Laurea Triennale in informatica-Università di Salerno

Corso di *Ingegneria del Software*-

Prof. Carmine Gravino, Prof.ssa Filomena Ferrucci

EasyAgreement

SDD: System Design Document

Versione **1.1**

Data	11/12/2019
Destinatario	Prof. Carmine Gravino, Prof.ssa Filomena Ferrucci
Presentato da	Tutti i team member
Approvato da	Francesco Califano, Domenico Marino

Revision History

Data	Versione	Descrizione	Autori
25/11/2019	0.1	Prima stesura	[Tutti]
26/11/2019	0.2	Aggiunta Introduzione	[Tutti]
04/12/2019	1.0	Completamento del documento	[Tutti]
11/12/2019	1.1	Correzione formattazione del documento	Marco Ciano, Armando Soddisfatto, Veronica Volpicelli

Sommario

1. Introduzione

1.1 Obiettivi del sistema

Il sistema che si vuole realizzare ha come scopo la semplificazione del processo di comunicazione tra studente, tutor accademico e tutor esterno, inerente alla compilazione ed accettazione del modulo del Learning Agreement.

I nostri obiettivi riguardano la realizzazione di un sistema che permetta di avere una massima automazione della fase di compilazione e gestione della modulistica. Nel caso del Learning Agreement si vuole ridurre il più possibile la documentazione cartacea, ottimizzare la compilazione e l'approvazione del Learning Agreement e fornire uno strumento di comunicazione istantaneo che agevoli lo scambio di informazioni tra lo studente, il tutor accademico e il tutor esterno.

Si vuole, inoltre, agevolare la gestione ed il controllo delle versioni durante l'intero ciclo di vita del Learning Agreement.

1.2 Design Goals

Rank / Priorità	ID Design Goal	Descrizione design goal	Categoria	Origine	Trade off
1	DG_1	Il prodotto software sviluppato deve avere tempi di risposta minimi nello svolgimento delle funzionalità offerte, trasmettendo all'utente sensazioni di fluidità e immediatezza per i dati richiesti.	Performance-Response Time	RNF_3	Response Time vs Memory: Se il software non rispetta i requisiti di tempo di risposta è necessario utilizzare più memoria per velocizzare il sistema.
2	DG_2	Il prodotto software deve essere disponibile in qualsiasi momento della giornata, tranne per i periodi di manutenzione. La memoria necessaria al funzionamento del sistema dipende da quella utilizzata per il mantenimento del Database.	Performance – Memory	RNF_3	Performance vs Maintenance: La manutenibilità sarà preferita alla performance in modo da facilitare gli sviluppatori nei processi di aggiornamento del software.
3	DG_3	Il sistema informerà l'utente in caso di immissione di input non validi	Dependability - Robustness	RNF_2	

		tramite messaggi.			
4	DG_4	Il sistema deve garantire l'affidabilità dei servizi proposti. Il login dell'utente sarà gestito in modo affidabile.	Dependability – Dependability	RNF_2	
5	DG_5	L'accesso al sistema per tutti gli utenti avviene tramite username e password. Ogni utente può svolgere le azioni in totale sicurezza.	Dependability - Security	RNF_2	
6	DG_6	Il sistema sarà disponibile a tutti gli utenti che ne richiederanno l'utilizzo e che sono registrati.	Dependability – Availability	RNF_2	Availability vs Fault Tollerance: Il sistema deve sempre essere disponibile all'utente in caso di errore in una funzionalità, anche al costo di rendere non disponibile quest'ultima per un lasso di tempo.
7	DG_7	Il sistema permette il salvataggio dei dati in modo automatico, così da evitare problemi in caso di fallimenti.	Dependability - Fault Tollerance	RNF_2	
8	DG_8	È stimato un costo di 50 ore per project member per la progettazione e lo sviluppo del sistema, per un totale di 400 ore.	Cost - Developement Cost		Developement Cost vs Security: Il sistema non permette di sviluppare un prodotto software che sia

					simultaneamente sicuro e costi poco.
9	DG_9	Il sistema funziona solo per il dipartimento di informatica e per l'Università degli studi di Salerno, ma è adattabile a più dipartimenti o a più università modificando i relativi dati.	Maintenance - Adaptability	RNF_4	
10	DG_10	Il sistema presenterà il supporto multilingua, in particolare per la lingua inglese, per dare all'utente un supporto maggiore per quanto riguarda l'usabilità del sistema. Inoltre, il sistema deve poter adattarsi a nuovi cambiamenti.	Maintenance - Extensibility	RNF_4	Memory vs Extensibility: Il sistema deve permettere l'estensibilità a discapito della memoria utilizzata, in modo tale da permettere al cliente di richiedere agli sviluppatori nuove funzionalità.
11	DG_11	Il sistema non permette agli utenti di modificare le sue funzionalità.	Maintenance - Changeability	RNF_4	
12	DG_12	L'interazione con il sistema avviene tramite browser poiché sviluppato come una web application. È accessibile da qualunque dispositivo che abbia un browser qualsiasi.	Maintenance – Portability	RNF_4	

13	DG_13	I requisiti sono tracciabili attraverso la matrice di tracciabilità.	Maintenance – Requirements Traceability	RNF_4	
14	DG_14	Il sistema prevede un'interfaccia ordinata, user-friendly e di facile utilizzo.	End User-Usability	RNF_1	
15	DG_15	Il sistema si rende utile in quanto velocizza la gestione del Learning Agreement e dei documenti gestendo i dati in formato digitale e non in maniera cartacea.	End User-Utility	RNF_1	

1.3 Definizioni, acronimi e abbreviazioni

EA: Easy Agreement.

