“Software Engineering”

Course

a.a. 2019-2020

Lecturer: Prof. Henry Muccini (henry.muccini@univaq.it)

Progetto 6
Mettimi in contatto con...

Date	<13/02/2020>
Deliverable	Deliverable #2
Team (Name)	Flop Team

Team Members		
Nome e cognome	**N# di matricola**	**E-mail**
Alessandro D’Orazio	251811	alessandro.dorazio2@student.univaq.it
Lorenzo Cilli	253121	lorenzo.cilli@student.univaq.it

Table of Contents of this deliverable

1. Specifiche del progetto
2. Challengy/Risks Requirements or Tasks
3. Stato dell'arte
4. Raffinamento dei requisiti
5. Architettura software
6. Dati e loro modellazione
7. Design decisions
8. Design di basso livello
9. Come i NFR e FR sono soddisfatti dal design
10. Effort Recording
11. Appendyx Prototype

1. Specifiche del progetto

UniChat è un servizio di messaggistica disponibile per contesti universitari. Permette infatti di effettuare lo scambio di messaggi tra studenti universitari, docenti e personale amministrativo. Ogni università ha un database differente, con lo stesso schema.

Le conversazioni sono raggruppate in stanze. Le stanze possono essere di tre tipi:

1. Chat diretta, in cui la stanza contiene solamente due partecipanti, ed entrambi i partecipanti possono inviare messaggi. La comunicazione è multidirezionale.
2. Gruppo, in cui la stanza contiene due o più partecipanti e tutti i partecipanti possono inviare messaggi. La comunicazione è multidirezionale.
3. Feed, in cui la stanza contiene due o più partecipanti, suddivisi in amministratori e lettori. Gli amministratori possono inviare e leggere messaggi, mentre i lettori possono esclusivamente leggerli. La comunicazione è monodirezionale.

2. Challenging/Risky Requirements or Tasks

Challenging Task	**Date the task is identified**	**Date the challenge is resolved**	**Explanation on how the challenge has been managed**
Gestione di chat diretta, gruppi e feed	09/12/2019	11/12/2019	Chat diretta, gruppi e feed vengono viste come stanze dal sistema. In questo modo, queste tre entità diventano una tipologia di stanza
Verifica di un utente	14/01/2020	14/01/2020	Un utente del personale amministrativo riceve i documenti e lo verifica
Gestione spam	14/01/2020	14/01/2020	Sistema automatizzato + segnalazioni
Gestione contatti	5/02/2020	5/02/2020	Inserimento social all'interno dell account
Sicurezza	6/02/2020	6/02/2020	Gestita tramite JSON Web Token

3. Stato dell'Arte

Abbiamo chiesto, in un gruppo Facebook dedicato a studenti universitari, di rispondere ad un questionario contenente 5 domande selezionate da noi. Sono arrivate risposte da 21 utenti.

Prima domanda: "Reputi importante avere una versione desktop del software?"
Il 76,2% degli intervistati reputa la versione desktop importante.

Seconda domanda: "È importante per te avere una sezione con i messaggi importanti contrassegnati da te?"
L'85,7% degli intervistati è interessato ad avere una sezione dedicata ai messaggi importanti.

Terza domanda: "Secondo te, quest'app dovrebbe essere in grado di metterti in comunicazione con la segreteria?"
Il 100% degli intervistati desidera questa funzione.

Abbiamo anche chiesto "**Quale sarebbe per te una funzione fondamentale?**"
Le funzioni più interessanti proposte da loro sono:

- Possibilità di inviare allegati
- Comunicare con gli insegnanti
- Risposte immediate da parte dei professori/segreterie

3.1 Schoology

Schoology è un social network e un ambiente di apprendimento virtuale.
Il software permette la condivisione di documenti in ambito universitario. Infatti, accendendo tramite la propria università ed iscrivendosi ai corsi di interesse, è possibile comunicare con il docente tramite un sistema di messaggistica, ed usufruire del materiale da lui caricato.
La condivisione dei documenti non è vincolata solamente ai docenti, poiché anche uno studente può caricare i propri file, creare le proprie cartelle e farle vedere a qualsiasi utente che sfoglia il profilo.
Il sistema, tuttavia, non permette la comunicazione tra studenti.
La sezione nella homepage delle attività recenti presenta un ottimo spunto per la gestione dei feed; infatti, si presenta come una pagina nella quale solo l'admin del corso può effettuare comunicazioni che verranno inviate ad ogni utente iscritto sotto forma di messaggio e di notifica.

3.2 Edmodo

Edmodo è un software per la comunicazione, collaborazione e coaching in ambito scolastico.

Esistono tre tipi di account: insegnante, studente e genitore.
Per accedere come studente o genitore bisogna avere il codice di accesso del corso da seguire o seguito dal proprio figlio.
L'insegnante può condividere materiale, contattare uno o più studenti ed assegnare compiti o quiz.
L'utente genitore invece ha una funzione più da spettatore; può vedere l'andamento del proprio figlio, i suoi voti e comunicare con il docente per qualsiasi eventualità.

