System Design Document

Studenti

Colella Maria Immacolata	0512110431
Mattiello Michele	0512110185
Sorrentino Carlo	0512110884

System Design Document

System Design Document

1. Introduzione

1.1 Scopo del sistema

StudyHall è un applicativo il cui scopo è quello di offrire un servizio di prenotazione posti presso le aule studio di UNISA. Grazie a StudyHall gli studenti gestiranno con facilità e al meglio il loro tempo all'interno del campus, potendo usufruire delle aule messe a disposizione. Le associazioni si occupano della completa gestione di esse, assicurando agli studenti un ambiente di lavoro sano e proficiente. Gli studenti potranno scegliere l'aula che più preferiscono in base all'edificio più vicino alla loro facoltà o alle loro esigenze.

1.2 Design Goals

Il sistema è stato progettato considerando i seguenti obiettivi di design: Affidabilità, Legali, Manutenzione, Performance e Usabilità.

1.2.1 Criteri di Affidabilità

Attendibilità	Non deve esserci differenza tra il comportamento atteso e quello osservato Priorità : Alta
Capacità di operare in condizioni di errore (tolleranza ai fault)	Il sistema deve essere in grado di gestire correttamente l'immissione di eventuali input errati e/o altri casi di errore e di rispondere a questi in maniera adeguata Questi controlli verranno implementati lato server mentre per i form più utilizzati dall'utenza saranno effettuati anche controlli sull'applicazione per evitare di sovraccaricare troppo il server quando non necessario. Entrambi i controlli verranno effettuati mediante TypeScript perché sia server che client sono scritti nello stesso linguaggio per favorire l'interscambiabilità tra front-end e back-end Priorità : Alta

Robustezza	Il sistema deve comportarsi in modo ragionevole in situazioni impreviste, non contemplate dalle specifiche Priorità : Alta
Sicurezza dei dati	Le credenziali di accesso devono essere protette quindi criptate all'interno del database (Firebase) per evitare che terzi vi accedano e di queste non deve essere possibile il recupero ma soltanto la sostituzione o il reset Della cifratura di questi dati sensibili se ne occupa Firebase attraverso Firebase Password Hashing utilizzando l'algoritmo di cifratura "Scrypt" Priorità : Alta
Sicurezza di Comunicazione	La comunicazione tra server e client deve essere cifrata Priorità : Alta

1.2.2 Criteri di Manutenibilità

Modificabilità	Il sistema deve poter essere facilmente modificabile in modo da correggere eventuali errori. Priorità : Media
Descrizione della logica & Identificazione delle variabili	Il codice dell'applicazione deve avere commenti adatti che descrivano le funzionalità principali. Inoltre, le variabili usate nel codice devono avere nomi indicativi del loro scopo. Priorità : Media

1.2.3 Criteri di Performance

Tempo di Risposta	Il sistema deve essere reattivo per tutte le operazioni avendo un tempo di risposta inferiore ai 3 secondi Priorità: Media
Usabilità	Il sistema deve essere in grado di gestire e sostenere almeno un carico massimo di 300 utenti che operano contemporaneamente Priorità: Media

1.2.4 Criteri di Usabilità

Interattività Utente-Sistema (Grafica)	Il sistema deve essere implementato con un'interfaccia grafica minimalista Priorità: Alta
Intuitività delle Operazioni (Icone)	Il sistema deve utilizzare icone intuitive per operazioni note. Priorità: Media
Evidenziare gli Errori	Il sistema deve essere in grado, in caso di errore di compilazione di un form, di mostrare all'utente il campo in cui si è commesso l'errore evidenziandolo di rosso e facendo comparire un testo con dei suggerimenti su cosa inserire. Priorità: Alta

1.3 Acronimi e Abbreviazioni

GESTIONE DEI FAULT : Capacità di un sistema di non subire avarie, interruzioni di servizio, anche in presenza di guasti ove per fault si intende un malfunzionamento del software causato da uno o più errori;

THROUGHPUT : Frequenza con cui vengono trasmessi i dati ovvero quantità di dati spostati con successo da un luogo all'altro in un determinato periodo;

RAD : Documento di Analisi dei Requisiti;

PS : Problem Statement;

1.4 Riferimenti

- Testo Object-Oriented Software Engineering Using UML, Patterns, and Java
- RAD (StudyHall)
- PS (StudyHall)
- Docs Modalità di autenticazione di Esse3 (https://esse3web.unisa.it/e3rest/docs/)
- Docs Modalità di autenticazione di Firebase (https://firebase.google.com/docs/reference/rest/auth)

1.5 Panoramica

Il System Design Document (SDD) è diviso in cinque parti.