RAD: Requirements Analysis Document.

SDD: System Design Document.

User-Friendly: letteralmente "amichevole per l'utente", di facile utilizzo anche per chi non è esperto.

DB: DataBase.

MongoDB: è un DBMS non relazionale orientato ai documenti. Sfrutta il formato JSON per la memorizzazione e rappresentazione dei dati

DBMS: Database Management System.

JSON: JavaScript Object Notation, è un formato adatto all'interscambio di dati tra applicazioni client/server.

MVC: Model View Controller.

Event-driven: meccanismo per il flusso di controllo globale. Il flusso globale del sistema è guidato da una serie di eventi.

1.4 Riferimenti

EA_RAD_V_2.1

Slide del corso, presenti sulla piattaforma e-learning,

Libro:

-- Object-Oriented Software Engineering (Using UML, Patterns, and Java) Third Edition

Autori:

-- Bernd Bruegge & Allen H. Dutoit

1.5 Panoramica

Capitolo 1: contiene l'introduzione con gli obiettivi del sistema, i design goals e un elenco di definizioni, acronimi e abbreviazioni utili alla comprensione dell'intera documentazione.

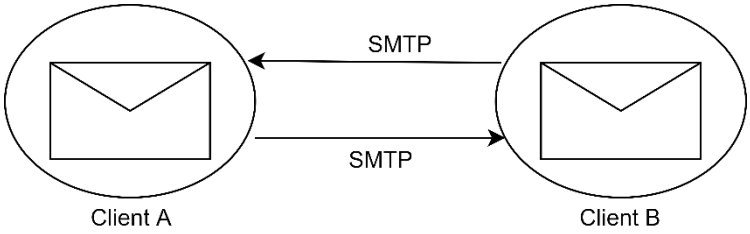
Capitolo 2: descrive le funzionalità offerte dal sistema corrente.

Capitolo 3: viene presentata l'architettura del sistema proposto, nella quale sarà gestita la decomposizione in sottosistemi, il mapping hardware/software, i dati persistenti, il controllo degli accessi e sicurezza, il controllo del flusso globale del sistema, le condizioni limite.

Capitolo 4: vengono presentati i servizi offerti dai sottosistemi.

2. Architettura del Sistema corrente

Non esiste, attualmente, un prodotto software finalizzato alla gestione del Learning Agreement per la mobilità per Traineeship dedicato all'Università degli Studi di Salerno. La gestione del Learning Agreement viene infatti eseguita manualmente collegandosi al sito dell'università, scaricando e stampando il documento. Una volta compilato va scannerizzato per inviarlo al tutor accademico tramite e-mail, che revisiona la compilazione dello studente e compila la sua parte stampando e firmando il documento, per poi scannerizzarlo di nuovo per inviarlo al tutor esterno. Il tutor esterno esegue lo stesso procedimento eseguito dal tutor accademico compilando la propria parte. In caso di approvazione completa il documento viene firmato da tutte le parti e lo studente può partire per il Traineeship. In caso di disapprovazione, il Learning Agreement ritorna allo studente per apportare le modifiche necessarie all'approvazione. Lo schema sotto proposto descrive il funzionamento del sistema corrente tra due generici attori attraverso lo scambio di e-mail.



Il sistema risulta sconveniente per i tutor, il cui tempo a disposizione è spesso limitato, che si trovano ad elaborare molte richieste gestite tramite e-mail. Per agevolare le operazioni citate, abbiamo deciso di sviluppare una piattaforma online che consenta di svolgere le procedure descritte in modo più efficiente e concentrato. Il sistema provvederà anche ad un sistema di messaggistica per facilitare la comunicazione fra i vari attori all'interno di un'unica piattaforma.