3.3 Telegram

Telegram è un servizio di messaggistica basato su cloud.
I messaggi inviati sono salvati sul cloud di Telegram, così da garantire la sincronizzazione istantanea. In questo modo il risultato Telegram consente all'utente di poter accedere ai messaggi da diversi dispositivi contemporaneamente.
Una delle caratteristiche principali di Telegram che lo distingue dal resto dei suoi concorrenti sono le chat segrete, i canali e i bot.
Le chat segrete utilizzano una cifratura end-to-end, ossia la conversazione è cifrata tra i due dispositivi. Le chat segrete dispongono di un'ulteriore funzione: l'autodistruzione; questa funzione permette di impostare un timer nella chat e di eliminare tutti i contenuti alla fine del countdown aumentando la sicurezza per la privacy.
I canali, chat in cui chiunque sia amministratore può inviare messaggi ai membri del canale, anche se questi ultimi non possono rispondere né commentare.
I Bot sono degli account Telegram, gestiti da un software, che offrono molteplici funzionalità con risposte immediate e completamente automatizzate.

3.4 Slack

Slack è una piattaforma per la collaborazione tra team. Nonostante essa non rappresenti un servizio molto simile a quello richiesto dal progetto, offre vari spunti interessanti da poter implementare nel nostro prodotto.
Slack integra un sistema di workspace, cioè spazi di lavoro differenti per ogni ambiente in cui si lavora. Ogni workspace viene organizzato in canali multidirezionali, e la conversazione, oltre ad avere un sistema di messaggistica tradizionale, integra un sistema basato sui thread, come nei forum, in modo tale da poter discutere riguardo un determinato topic evitando la perdita di informazioni.
Il sistema di ricerca dei messaggi è molto interessante dal punto di vista dell'usabilità, poiché tramite un'unica barra di ricerca, è possibile ritrovare messaggi, file, canali ed utenti.
È presente anche una funzione per contrassegnare dei messaggi come importanti, visibili cliccando sull'apposita icona (stella).

La sicurezza è garantita dall'autenticazione a due fattori e da un meccanismo di data encryption.

Vengono stimati circa 10 milioni di utenti. Effettuando ricerche in merito alle tecnologie utilizzate, emerge che il sistema di messaging è scritto in Java, mentre il front end del sito Web è scritto con React.

3.5 Whatsapp

WhatsApp è un'applicazione che permette lo scambio di messaggi di testo, immagini, video e file audio.
Nasce inizialmente come applicazione mobile, successivamente approva su desktop tramite Whatsapp Web.

Uno tra i maggiori problema noti di Whatsapp riguardo la privacy è che richiede l'accesso alla rubrica.
WhatsApp utilizza la crittografia end-to-end, garantisce che solo il mittente e il destinatario possano leggere ciò che viene inviato.
Ciascuna delle chat ha un proprio codice di sicurezza, per verificare che le tue chiamate e i messaggi inviati in una determinata chat siano crittografati end-to-end.
I messaggi sono protetti con dei lucchetti, e solo i due interlocutori avranno le chiavi necessarie per poterli aprire e leggere i messaggi. Per una maggiore protezione, ciascun messaggio inviato ha un proprio lucchetto e una propria chiave.

4. Raffinamento dei Requisiti

4.1 Servizi (con prioritizzazione)

Importanza alta = 3 / Importanza media = 2 / Importanza bassa = 1
Complessità alta = 3 / Complessità media = 2 / Complessità bassa = 1

1. **Stanza**
   1.1. Visualizzazione delle stanze. I=3 / C=2
   L'utente può visualizzare le stanze in cui è partecipante o amministratore, in modo tale da poter inviare e visualizzare i messaggi relativi alla stanza, e nel caso sia amministratore, gestire la stanza.

   1.2. Accesso ad una stanza. I=2 / C=2
   L'accesso ad una stanza in un gruppo o feed può avvenire in tre modalità:

      1.2.1. Ricerca della stanza.
      L'utente cerca la stanza tramite l'apposito pannello, premendo il pulsante "entra"
      1.2.2. Link di invito.
      Un utente può condividere agli altri utenti un gruppo o un feed utilizzando l'apposito strumento di condivisione. Permette di condividere un link ad altri utenti, che cliccando sul link potranno entrare nel gruppo/feed
      1.2.3. Da parte di un amministratore.
      Un amministratore di un feed o di un gruppo può inserirti all'interno di esso

      L'accesso ad una chat diretta non è possibile a seguito della creazione della chat. Infatti, durante la creazione della chat diretta, viene prima selezionato l'altro utente che farà parte della chat (oltre all'utente che la crea), e successivamente nessun utente potrà entrare nella stanza.

   1.3. Comunicazioni importanti. I=2 / C=2
   L'utente amministratore di un gruppo o feed può mettere in evidenza dei messaggi. Evidenziando un messaggio, esso sarà sempre visibile per un periodo di tempo stabilito dall'amministratore, e sarà inviata una notifica ai partecipanti.
   Nelle chat dirette entrambi i partecipanti possono evidenziare i messaggi, e gli effetti sono i medesimi dei messaggi evidenziati in

gruppi e feed.

1.4. Informazioni della stanza. I=2 / C=1
Sarà possibile visionare i partecipanti della stanza e, se presenti, nome e amministratori di essa.

1.5. Archiviazione di una stanza. I=2/C=1
L'utente può archiviare una stanza di cui fa parte tramite l'apposito pulsante "archivia stanza". Per visualizzare l'elenco delle stanze archiviate, l'utente può aprire l'elenco cliccando sul pulsante presente nella home. La chat sarà archiviata solo per l'utente che archivia la chat.