1. Nella prima parte si affronta l'obiettivo del sistema con i relativi Design Goals, altre informazioni riguardanti le funzionalità e/o requisiti che l'applicativo deve fornire possono essere trovate nel Requirement Analysis Document (RAD) mentre una descrizione concisa del problema che deve essere risolto oltre ad una panoramica generale può essere trovata nel Problem Statement (PS).
2. Nella seconda parte vengono illustrate le caratteristiche del sistema corrente.
3. La terza è dedita alla descrizione del software proposto:
   - DECOMPOSIZIONE IN SOTTOSISTEMI: il sistema viene suddiviso in sottosistemi i quali vengono descritti e associati alle responsabilità che hanno verso gli altri sottosistemi;
   - MAPPING HARDWARE / SOFTWARE: si decide l'hardware dove il sistema deve funzionare e si mappano le componenti su di essa.
   - GESTIONE DEI DATI PERSISTENTI: descrive i dati memorizzati e il database usato per memorizzarli.
   - CONTROLLO DEGLI ACCESSI E SICUREZZA: vengono esplicate per ogni tipologia di utente le possibili azioni che essi possono svolgere.
   - CONTROLLO DEL SOFTWARE GLOBALE: descrive la matrice degli accessi.

4. La quarta parte è per i servizi dei sottosistemi che descrive in termini di operazioni i servizi forniti dai singoli sottosistemi
5. L'ultima parte del documento è il glossario che si occupa di illustrare il significato di alcuni termini utilizzati.

2. Architettura Software Attuale

2.1 Panoramica

Non vi è un'architettura software che offre servizi come quelli proposti dal nostro applicativo.