3. Architettura del Sistema proposto

3.1 Panoramica

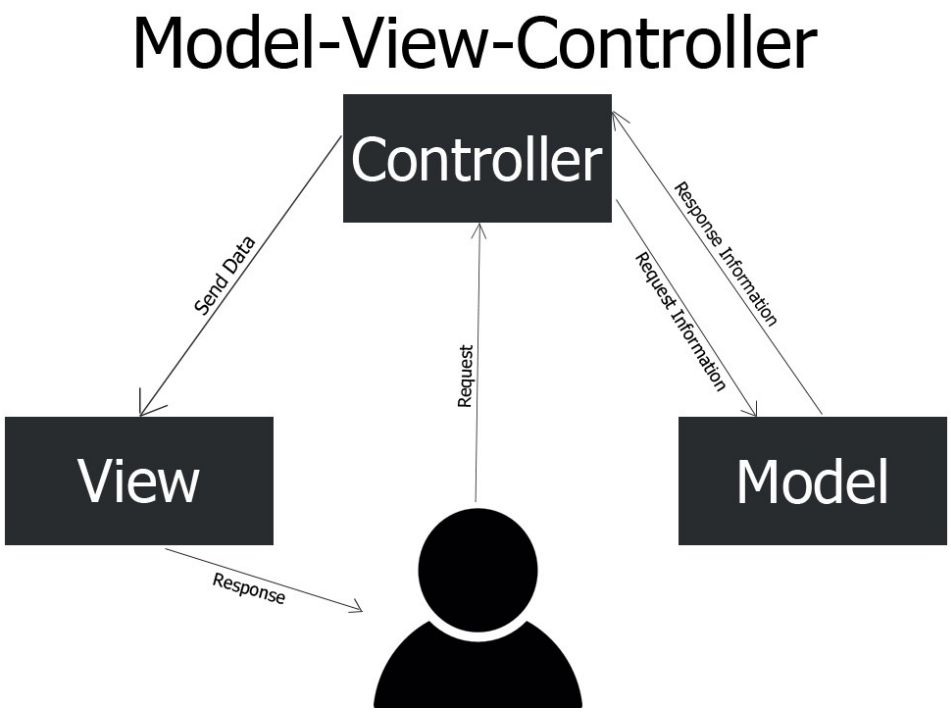
Il sistema proposto consiste in una piattaforma Web che si rivolge agli studenti che partecipano al bando ERASMUS+ per Traineeship, ai tutor accademici ed ai tutor esterni delle organizzazioni ospitanti. Gli utenti che avranno accesso alla piattaforma saranno quindi: studente, tutor accademico, tutor esterno e un amministratore. Tutti gli utenti potranno effettuare il login e il logout, mentre solo gli studenti e i tutor accademici potranno effettuare la registrazione, in quanto i tutor esterni saranno inseriti manualmente dall'amministratore.
Una volta effettuato il login la piattaforma metterà a disposizione diverse funzionalità a seconda del tipo di utente ad aver effettuato l'accesso. Lo studente potrà compilare il Learning Agreement e inviarlo per l'approvazione al tutor accademico. Il sistema a questo punto genererà una nuova richiesta contenente le informazioni dello studente e la sua compilazione del documento che sarà inviata al tutor accademico che potrà decidere di rifiutarla o approvarla. In caso di approvazione il tutor accademico dovrà compilare la propria sezione del documento e a sua volta inoltrare la richiesta al tutor esterno, che avrà anche lui la possibilità di rifiutarla o approvarla compilando a sua volta la sezione dedicata del documento. In caso di rifiuto lo studente dovrà ricompilare il documento e ripetere le operazioni. Lo studente avrà la possibilità di caricare il proprio Curriculum Vitae e il proprio documento d'identità, che potranno essere visualizzati da parte dei tutor nei dettagli della richiesta inviata. Tutti gli utenti, eccetto l'amministratore, avranno accesso ad un sistema di messaggistica istantanea per poter comunicare tra di loro. L'architettura utilizzata è di tipo *repository*, in quanto adatta quando il sistema è basato sull'archivio dei dati e la loro modifica è frequente e coinvolge più parti. In questo tipo di architettura i sottosistemi che compongono il software accedono e modificano una singola struttura dati, chiamata repository; questo fa sì che i vari sottosistemi siano fra di loro relativamente indipendenti, in quanto interagiscono solo mediante il repository. Come pattern è utilizzato l'MVC (*Model-View-Controller*), un pattern architetturale molto diffuso nello sviluppo di sistemi software, in particolare nell'ambito object-oriented, in grado di separare la logica di presentazione dei dati, dalla logica di business.

