **AgendaManagement**: questo sottosistema permette all'utente di tipo "Studente" di creare e modificare la propria agenda personale, comprendendo un insieme di lezioni appartenenti al proprio dipartimento e non. I servizi saranno: *creaAgenda, modificaAgenda (aggiunta o modifica delle lezioni)*.

- **UsersManagement**: questo sottosistema permette la gestione degli utenti. I servizi presenti sono: *creaUtente, modificaUtente, rimuoviUtente* per permettere la creazione di un utente, con un suo ruolo (Studente, Docente etc.), modifica del ruolo o altre informazioni e rimozione.

- **EsamiManagement**: questo sottosistema permette all'utente di gestire gli esami in modo diverso, in base al suo ruolo/tipo. I servizi sono: *prenotaEsame*, *visualizzaPrenotati*, *creaEsame*, *modificaEsame*, *rimuoviEsame*, *visualizzaEsame*.

- **ConversazioniManagement**: questo sottosistema rappresenta la gestione delle varie conversazioni tra gli utenti del sistema, comprendendo la possibilità di inviare o ricevere messaggi e annunci. I servizi sono: *creaChat, inviaMessaggio, inviaBroadcast, leggiChat*


creaEsame
modificaEsame
rimuoviEsame
visualizzaPrenotati
visualizzaEsami
<<component>>
EsamiManagement
prenotaEsame
inviaBroadcast
leggiChat
inviaMessaggio
<<component>>
ConversazioniManagement
creaChat
<<component>>
OrariManagement
visualizzaOrari
<<component>>
AuleManagement
visualizzaAule
rimuoviLezione
modificaLezione
creaLezione
<<component>>
LezioniManagement
<<component>>
AgendaManagement
creaAgenda
modificaAgenda
modificaUtente
<<component>>
UsersManagement
creaUtente
rimuoviUtente