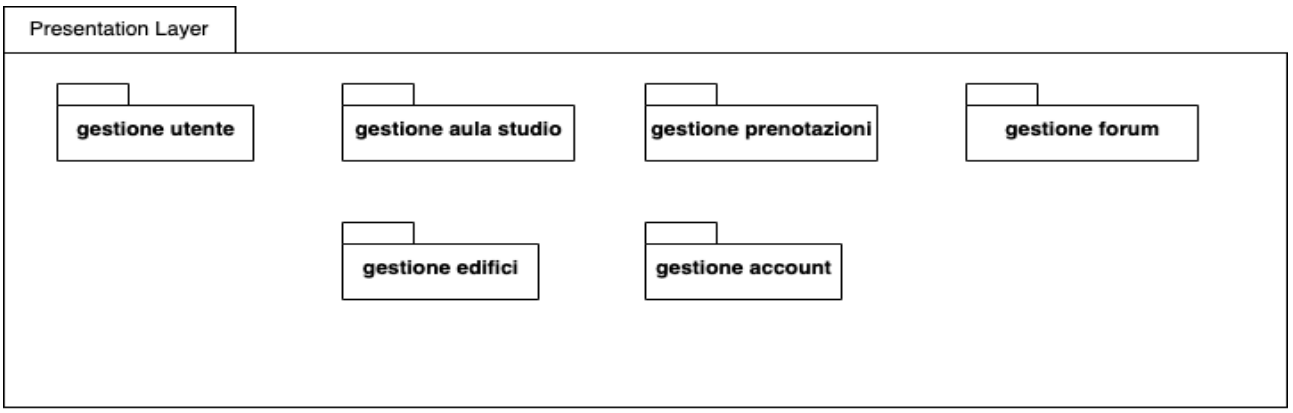
3. Architettura Software Proposto

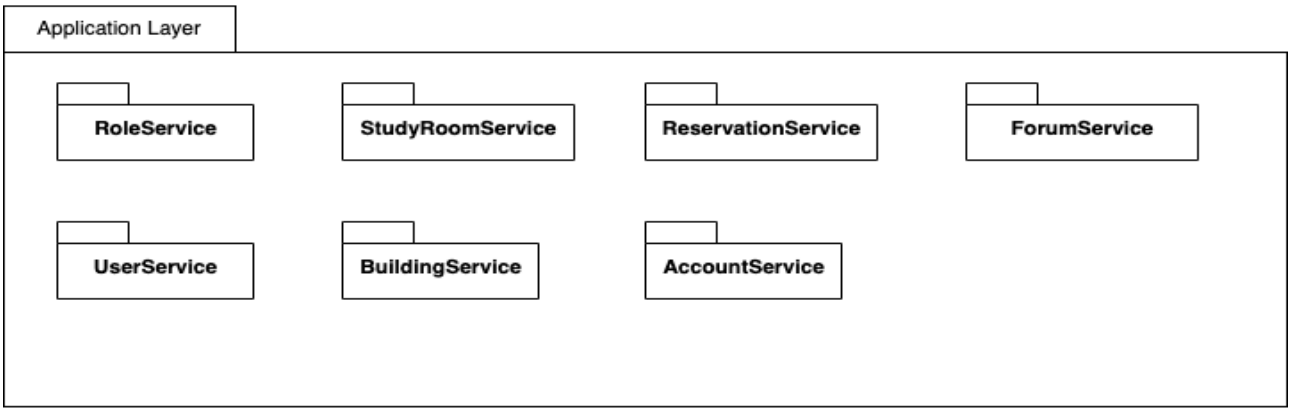
3.1 Panoramica

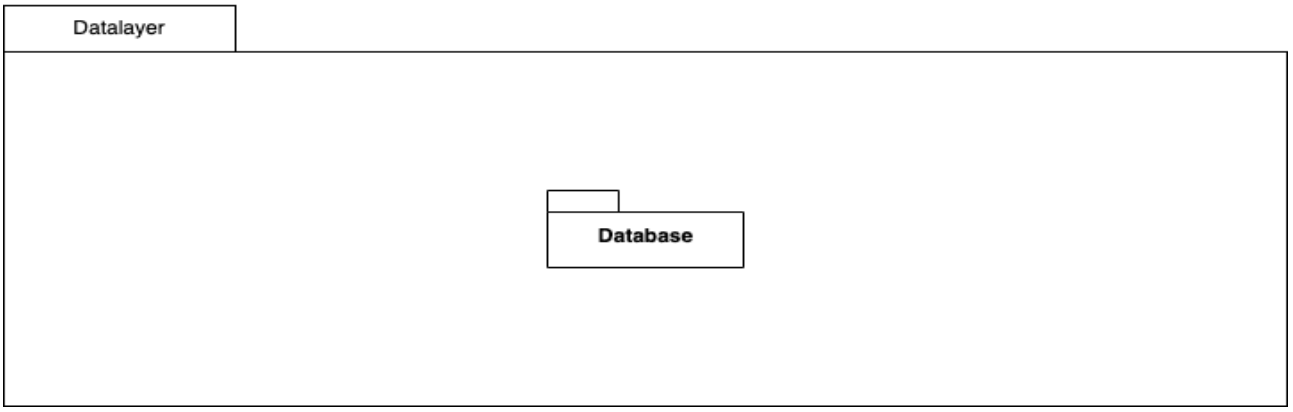
L'architettura three-tier è un'applicazione software ben consolidata che organizza applicazioni in tre tier di calcolo logici e fisici: il tier di presentazione (interfaccia utente), il tier dell'applicazione dove vengono elaborati i dati, e il tier dei dati dove vengono archiviati e gestiti i dati associati all'applicazione.

Il vantaggio principale dell'architettura three-tier è che, poiché ciascun tier viene eseguito sulla propria infrastruttura, può essere sviluppato contemporaneamente da un team di sviluppo separato e può essere aggiornato o scalato in base alle necessità senza effetti sugli altri tier.