3.2 Decomposizione in sottosistemi

3.2.1 Decomposizione in layer

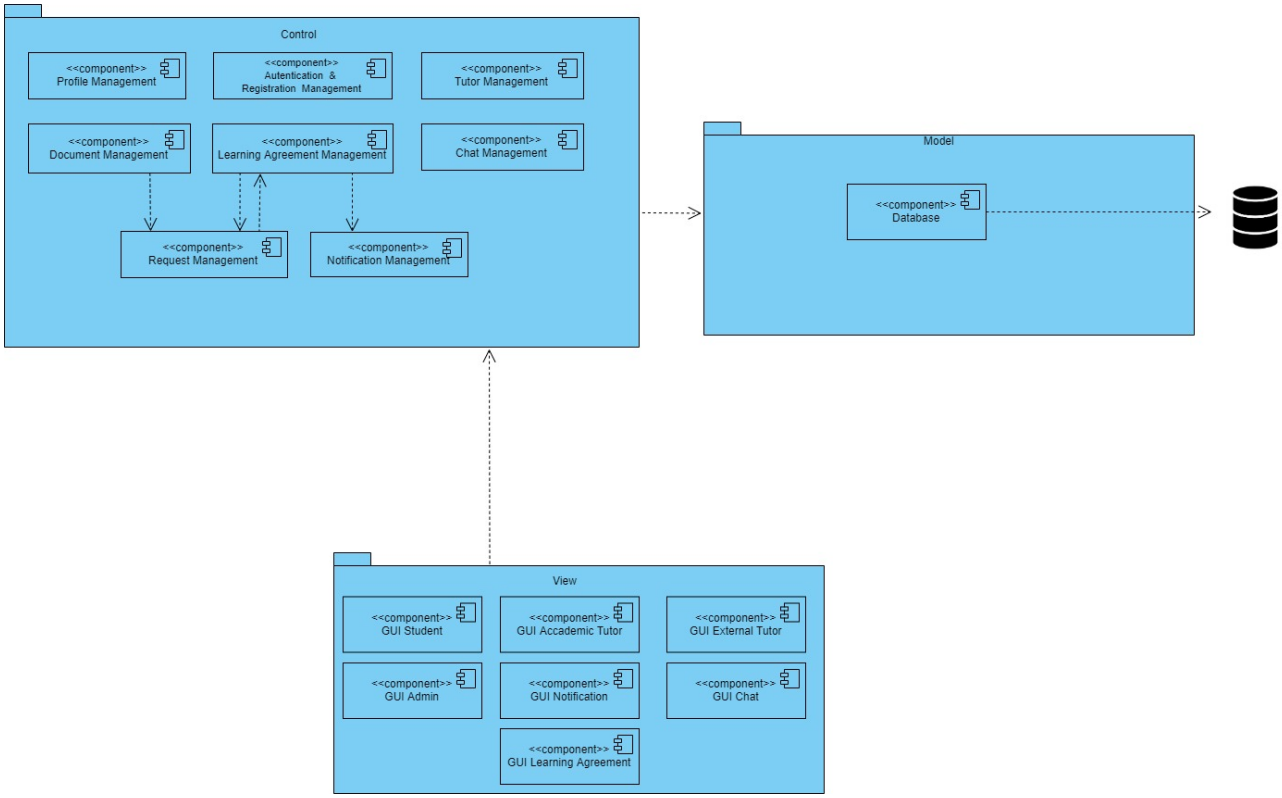
Il sistema prevede una decomposizione in tre layer. Quest'ultimi gestiscono diversi aspetti e funzionalità del sistema:

- **Model:** si occupa della gestione dei dati e contiene i metodi d'accesso ad essi.
- **View:** si occupa di visualizzare i dati all'utente e gestisce l'interazione con quest'ultimo;
- **Controller:** si occupa della logica del sistema. Riceve i comandi dell'utente attraverso la view e reagisce eseguendo delle operazioni che possono interessare il model e che portano ad un cambiamento di stato del view.



3.2.2 Decomposizione in sottosistemi

Dopo un'analisi effettuata sul sistema, abbiamo deciso di suddividerlo nei seguenti sottosistemi in modo tale da utilizzare un'architettura aperta per motivi di efficienza.



Il livello View prevede la gestione di sette sottosistemi, identificati come oggetti "boundary" nel RAD.

- GUI Student
- GUI Academic Tutor
- GUI External Tutor
- GUI Admin
- GUI Notification
- GUI Chat
- GUI Learning Agreement

Il livello Control prevede la gestione di otto sottosistemi:

- Profile Management
- Document Management

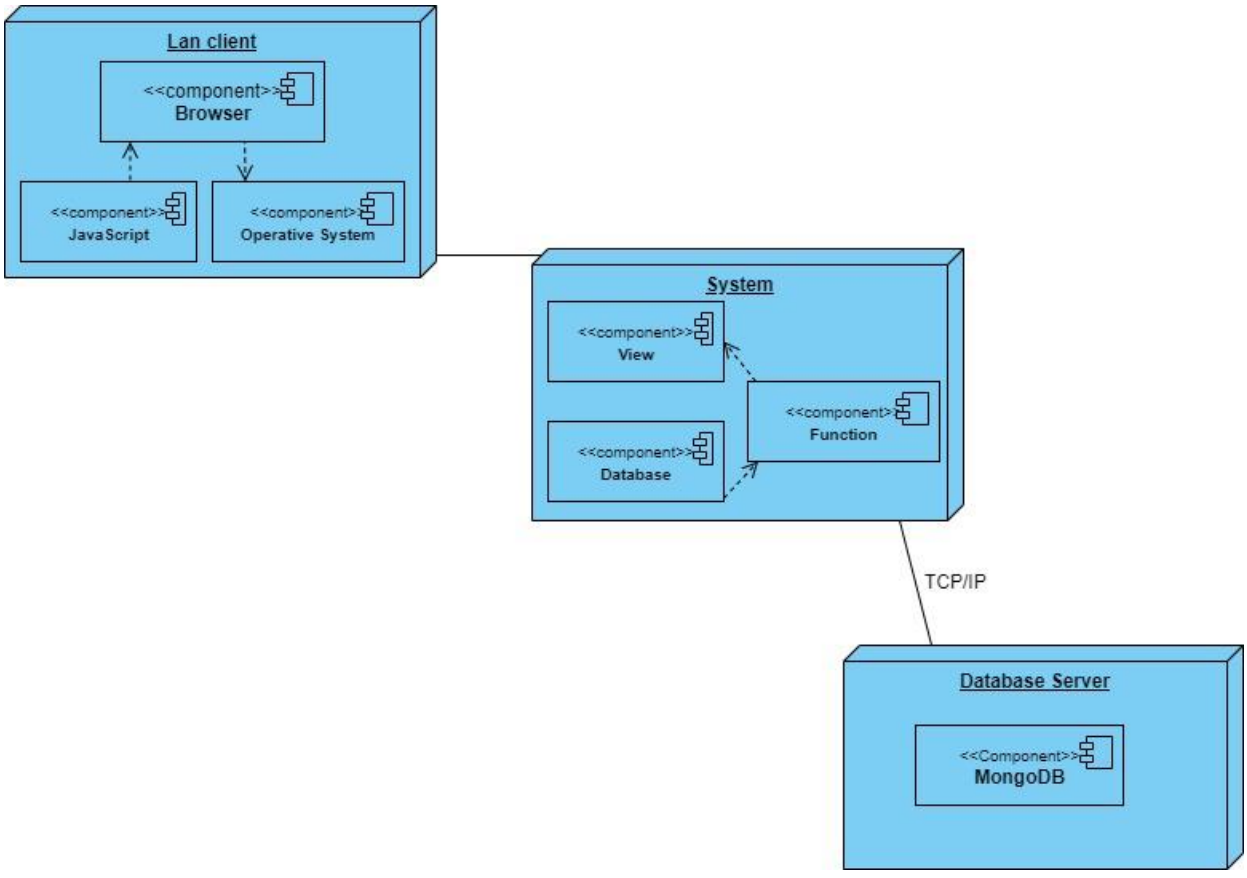
- Autentication & Registration Management
- Tutor Management
- Chat Management
- Notification Management
- Learning Agreement Management
- Request Management