1.6. **Gestione dello SPAM**. I=2 / C=2

1.6.1. Sistema antiflood. I=2 / C=3
L'utente non può inviare un messaggio con lo stesso contenuto per tre volte di fila nell'arco di un minuto. In questo modo, la conversazione risulterà più pulita e funzionale.

1.6.2. Segnalazioni. I=2 / C=2
L'utente di una stanza può segnalare un messaggio.
La segnalazione arriverà all'amministratore tramite notifica, che potrà decidere se silenziare o meno il mittente del messaggio.

2. Chat diretta

2.1. Creazione di una chat diretta. I=3 / C=2
L'utente che vuole comunicare tramite chat diretta con un altro utente può creare una stanza apposita, nel caso in cui non esista già. Se la stanza esiste già, la comunicazione tra i due utenti mediante chat diretta viene effettuata nella stanza già esistente.

3. Gruppo

3.1. Creazione di un gruppo. I=3 / C=2
All'atto di creazione di un gruppo, viene creata una stanza dove l'utente creatore viene nominato amministratore. Deve essere assegnato un nome al gruppo.

3.2. Gli amministratori di un gruppo possono modificare il nome del gruppo, aggiungere/rimuovere partecipanti, nominare altri amministratori ed evidenziare i messaggi importanti.

3.3. Gestione del ban I=2 / C=3

3.3.1. Gli amministratori di una stanza possono silenziare un utente all'interno di essa per un periodo di tempo scelto

dall'amministratore. Prima del ban, l'amministratore può consultare uno storico relativo all'utente, che può utilizzare come parametro per stabilire la durata del ban.

3.3.2. Il ban può essere rimosso da un amministratore manualmente, oppure viene rimosso automaticamente alla scadenza del periodo di tempo

3.4. Ban automatico. I=1 / C=3
È presente un dizionario di parole, stabilito dal personale amministrativo, in cui sono presenti tutte le parole vietate.

3.4.1. Nel caso in cui venga inviato un messaggio con una parola vietata, il messaggio non viene inviato e viene inviata una notifica al mittente, contenente "il messaggio è stato eliminato per parole vietate". La terza volta che uno stesso utente utilizza una parola vietata, esso viene silenziato per un giorno.

3.4.2. Gli sviluppatori del software forniscono una lista di parole vietate, per agevolare i compiti del personale amministrativo, che si occupa di inserire e rimuovere le parole nel dizionario.

4. Feed

4.1. Creazione di un feed. I=3 / C=2
All'atto di creazione di un feed, viene creata una stanza dove l'utente creatore viene nominato amministratore. Deve inoltre essere assegnato un nome al feed.

4.2. Gli amministratori di un feed possono modificare il nome del feed, aggiungere/rimuovere partecipanti, nominare altri amministratori ed evidenziare i messaggi importanti.

5. Gestione degli utenti

5.1. Visualizzazione degli utenti presenti nella piattaforma. I=3 / C=1
UniChat prevede un sistema per la visualizzazione degli utenti registrati alla stessa università dell'utente autenticato. Qui è possibile vedere le informazioni di contatto degli utenti

5.2. Ban dalla piattaforma. I=2 / C=2
Qualora l'utente effettui molteplici volte delle azioni non consentite, come lo spam di messaggi e l'utilizzo di vocaboli inadeguati, è dovere del personale amministrativo di bannare l'utente dalla piattaforma per un periodo di tempo prefissato. In questo modo, l'utente non potrà temporaneamente inviare messaggi. Nel caso si tratti di un

errore, l'utente può inviare una mail per effettuare un "ricorso", in modo tale da rimuovere il ban prima che il tempo scada.

6. **Filtraggio degli utenti**. I=3 / C=2
Gli utenti possono essere filtrati per tipologia, facoltà e dipartimento. Tramite questi filtri è possibile inviare messaggi, creare gruppi e feed.

7. **Invio e ricezione di messaggi**. I=3 / C=2
Questo servizio consiste nello scambio di messaggi all'interno di una stanza. I messaggi devono essere scambiati real-time.

   7.1. Invio di messaggi a più stanze. È possibile inviare lo stesso messaggio a più stanze tramite un sistema di filtraggio che mostra tutte le stanze dell'utente. È presente una funzione "Invio multiplo" da cui sarà inserito il contenuto del messaggio, e successivamente sarà possibile selezionare le stanze a cui mandarlo.

   7.2. Thread. I=1 / C=2
   È possibile commentare un messaggio, per facilitare l'esperienza d'uso nel caso ci siano molti messaggi.

   7.3. Allegati. I=2 / C=3
   È possibile inviare un allegato tramite l'apposito pulsante vicino al campo di testo del relativo messaggio.

8. **Registrazione**. I=3 / C=1
Gli utenti si possono registrare fornendo nome, cognome, email, password, tipo, università, anno di immatricolazione e facoltà. La registrazione viene effettuata autonomamente dall'utente.

   8.1. Tipologie di utenti: studente, docente o personale dell'amministrazione. Esiste anche un tipo speciale, lo sviluppatore del software, che può essere creato solamente da un altro sviluppatore del software.