2.2 Decomposizione del Sistema

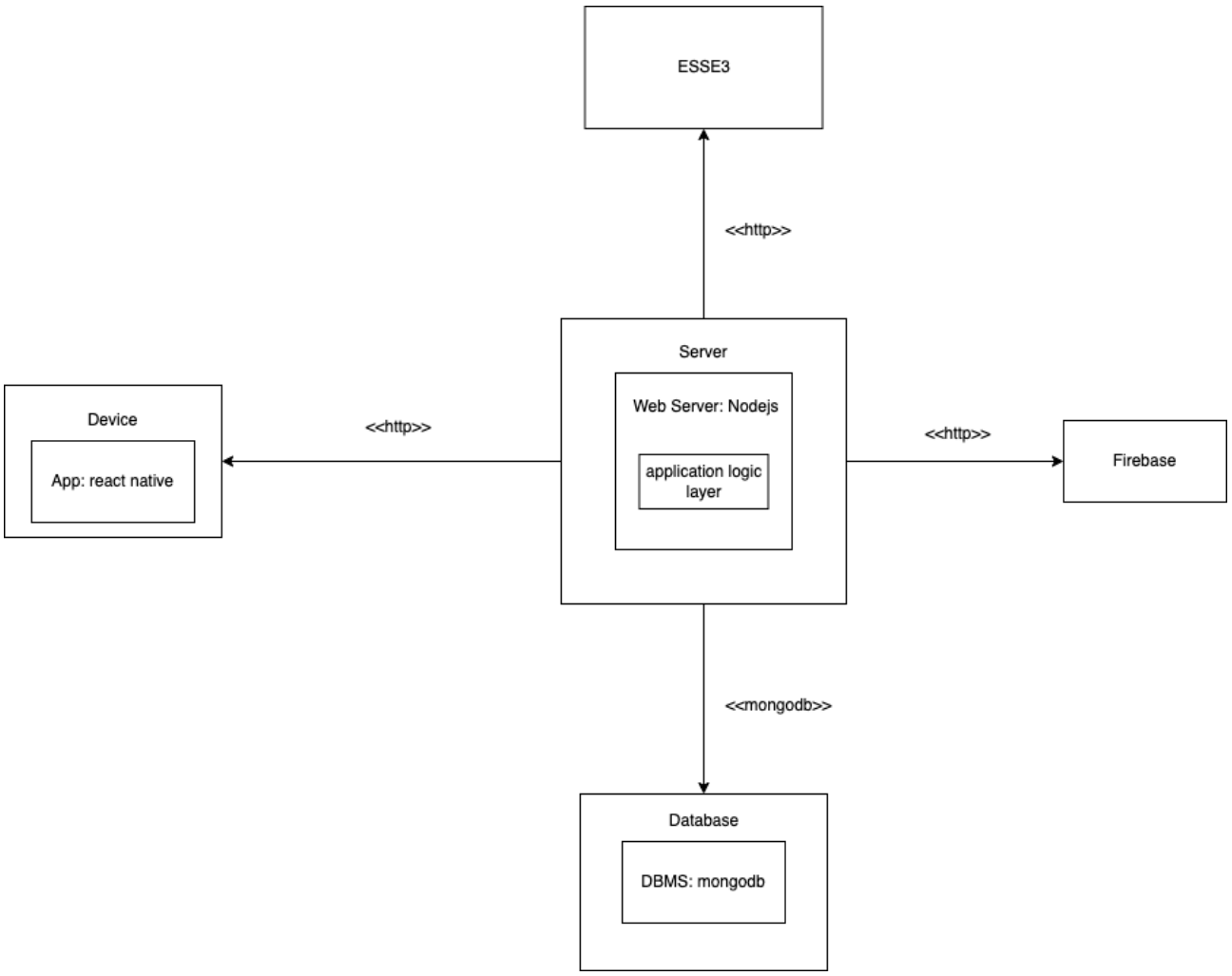






2.3 Mapping

Il sistema utilizza un'architettura three tier. L'interfaccia utente è rappresentata da un applicazione installata sul dispositivo dell'utente scritta in Typescript utilizzando il framework react native. La logica funzionale viene sviluppata su un server scritto nello stesso linguaggio utilizzato per lo sviluppo dell'interfaccia utente così da facilitare la fase di implementazione. In questo caso il framework utilizzato sarà express, eseguito su un webserver in nodejs. La gestione dell'autenticazione viene delegata a due servizi esterni ovvero: Esse3 per l'autenticazione degli studenti e Firebase per l'autenticazione dei gestori delle aule studio. Infine la gestione dei dati persistenti viene effettuata da un database a documenti (mongodb)



3.4 Dati Persistenti

La documentazione in merito è stata inserita nel documento "Persistent Data Management".

3.5 Controllo degli Accessi e Sicurezza

Objects/ Actors	**Account**	**Aula studio**	**Prenotazione**	**Edificio**
Studente	Login Logout	viewStudyRoom addToFavourite reportProblem	addReservation removeReservation viewReservation	
Gestore	Login Logout	viewStudyRoom addStudyRoom editStudyRoom removeStudyRoom makeInactiveStudyRoom	viewReservation	viewBuilding

3.6 Controllo Software Globale

Viene effettuato dal Web Server, che si occupa di smistare le varie richieste alle “Routes” (ci si riferisce a come un End Point di un'applicazione server risponde ad una richiesta del client), le quali ritornano dati formato JSON i quali mediante lo standard RESTful API vengono ricevuti dall’applicativo il quale li fornisce all’utente mediante interfaccia grafica.

