Università degli Studi di Salerno

Corso di Ingegneria del Software

UniMeet
Object Design
Versione 1.0

GitHub del progetto: /UniMeet

Data: 15/12/2024

Progetto: UniMeet	Versione: 1.0
Documento: System Design	Data: 20/11/2024

Partecipanti:

Nome	**Matricola**
Lorenza Rosa Pia Natale	0512116647
Giovanni Tufano	0512112027
Antonio Del Vecchio	0512118501
Ciro Danzilli	0512111007

Scritto da:	Tutti i membri del gruppo

Revision History

Data	Versione	Descrizione	Autore
14/10/2024	1.0	Problem Statement	Tutto il gruppo
28/10/2024	1.0	Requisiti e casi d'uso	Tutto il gruppo
11/11/2024	1.0	Requirements Analysis Document (RAD)	Tutto il gruppo
19/11/2024	1.1	Revisione RAD	Tutto il gruppo
20/11/2024	1.0	System Design	Tutto il gruppo
13/12/2024	1.0	Object Design	Tutto il gruppo
13/02/2025	1.0	Revisione ODD	Tutto il gruppo

Sommario

1. Descrizione documento *4*

2. Package *5*

3. Specifica delle interfacce *6*

3.2 Control **12**

2.3 Database **16**

2.4 Utils **16**

1. Descrizione documento

Lo scopo del seguente documento è quello di rappresentare in termini di oggetti il sistema UniMeet che andremo a realizzare. Nello specifico saranno rappresentate tutte le funzionalità descritte nel Requirement Analysis Document e System Design Document. In questo documento saranno definite le interfacce delle classi, le operazioni, i tipi gli argomenti e le signatures.
Inoltre sono specificati i trade-off.

1.1 obiettivi di design trade-off

- **tempo di rilascio vs qualità:** al fine di fornire tutti i servizi del sistema correttamente si è preferito sacrificare il tempo di rilascio per garantire il corretto funzionamento
- **portabilità vs efficienza:** è stato utilizzato java come linguaggio di programmazione in quanto esso è portabile, quindi indipendente dalla macchina, ciò ha influito sull'efficienza a causa delle sue prestazioni inferiori se paragonato ad altri linguaggi.
- **Interfaccia vs usabilità:** l'interfaccia grafica è stata realizzata in modo da essere molto semplice da utilizzare, essa fa uso di form, menu e pulsanti in maniera da rendere più semplice l'utilizzo del sistema.

1.2 linee guida per la documentazione delle interfacce

saranno seguite le seguenti convenzioni per la stesura del codice:

Naming convention

Saranno utilizzate delle nomenclature che utilizzeranno la struttura camelCase

Metodi

I nomi dei metodi saranno scritti in camelCase e indicano il verbo che rappresenta l'azione

Classi

I nomi delle classi cominciano con una lettera maiuscola come da convenzione in Java.

Riferimenti

Per la stesura del documento faremo riferimento al Requirement Analysis Document con l'acronimo di "RAD" e al System Design Document con l'acronimo di "SDD".

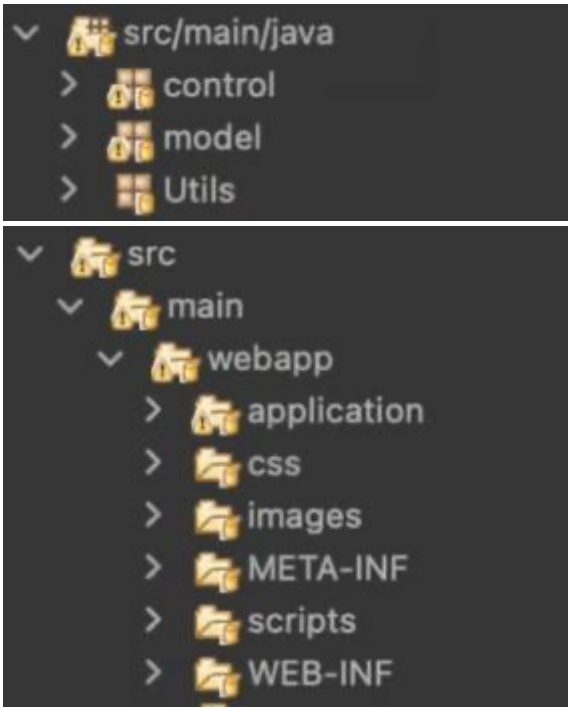
2. Package

Durante la scomposizione del sistema sono stati individuati i seguenti packages:
nel package /java, troviamo:

- **Model:** che fornisce funzionalità riguardo l'interazione del sistema con il database. Al suo interno troviamo classi bean e service per le interazioni con la base di dati;
- **Control:** che contiene le servlet;
- **Utils:** che contiene classi Java di utility per il sistema.