   8.2. Verifica degli utenti. I=3 / C=2
   I docenti e il personale dell'amministrazione devono essere verificati per motivi di sicurezza, altrimenti qualsiasi utente potrebbe registrarsi come docente o personale amministrativo.
   La verifica avviene mediante l'invio di un'email ad un utente del personale amministrativo dalla casella universitaria, contenente una scansione dei propri documenti. Il personale amministrativo dovrà poi confermarlo tramite un apposito pannello.

9. **Autenticazione**. I=3 / C=1
   Gli utenti effettuano il login inserendo email e password
   9.1. Reset della password. I=3 / C=1
   Gli utenti possono autonomamente reimpostare la propria password nel caso in cui sia stata dimenticata.

10. Comunicazioni con servizi esterni.
   10.1. Visualizzazione del numero telefonico. I=3 / C=1
   Un utente può visualizzare il numero telefonico di un altro utente, per effettuare chiamate e inviare SMS.

   10.2. Email. I=3 / C=1
   L'utente può visualizzare l'email di un altro utente per avviare una conversazione. La conversazione avviene al di fuori di UniChat

   10.3. App di messaggistica. I=2 / C=1
   L'utente può impostare i propri contatti delle app di messaggistica, come Skype e Twitter.

11. **Ricerca dei messaggi.** I=2 / C=3
   L'utente può effettuare una ricerca intelligente dei messaggi, che permette di visualizzare i messaggi contenenti una determinata frase (o parola), oppure i messaggi inviati in una determinata data.

12. **Gestione delle informazioni dell'università**. I=2 / C=2
   Il personale amministrativo può inserire e rimuovere dipartimenti e corsi di laurea.
   12.1. Inserimento di un'università. I=2 / C=1
   È compito del team di sviluppo di inserire, modificare ed eliminare le università, tramite un apposito pannello.

4.2 Requisiti non Funzionali (ISO/IEC 25010)

Compatibilità

Interoperability

- Il software non ha bisogno di comunicare con altri software per il suo funzionamento. E' comunque possibile invocare delle API per prelevare delle informazioni presenti su UniChat.

Performance

- Per limiti di spazio, è possibile inviare solamente allegati con dimensione massima di 10 megabyte.

Usabilità

Learnability

- Gli studenti e i docenti saranno in grado di utilizzare al meglio la piattaforma dopo un training di 2 ore, mentre il personale amministrativo dopo un training di 3 ore, date le funzionalità aggiuntive a loro disposizione.

Operability, User interface aesthetics

- Ogni operazione presente in UniChat dovrà essere utilizzabile raggiungendo il miglior compromesso possibile tra minor numero di click e interfaccia pulita.

Accessibility

- Il sistema deve essere facile da usare e deve essere organizzato in modo tale che gli errori commessi dall'utente siano ridotti al minimo.

Reliability

- Dovrà essere presente un backup giornaliero dei dati

Availability

- UniChat deve garantire uno scambio di messaggi quasi sempre disponibile, poiché la perdita di comunicazioni potrebbe causare problematiche di natura logistica e organizzativa.

Fault tolerance

- Il sistema richiede due macchine per garantire l'operatività a seguito di guasti, che vengono risolti, in base al tipo di guasto, da ripresa a caldo o ripresa a freddo

Recoverability

- I messaggi saranno salvati sul server e non localmente su ogni dispositivo

Sicurezza

Confidentiality

- Gli studenti non possono visualizzare il numero di telefono di docenti e personale amministrativo, poiché potrebbero utilizzarlo in modo inappropriato.

Integrity

- La password viene codificata mediante un sistema "Bcrypt", per garantire la sicurezza della password in caso di Data Breach

Authenticity

- Il sito Web di UniChat sarà dotato di un certificato SSL, per garantire la connessione sicura

- Alla richiesta di reset della password, viene inviata un'email all'utente per notificare quest'operazione

- I docenti e il personale amministrativo vengono verificati dagli utenti del personale amministrativo tramite un apposito pannello

- Inizialmente dovrà essere presente un utente master per ogni università, che si occuperà di verificare il personale amministrativo e i docenti. Le credenziali dell'utente master saranno relative ad un membro del personale amministrativo, scelto dall'università.

Manutenibilità

Modularity

- Il sistema di autenticazione può essere modificato senza compromettere le funzionalità del software.

Reusability

- UniChat non ha alcun tipo di dipendenza da nessuna università. In questo modo, UniChat può supportare potenzialmente un numero infinito di università

Portabilità

- UniChat può essere utilizzato su una macchina contenente i software necessari a far funzionare uno stack LAMP e Laravel 6.0

Adaptability

- Il software dovrà essere modulare e permettere l'integrazione con altri sistemi

4.3 Scenari d'uso dettagliati

1. Giorgio è uno studente di biologia. Utilizza ogni giorno UniChat tramite smartphone per scambiare messaggi relativi al progetto a cui sta lavorando, mediante il gruppo che ha creato lui stesso. Non ci ha messo molto a cercare gli altri studenti, poiché ha utilizzato il sistema dei filtri per facoltà. Fissa una scadenza per la consegna del progetto ed evidenzia il messaggio, in questo

modo gli altri riceveranno una notifica per leggere il messaggio.