4. Servizi dei Sottoinsiemi

Gestione account	
Login studente	Consente ad uno studente di effettuare l'accesso al sistema.
Login gestore	Consente ad un gestore di effettuare l'accesso alla sezione gestionale del sistema.
Logout	Consente ad un utente che ha effettuato l'accesso di uscire dal sistema.
Gestione token di accesso studente	Gestisce la sessione di un utente
Gestione token di accesso gestore	Gestisce la sessione di un gestore
Visualizza profilo	Consente allo studente di visualizzare le proprie informazioni personali.

Gestione aula studio	
Aggiunta aula studio	Consente al gestore dell'aula studio di aggiungerne una in un determinato edificio.
Visualizza aula studio	Consente ad un utente di visualizzare le informazioni di essa.
Rimozione/sospensione aula studio	Consente al gestore di eliminare o sospendere un'aula studio.
Modifica aula studio	Consente al gestore di modificare i dettagli di essa.
Salvataggio aula nei preferiti	Consente allo studente di visualizzare le aule studio preferite in cima alla lista.
Segnalazione problemi	Consente allo studente di segnalare eventuali problemi riguardante l'aula studio

Gestione prenotazioni	
Aggiunta prenotazione	Consente allo studente di effettuare una prenotazione per una determinata aula studio in un giorno e una fascia oraria prestabilita.
Eliminazione prenotazione	Consente all'utente di eliminare una prenotazione attiva.
Visualizza prenotazioni	Consente allo studente di visualizzare lo storico delle proprie prenotazioni; consente al gestore di visualizzare lo storico delle prenotazioni in base all'utente o all'aula studio.

Gestione edifici	
Visualizzazione mappa edifici	Permette allo studente di visualizzare la mappa con tutti gli edifici di Unisa.

5. Glossario

Login: procedura con la quale, grazie ai dati di autenticazione, si effettua l'accesso al sistema.

Logout: procedura di disconnessione dal sistema.

Form: spazio composto da campi predefiniti che permette all'utente di inserire informazioni da inviare ad un server.

Fault: qualsiasi errore o malfunzionamento critico che, nel caso non fosse adeguatamente gestito, porterebbe all'arresto del sistema.

RAD: documento di analisi dei requisiti.

Throughput: potenza effettiva di un canale di comunicazione.

Three-tier: particolare architettura software e hardware per l'esecuzione di un'applicazione.

Client: software che usufruisce dei servizi di un server.

Server: software che offre un servizio specifico richiesto da un client.

Typescript: linguaggio di programmazione, superset di JavaScript, che aggiunge tipi, classi, interfacce.

JavaScript: linguaggio di programmazione utilizzato per realizzare pagine web interattive.

Front-end: parte visibile all'utente con cui egli può interagire.

Back-end: parte che permette il funzionamento delle interazioni che ha l'utente col sistema.

Firebase: piattaforma che permette di salvare e sincronizzare dati elaborati da applicazioni.

Cifratura: forma di crittografia che consiste nel criptare i dati per renderli incomprensibili a prima vista.

Script: sequenza di istruzioni che viene interpretata o portata a termine da un altro programma.

Web server: applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client.

Endpoint: endpoint server rappresenta una posizione specifica in un server

JSON: (JavaScript Object Notation), formato di serializzazione basato su testo per lo scambio di dati tra server e applicazione.

RESTful API: interfaccia di programmazione delle applicazioni conforme ai vincoli dello stile architetturale REST, che consente l'interazione con servizi web RESTful.

REST: (REpresentational State Transfer), trasferimento della rappresentazione dello stato. L'architettura REST è compatibile con qualsiasi protocollo o formato di dati, ma prevalentemente utilizza il protocollo HTTP e trasferisce i dati con JSON. Il funzionamento prevede una struttura degli URL ben definita che identifica univocamente una risorsa o un insieme di risorse e l'utilizzo dei metodi HTTP specifici per il recupero di informazioni (GET), per la modifica (POST, PUT, PATCH, DELETE) e per altri scopi (OPTIONS, ecc.).

Token di accesso: contiene la sessione di accesso e identifica l'utente, i suoi privilegi e il gruppo di utenti al quale appartiene.