Nel package /webapp troviamo:

- **Application:** che contiene tutte le JSP del sistema;
- **CSS:** che contiene i file di css delle varie pagine del sistema;
- **Image:** che contiene i loghi e le immagini che sono visualizzate nel sistema;
- **Scripts:** che contiene i file in javaScript utilizzati all'interno del sistema.



3. Specifica delle interfacce

3.1 model

Nome Classe:	Studente
Descrizione:	Classe che descrive lo studente
Firma dei Metodi:	+Studente (String, String, String, String, String,String,String) +getNome() : String +getCognome() : String +getEmail() : String +getPassword() : String +getMatricola() : String +getDomanda():String +getRisposta():String +setNome(String) : void +setCognome(String) : void +setEmail(String) : void +setPassword(String) : void +setMatricola(String) : void +setDomanda(String): void +setRisposta(String): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	Professore
Descrizione:	Classe che descrive il professore
Firma dei Metodi:	+Professore(String, String, String, String, String, String, String,String) +getNome() : String +getCognome() : String +getEmail() : String +getPassword() : String +getCodiceProfessore() : String +getDomanda():String +getRisposta():String +setNome(String) : void +setCognome(String) : void +setEmail(String) : void +setPassword(String) : void +setCodiceProfessore(String) : void +setDomanda(String):void +setRisposta(String): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	Ricevimento
Descrizione:	Classe che descrive il ricevimento
Firma dei Metodi:	+Ricevimento(int, String String, String, String) +getGiorno() : String +getOra() : String +getCodice() : String +getNote() : String +getCodiceProfessore() : String +setGiorno(String) : void +setOra(String) : void +setCodice(String) : void +setNote(String) : void +setCodiceProfessore(String) : void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	Insegnamento
Descrizione:	Classe che descrive l'insegnamento
Firma dei Metodi:	+insegnamento(String, String) +getCodice() : int +getNomeInsegnamento() : String +getCodiceProfessore() : String +setCodice(int) : void +setNomeInsegnamento(String) : void +setCodiceProfessore(String) : void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	Prenotazione Ricevimento
Descrizione:	Classe che descrive la prenotazione del ricevimento
Firma dei Metodi:	+prenotazioneRicevimento(int, String, String, String, String, String,String) +getCodice() : int +getStato() : String +getGiorno() : String +getOra() : String +getNote() : String +getCodiceProfessore() : String +getMatricolaStudente() : String +getNomeProfessore():String +getCognomeProfessore():String +setCodice(int) : void +setStato(String) : void +setGiorno(String) : void +setOra(String) : void +setNote(String) : void +setCodiceProfessore(String) : void +setMatricolaStudente(String) : void +setCognomeProfessore(String):void +setNomeProfessore(String):void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	RicevimentoService
Descrizione:	Classe che permette l'interazione con la tabella Ricevimento nel database
Firma dei Metodi:	+ aggiungiRicevimento(Ricevimento) : boolean + modificaRicevimento(Ricevimento) : boolean +modificaRicevimentoBygiornoAndOraAndCodicePofessore (Ricevimento):boolean +rimuoviRicevimento(Ricevimento):boolean +getGiornoEOraRicevimentoByProfessore(String):List<Ricevimento> +getRicevimentiByProfessore(String):List<Ricevimento> +getRicevimentoByProfessoreGiornoOra(String,String,String):Ricevimento
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	PrenotazioneRicevimentoService
Descrizione:	Classe che permette l'interazione con la tabella PrenotazioneRicevimento nel database
Firma dei Metodi:	+aggiungiPrenotazioneRicevimentoByProfessore(PrenotazioneRicevimento) : boolean +aggiungiPrenotazioneRicevimento(PrenotazioneRicevimento):boolean +rimuoviPrenotazione(PrenotazioneRicevimento):boolean +rimuoviPrenotazionePerCodice(int):boolean +modificaPrenotazione(PrenotazioneRicevimento):boolean +ricercaPrenotazione (int) : PrenotazioneRicevimento +stampaPrenotazioni(String):List<PrenotazioniRicevimento> +ricercaPrenotazioniPerProfessore(Professore):List<PrenotazioneRicevimento> +getPrenotazioniAccettateByProfessore(String):List<PrenotazioneRicevimento> +getCodiceProfessoreDiPrenotazione(int): String +stampaCodiceRicevimento(String, String,String):int +getCodicePerGiornoEProfessore(String,String):int
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	StudenteService
Descrizione:	Classe che permette l'interazione con la tabella Studente nel database
Firma dei Metodi:	+ aggiungiStudente(Studente) : int + rimuoviStudente (Studente) : boolean + modificaStudente (Studente) : boolean + cercaStudenteEmail (String) : Studente +trovaPerMatricola(String): Studente +rimuoviStudente(String):boolean
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	ProfessoreService
Descrizione:	Classe che permette l'interazione con la tabella Professore nel database
Firma dei Metodi:	+ aggiungiProfessore(Professore) : int + rimuoviProfessore (Professore) : boolean + modificaProfessore(Professore) : boolean + cercaProfessori(String):ArrayList<Professore> + cercaProfessoreEmail(String): String + getCodiceProfessore():String +getCognomeProfessoreByCodice(String):String +getNomeProfessoreByCodice(String):String +getUfficioProfessore(String,String):String +cercaInsegnamento(String) : String +stampaListaProfessori():List<Professore> +getProfessoreByCodice(String):Professore +rimuoviProfessoreByCodice(String):boolean +cercaProfessoreByCodice(Professore):String
Pre-condizioni	
Post-condizioni	
Invariante	