2. Mattia, studente di chimica, deve sostenere un esame orale di chimica, ma non può presentarsi all'orario deciso dal docente, quindi tramite la chat diretta informa il professore di questa problematica.

3. Carlo, studente di ingegneria informatica, ha ricevuto qualche settimana fa un messaggio in cui era previsto un cambio aula per la lezione di fisica, ma non riesce a ricordarne i dettagli. Utilizza quindi la funzione di ricerca dei messaggi, filtrando tra i messaggi dell'ultimo mese, quelli in cui è presente il testo "cambia aula".

4. Daniele è un docente di matematica, e ogni mattina, dopo aver bevuto il suo caffè, accende il computer ed apre UniChat. È un docente molto attivo, infatti possiede un feed dedicato all'insegnamento di "Analisi 1", da cui invia le comunicazioni agli studenti. Daniele sa che giovedì prossimo non sarà presente a lezione, quindi tramite i messaggi in evidenza, invia una notifica a tutti gli studenti per informarli. Non deve preoccuparsi di inserire i nuovi studenti nel proprio feed, poiché nella sua pagina personale ha inserito un link per partecipare al feed di "Analisi 1", di cui è docente insieme a due collaboratori, entrambi nominati da Daniele. I tre condividono un gruppo per fissare gli appuntamenti e gli orari delle esercitazioni.

5. Stefano lavora nel personale amministrativo, e si occupa dei bandi pubblici. È appena uscito un bando pubblico per una borsa di studio riservata agli studenti di medicina. Stefano quindi, tramite il sistema di ricerca integrato in UniChat, invia la comunicazione a tutti gli studenti di medicina, senza doverli cercare manualmente.
   Inoltre, ha un nuovo collega di nome Matteo nel personale amministrativo, da cui riceve una mail contenente una scansione dei documenti, per verificarlo nel sistema UniChat. La verifica avviene in meno di 30 secondi!

6. Mario è un ragazzo che si prende gioco degli altri. Prova a registrarsi all'interno di UniChat come docente, per inviare comunicazioni false, ma quando gli viene chiesto di inviare una mail dalla casella universitaria del docente contenente la scansione dei documenti, fallisce nel suo tentativo. Si registra quindi come studente, ed inizia a inviare parole offensive a tutti gli studenti, che però non ricevono nulla, poiché il sistema di SPAM non ha permesso l'invio dei messaggi. Arriva una comunicazione al personale amministrativo, che silenzia Mario dalla piattaforma.

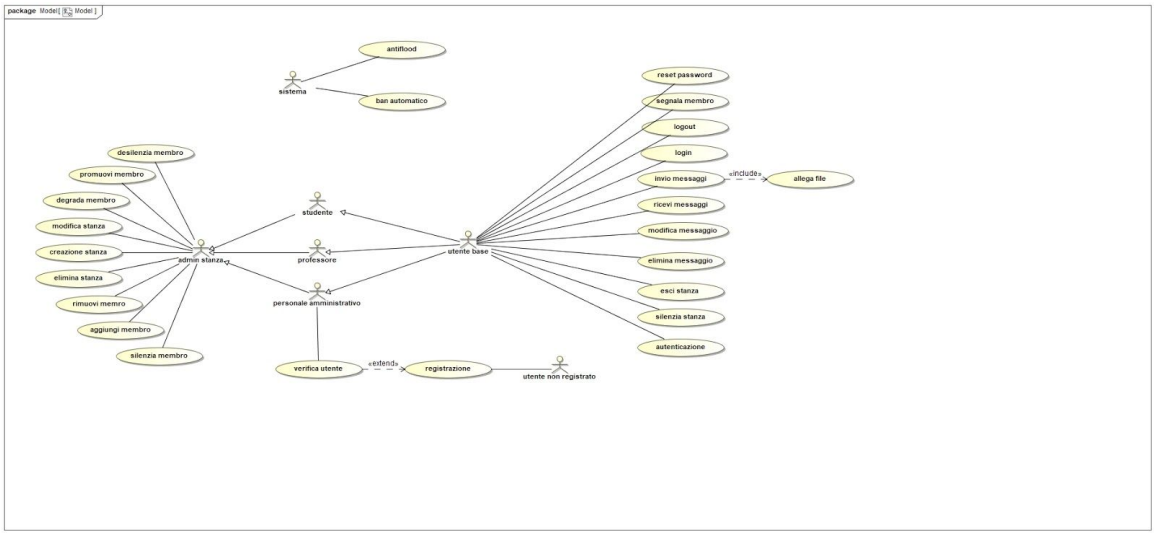
4.4 Excluded Requirements

1. Accesso con SPID o credenziali universitarie. Non viene gestito poiché allungherebbe le tempistiche, e non è detto che gli utenti abbiano delle credenziali di questi due servizi. È possibile gestire questa tipologia di accesso come modalità secondaria, ma non è stato fatto per motivi di tempo.
2. Segreteria. In base alle risposte date nell'intervista, la segreteria sembra una funzionalità interessante. Il progetto è incentrato su un'applicazione di messaggistica, quindi realizzare la segreteria non è compito di UniChat. Un workaround nel caso in cui un'università voglia avere la segreteria in UniChat, è di creare un utente del personale amministrativo "Segreteria" seguito dal nome dell'università.