Il livello Model prevede la gestione di un sottosistema:

- Database

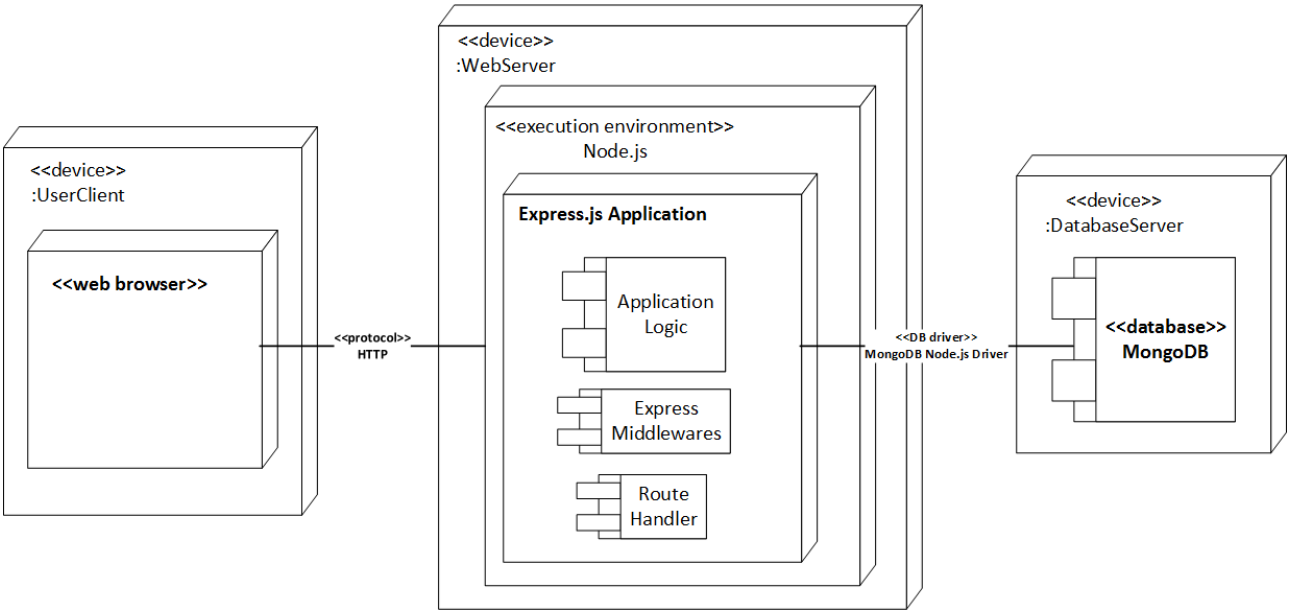
Deployment Diagram

L'utente (Lan Client), per interagire con l'interfaccia del sistema necessita di un browser capace di interpretare il linguaggio javascript. Il tier del Lan Client connette lo strato di view del System sul quale vengono eseguite le funzioni del completamento degli obiettivi del Lan Client. Infine, la parte Server racchiude e gestisce la persistenza dei dati.



3.3 Mapping Hardware/Software

Il sistema, per garantire i servizi, utilizza un'architettura di tipo Client-Server. Il Client è rappresentato dal Browser utilizzato dall'utente. Lato Server è presente un ambiente run-time, open-source e multipiattaforma chiamato **Node.js**, grazie a esso abbiamo la possibilità di eseguire il codice JavaScript lato server e produrre pagine web dinamiche prima che le stesse siano inviate al web browser dell'utente. Abbiamo scelto Node.js poiché possiede un'architettura ad eventi capace di realizzare operazioni input/output (I/O) asincrone, questa scelta di design punta a ottimizzare la velocità di esecuzione e la scalabilità nella nostra applicazione web. Dal punto di vista di memorizzazione dei dati, poiché utilizziamo il formato JSON, la soluzione migliore è l'utilizzo di **MongoDB,** un database non relazionale (NoSQL).