Nome Classe:	InsegnamentoService
Descrizione:	Classe che permette l'interazione con la tabella Insegnamento nel database
Firma dei Metodi:	+aggiungiInsegnamento(insegnamento) : boolean +rimuoviInsegnamento(insegnamento): boolean +ricercaInsegnamento(String) :Insegnamento +cercaInsegnamentiPerProfessore(String):List<Insegnamento>
Pre-condizioni	
Post-condizioni	
Invariante	

3.2 Control

Nome classe:	LoginServlet
Descrizione:	Servlet che gestisce l'accesso alla piattaforma
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	RegistrazioneServlet
Descrizione:	Servlet che gestisce la registrazione di un professore alla piattaforma
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • Permette una registrazione di un nuovo professore
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	RegistrazioneStudenteServlet
Descrizione:	Servlet che gestisce la registrazione di uno studente alla piattaforma
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • Permette una registrazione di un nuovo Studente
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	RiepilogoRicevimentiServlet
Descrizione:	Servlet che si occupa del riepilogo ricevimenti
Firma dei metodi:	#doGet(HttpServletRequest,HttpServletResponse): void Action: • Mostra l'elenco delle prenotazioni in sospeso per il professore loggato #doPost(HttpServletRequest,HttpServletResponse): void Action: • Permette al professore di accettare o rifiutare una prenotazione
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	ModificaPasswordServlet
Descrizione:	Servlet che si occupa della modifica della password
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • modificaPassword
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	EliminaRicevimentoRiepilogoServlet
Descrizione:	Servlet che si occupa dell'eliminazione del ricevimento nel riepilogo
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • eliminaRicevimento
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	GestioneRicevimentoServlet
Descrizione:	Servlet che si occupa della gestione del ricevimento
Firma dei metodi:	#doGet(HttpServletRequest,HttpServletResponse): void • mostra la pagina di gestione ricevimenti #doPost(HttpServletRequest,HttpServletResponse): void Action: • aggiungi, modifica, elimina ricevimento
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	LogoutServlet
Descrizione:	Servlet che gestisce l'uscita dalla piattaforma
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • invalida la sessione
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	ModificaPasswordServlet
Descrizione:	Servlet che gestisce la modifica della password
Firma dei metodi:	#doPost(HttpServletRequest,HttpServletResponse): void Action: • modifica Password
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	PrenotazioneServlet
Descrizione:	Servlet che gestisce la prenotazione
Firma dei metodi:	#doGet(HttpServletRequest,HttpServletResponse): void Action: • recupera e mostra I ricevimenti disponibili per un professore #doPost(HttpServletRequest,HttpServletResponse): void Action: • permette ad uno studente di prenotare un ricevimento
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	RicevimentiInProgrammaServlet
Descrizione:	servlet che gestisce i ricevimenti programmati
	#doGet(HttpServletRequest,HttpServletResponse): void Action: • mostra I ricevimenti programmati per un professore
Pre-condizioni	
Post-condizioni	
Invariante	

Nome classe:	AjaxSearch
Descrizione:	Servlet che si occupa della ricerca
Firma dei metodi:	#doGet(HttpServletRequest,HttpServletResponse): void Action: • cerca professore
Pre-condizioni	
Post-condizioni	
Invariante	
Nome classe:	EliminaRicevimentoRiepilogoServlet

2.3 Database

Nome classe:	DriverManagerConnectionPool
Descrizione:	Classe che permette l'interazione con il database
Firma dei metodi:	+creaConnessioneDB() : Connection +getConnessione() : Connection +rilasciaConnessione() : void
Pre-condizioni	
Post-condizioni	
Invariante	

2.4 Utils

Nome classe:	PasswordHasher
Descrizione:	Classe di supporto che effettua l'hash della password
Firma dei metodi:	+hashPassord(String):String +verifyPassword(String,String):boolean
Pre-condizioni	
Post-condizioni	
Invariante	