4.5 Assunzioni

1. I permessi di ogni tipologia di utente sono gli stessi, quindi un utente del personale amministrativo ha gli stessi permessi di tutti gli altri utenti del personale amministrativo.
2. Ogni docente o membro del personale amministrativo ha una mail universitaria
3. Ogni docente creerà un feed per ogni suo insegnamento e condividerà il link dei suoi studenti. In questo modo si riducono le tempistiche di realizzazione del software, poiché non deve essere il software a creare i feed per ogni insegnamento

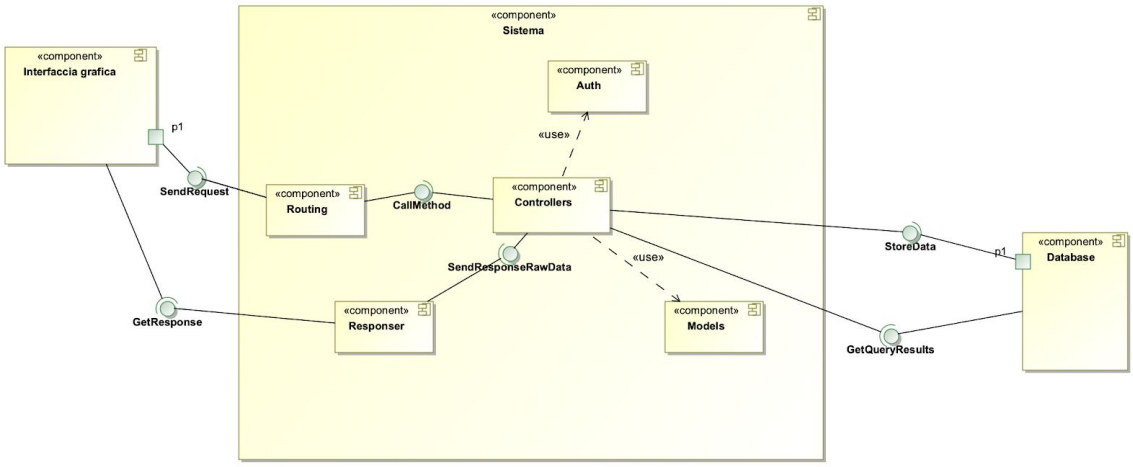
4.6 Use Case Diagrams



Lo use-case si basa principalmente su un unico attore principale che è l'utente base, il quale ha un'evoluzione in base al ruolo che ricopre all'interno del software.
Essendo un utente base, questo ha a disposizione le principali opzioni base per usufruire del servizio, quali login, logout, inviare e ricevere i messaggi ecc.
L'utente base può essere di tre tipi studente, professore e personale amministrativo i quali hanno tutti le stesse operazioni dell'utente base se non fosse che l'utente del personale amministrativo ha anche il compito di autenticare un utente che ancora si registra.
Tutti e tre gli utenti possono essere degli admin di una stanza che possono creare loro o essere promossi da un admin di una stanza di cui fanno parte.
Gli admin possono creare modificare ed eliminare una stanza (gruppo o feed) ed aggiungere, espellere, promuovere, degradare, silenziare e desilenziare un membro.
L'ultimo attore del diagramma è il sistema che si occupa di due operazioni automatiche che sono l'antiflood (l'eliminazione di messaggi uguali se si supera il numero massimo di spam nell'arco di un minuto) e il ban automatico (che elimina i messaggi inviati con all'interno una parola presente nel dizionario delle parole proibite, se si supera il numero limite di messaggi con all'interno parole presenti nel dizionario si viene silenziati per un giorno).