3.4 Gestione dei dati persistenti

Per la memorizzazione dei dati è stato scelto un Database non relazionale, utilizzando il DBMS orientato ai documenti MongoDB. Il vantaggio della scelta di un Database orientato ai documenti consiste nella possibilità di conservare le informazioni in documenti JSON, ciò consente di memorizzare molti dati, anche annidati uno nell'altro; i documenti sono raggruppati in *collezioni* che possono essere anche eterogenee, ovvero non c'è uno schema fisso per i documenti che ne fanno parte. Un'organizzazione di questo tipo consente di apportare modifiche agli attributi di un'entità anche in corso d'opera, inoltre le tecniche di indicizzazione garantiscono alte performance di accesso ai dati nonostante la possibile assenza di relazioni tra gli oggetti.

3.4.1 Descrizione delle entità persistenti

Student

- StudentID**:** string
- DegreeCourse: string
- Address: string
- City: string
- E-mail: string
- Password: string
- Surname: string
- Name: string
- CV: PDF
- IDCard: PDF

La collection Student contiene le informazioni anagrafiche dello studente, la sua e-mail, la password, il Curriculum Vitae e la Carta d'identità. Uno studente è identificato univocamente dalla sua matricola (non possono esistere più studenti con la stessa matricola).

Administrator

- E-mail: string
- Password: string
- Surname: string
- Name: string

La collection Administrator contiene le informazioni anagrafiche più la sua e-mail e la password.

Academic Tutor

- E-mail: string
- Password: string
- Surname: string
- Name: string
- Department: string

La collection Academic Tutor contiene le informazioni anagrafiche del tutor accademico compreso il dipartimento l'e-mail e la password.

External Tutor

- E-mail: string
- Password: string
- Surname: string
- Name: string
- Organization: string

La collection External Tutor contiene le informazioni anagrafiche del tutor esterno compreso il l'organizzazione di cui fa parte, l'e-mail e la password.

Host Organization

- ErasmusCode: string
- Faculty: string
- Address: string
- Size: string
- Country: string
- Contacts: string
- Name: string

La collection Host Organization contiene le informazioni dell'organizzazione compreso il codice erasmus e la facoltà.

Learning Agreement

- Document: PDF
- LearningAgreementID: string
- AssociatedStudentID: string

- State: string
- Date: data

La collection Learning Agreement conterrà il PDF del learning agreement con la data di compilazione, il relativo id dello studente di appartenenza e un identificativo univoco.

Message

- SenderID: string
- RecipientID: string
- Text: string

La collection Message conterrà il testo del messaggio con il relativo id del mittente e del destinatario.

Notification

- AssociatedID: string
- NotificationID: int
- Text: string

La collection Notification conterrà il testo delle notifiche con il relativo id di appartenenza ed un identificativo univoco.

Request

- StudentID: string
- AcademicTutorID: string
- ExternalTutorID: string
- LAID: string

La collection Request conterrà le richieste degli studenti con i relativi id di appartenenza delle parti e un identificativo univoco del Learning Agreement.

3.5 Controllo degli accessi e sicurezza

All' interno del sistema "EA Easy Agreement", gli attori hanno il permesso di eseguire diverse operazioni. Grazie a questo sistema ci è possibile gestire in sicurezza gli accessi dei vari utenti alle funzionalità del software. Per approfondire al meglio il controllo degli accessi, viene utilizzata una matrice degli accessi, all'interno della quale le righe rappresentano gli attori e le colonne le classi. Ogni entry (attore, classe) della matrice contiene le operazioni consentite da quell'attore sulle istanze di quella classe.

Sottosistema Attore	Profilo	Learning Agreement	Richiesta	Chat
Studente	• Visualizzazione profilo • Modifica profilo anagrafico	• Compilazione • Visualizzazione • Controllo stato • Visualizzazione versioni precedenti • Modifica • Invio al Tutor Accademico	• Generazione e richiesta	• Creazione chat • Visualizzazione chat
Tutor Accademico	• Visualizzazione profilo • Modifica password	• Visualizzazione • Determina esito richiesta • Visualizzazione versioni precedenti • Modifica • Invio richiesta dello Studente al Tutor Esterno	• Visualizzazione richiesta • Visualizza info richiesta	• Creazione chat • Visualizzazione chat
Tutor Esterno	• Visualizzazione profilo • Modifica password	• Visualizzazione • Determina esito richiesta • Modifica • Visualizzazione versioni precedenti	• Visualizzazione richiesta • Visualizza info richiesta	• Creazione chat • Visualizzazione chat
Amministratore	• Modifica password	-	-	-

Sottosistema Attore	Messaggio	Notifica	Documento	Gestione
Studente	• Invio messaggio	• Ricezione notifica	• Visualizzazione documento	-

	• Rimozione messaggio • Ricezione messaggio • Modifica messaggio	• Visualizzazione elenco notifiche • Rimozione notifica • Generazione notifica	• Upload documento • Rimozione documento	
Tutor Accademico	• Invio messaggio • Rimozione messaggio • Ricezione messaggio • Modifica messaggio	• Ricezione notifica • Visualizzazione elenco notifiche • Rimozione notifica • Generazione notifica	• Visualizzazione documento	-
Tutor Esterno	• Invio messaggio • Rimozione messaggio • Ricezione messaggio • Modifica messaggio	• Ricezione notifica • Visualizzazione elenco notifiche • Rimozione notifica • Generazione notifica	• Visualizzazione documento	-
Amministratore	-	-	-	• Inserimento Tutor esterno • Rimozione Tutor esterno • Visualizzazione Tutor e Organizzazioni Ospitanti • Visualizzazione info Tutor • Visualizzazione info Organizzazione Ospitante