5. Architettura Software

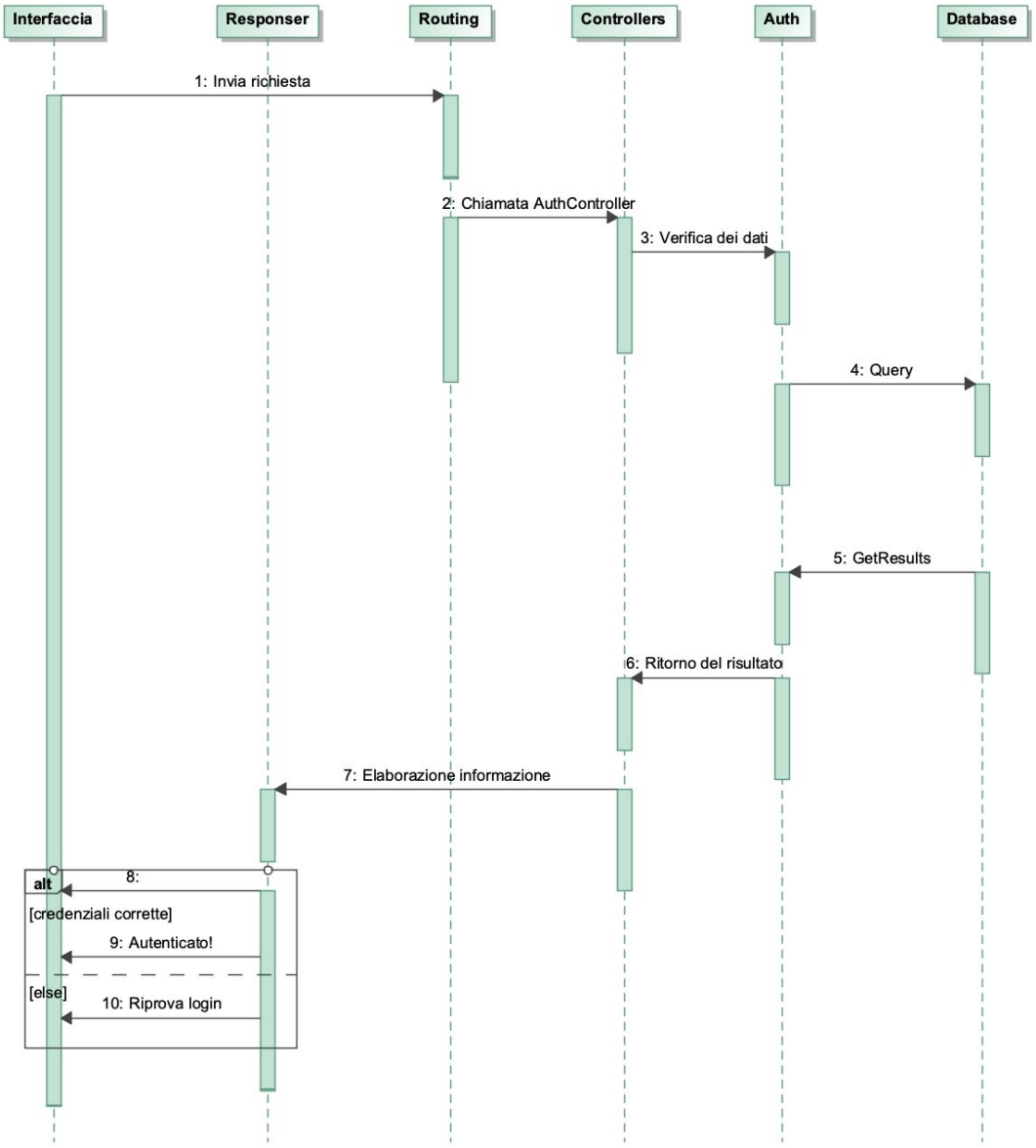
5.1 Vista statica del sistema: Component Diagram



Interfaccia grafica: componente che si occupa di mostrare le viste di UniChat
Sistema: componente principale di che elabora i dati forniti dall'interfaccia grafica mediante i suoi sottocomponenti.
La componente di **Routing** riceve una richiesta dall'interfaccia grafica. Il Routing chiama il metodo richiesto del relativo controller, che utilizza l'**Auth** per verificare i permessi dell'utente. Il Controller utilizza anche i **Models** per ottenere i modelli del sistema. Inoltre, il controller invia le query al **Database**, che restituisce il risultato. Il Responser si occupa di tradurre la risposta del Controller in formato JSON, per passarlo all'interfaccia grafica.

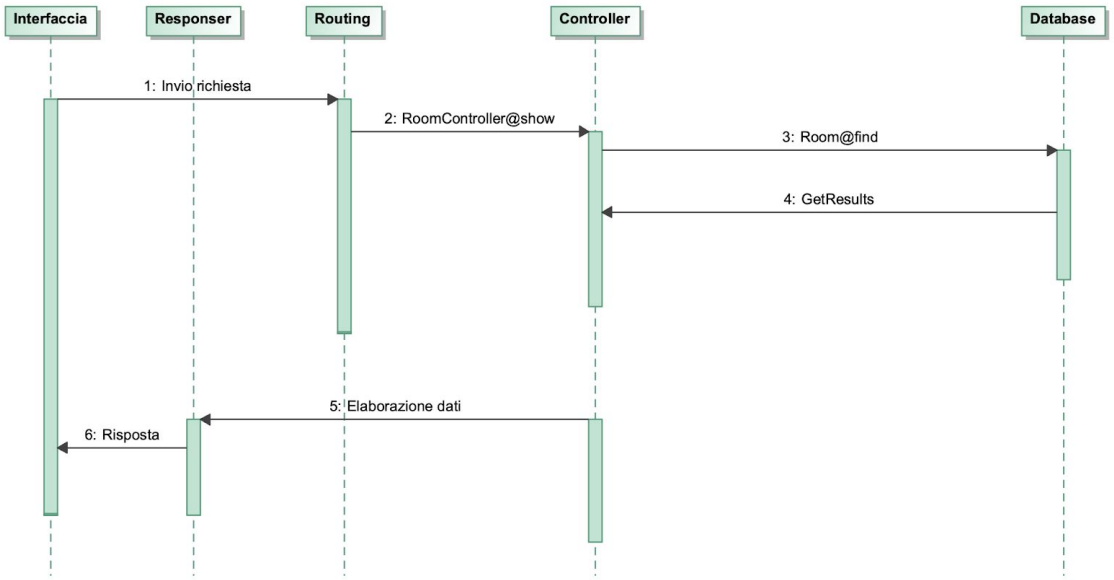
5.2 Vista dinamica del sistema: Sequence Diagram