3.6 Controllo flusso globale del sistema

Il flusso del sistema "Easy Agreement" fornisce una funzionalità che richiede una continua interazione da parte dell'utente, per cui il controllo del flusso globale del sistema è di tipo "Event-driven" control, ovvero guidato dagli eventi. Il ciclo principale attende un evento esterno, quando un evento si verifica è spedito all'oggetto appropriato (sulla base dell'info associata all'evento). Il controllo risiede in un dispatcher che chiama le funzioni di sottosistema. Tale meccanismo è da considerarsi flessibile e ideale per le interfacce utenti.

3.7 Condizioni Boundary

START-UP

Per il primo start-up del sistema "Easy Agreement" è necessario l'avvio della runtime di javascript "Node.js" che fornisce il servizio d'esecuzione di codice Javascript per la gestione dei dati persistenti. Successivamente ci sarà:

- Un'interfaccia per il SIGN UP che permetta di effettuare la registrazione, attraverso opportuni campi (e-mail, password, tipologia utente) nel sistema. È possibile registrarsi nel sistema con una delle seguenti tipologie: *Studente* (utente che vuole compilare il modulo Learning Agreement per partecipare al bando ERASMUS+ per traineeship), *Tutor Accademico* (utente che fa parte del ciclo di vita del Learning Agreement; in particolare per la fase di approvazione/disapprovazione del Learning Agreement)
- Un'interfaccia per il LOGIN che permette di autenticarsi attraverso opportune credenziali (e-mail e password). Una volta effettuato l'accesso, "Easy Agreement" presenta all'utente la propria HomePage dalla quale è possibile usufruire ed eseguire tutte le operazioni che la piattaforma mette a disposizione (Es. Allo studente permette la compilazione del Learning Agreement, ai tutor permette l'approvazione/disapprovazione del Learning Agreement, ecc…).

TERMINAZIONE

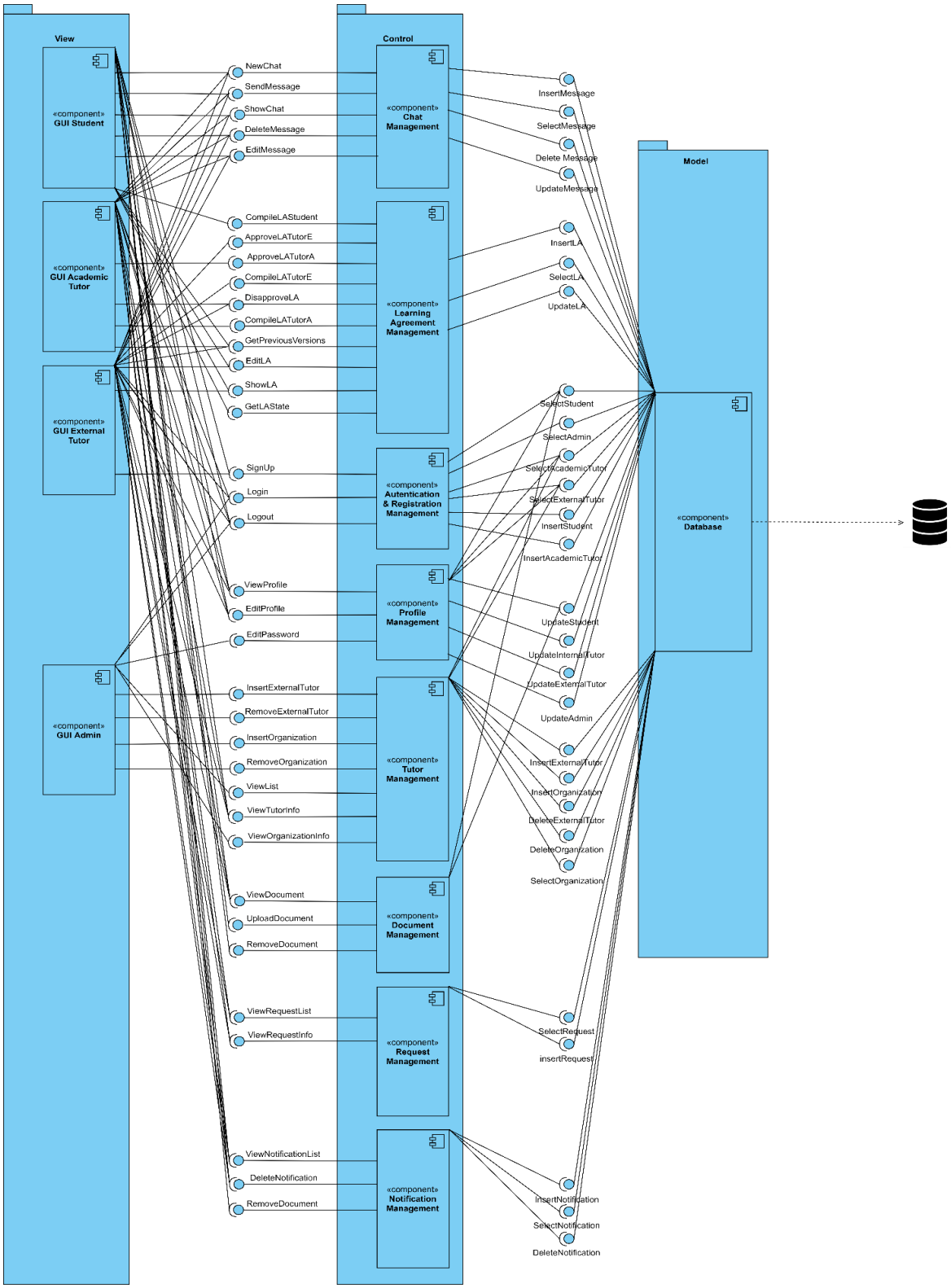
Al momento della corretta chiusura dell'applicazione, si ha la terminazione del sistema accompagnata da un regolare log-out che permetta all'utente di scollegarsi dall'intero sistema.

FALLIMENTO

1. Nel caso di guasti dovuti al sovraccarico del database con un eventuale fallimento dello stesso, è possibile utilizzare la funzionalità di sincronizzazione dei dati (detta "Replica") di MongoDB per avere più copie dei dati e poterne effettuare così la rigenerazione.

2. Nel caso in cui si verifichi un'interruzione inaspettata dell'alimentazione, non sono previsti metodi che ripristino lo stato del sistema precedente allo spegnimento non voluto.
3. Altro caso di fallimento potrebbe essere dovuto ad un errore critico nell'hardware, per il quale non è prevista alcuna contromisura.
4. Ancora, un altro caso di fallimento potrebbe essere causato dal software stesso che provoca una chiusura non prevista per errori durante la fase d'implementazione. Non essendo stipulate politiche correttive, le uniche operazioni disponibili e consentite sono quelle di chiusura e riavvio successivo del sistema.

4. Servizi dei sottosistemi



GUI Student offre 23 servizi all'interfaccia Control:

- ViewList
- ViewTutorInfo
- ViewOrganizationInfo
- CompileLAStudent
- GetLAState
- GetPreviousVersion
- EditLA
- ShowLA
- Login
- Logout
- SignUp
- ViewNotificationList
- DeleteNotification
- ViewProfile
- EditProfile
- ViewDocument
- UploadDocument
- RemoveDocument
- NewChat
- SendMessage
- ShowChat
- DeleteMessage
- EditMessage

GUI Academic Tutor offre 24 servizi all'interfaccia Control:

- ViewList
- ViewTutorInfo
- ViewOrganizationInfo
- CompileLATutorA
- ApproveLATutorA

- DisapproveLA
- GetPreviousVersion
- EditLA
- ShowLA
- Login
- Logout
- SignUp
- ViewNotificationList
- DeleteNotification
- ViewProfile
- EditProfile
- ViewRequestList
- ViewRequestInfo
- ViewDocument
- NewChat
- SendMessage
- ShowChat
- DeleteMessage
- EditMessage

GUI External Tutor offre 23 servizi all'interfaccia Control:

- ViewList
- ViewTutorInfo
- ViewOrganizationInfo
- CompileLATutorE
- ApproveLATutorE
- DisapproveLA
- GetPreviousVersion
- EditLA
- ShowLA
- Login

- Logout
- ViewNotificationList
- DeleteNotification
- ViewProfile
- EditProfile
- ViewRequestList
- ViewRequestInfo
- ViewDocument
- NewChat
- SendMessage
- ShowChat
- DeleteMessage
- EditMessage

GUI Admin offre 10 servizi all'interfaccia Control:

- InsertExternalTutor
- RemoveExternalTutor
- InsertOrganization
- RemoveOrganization
- ViewList
- ViewTutorInfo
- ViewOrganizationInfo
- Login
- Logout
- EditPassword

Chat Management offre 4 servizi all'interfaccia Model:

- InsertMessage
- SelectMessage
- DeleteMessage
- UpdateMessage

Learning Agreement Management offre 3 servizi all'interfaccia Model:

- InsertLA
- SelectLA
- UpdateLA

Autentication & Registration Management offre 6 servizi all'interfaccia Model:

- SelectStudent
- SelectAdmin
- SelectAcademicTutor
- SelectExternalTutor
- InsertStudent
- InsertAcademicTutor

Profile Management offre 7 servizi all'interfaccia Model:

- SelectStudent
- SelectAcademicTutor
- SelectExternalTutor
- UpdateStudent
- UpdateAcademicTutor
- UpdateExternalTutor
- UpdateAdmin

Tutor Management offre 7 servizi all'interfaccia Model:

- InsertExternalTutor
- InsertOrganization
- DeleteExternalTutor
- DeleteOrganization
- SelectAcademicTutor
- SelectExternalTutor
- SelectOrganization

Document Management offre 2 servizi all'interfaccia Model:

- SelectStudent
- UpdateStudent

Request Management offre 2 servizi all'interfaccia Model:

- SelectRequest
- InsertRequest

Notification Management offre 3 servizi all'interfaccia Model:

- InsertNotification
- SelectNotification
- DeleteNotification