5.3.1 Autenticazione



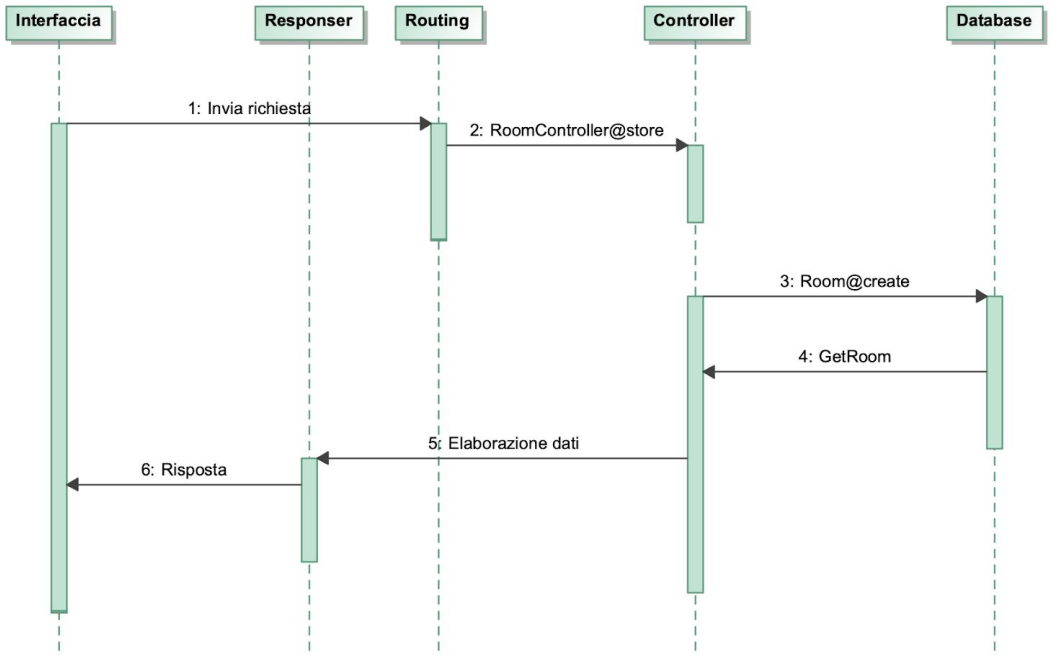
L'interfaccia invia la richiesta tramite le API, che, tramite la componente Routing richiama la funzione del Controller, che verifica i dati tramite Auth. Auth effettua la query e ritorna il risultato al Controller. Il Controller elabora il risultato e invia il risultato dell'elaborazione al Responser, che, nel caso siano credenziali corrette, autentica l'utente, altrimenti lo riporta al login.

5.3.2 Visualizzazione stanza



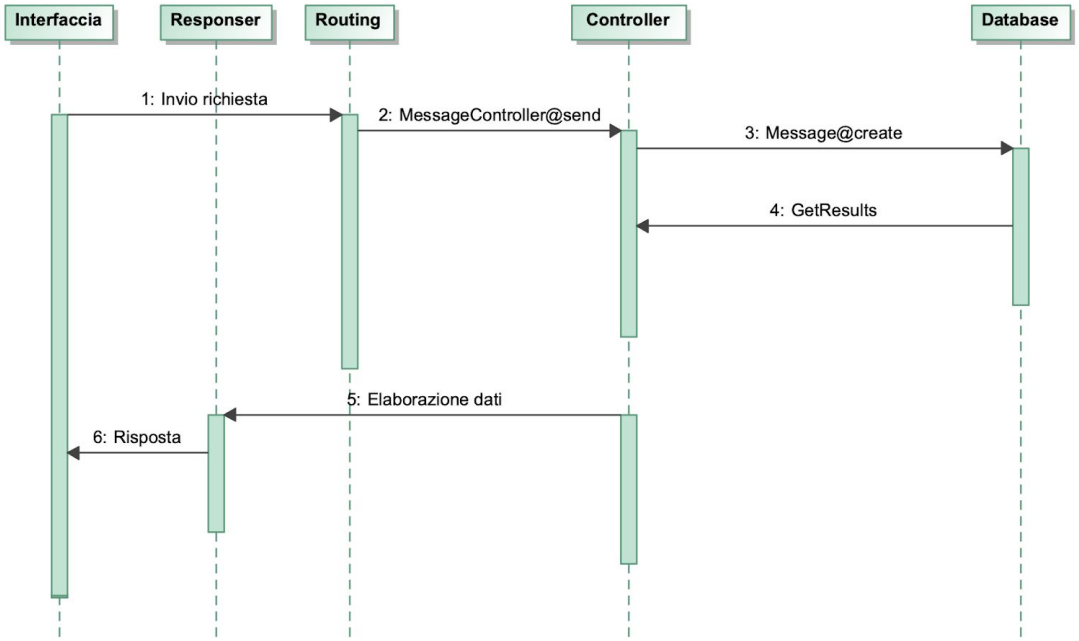
L'interfaccia invia l'id della stanza a Routing, che chiama il metodo show( ) presente nel Controller delle stanze. Il Controller cerca la stanza specifica con una query al database e verifica i permessi dell'utente. Successivamente vengono elaborati i dati dal Responser per essere mostrati tramite l'interfaccia grafica.

5.3.3 Creazione di una stanza



L'interfaccia invia la richiesta, che viene elaborata tramite il Controller delle stanze. Il controller crea la stanza tramite il Database e li elabora per fornirli al Responser, che li passa all'interfaccia.

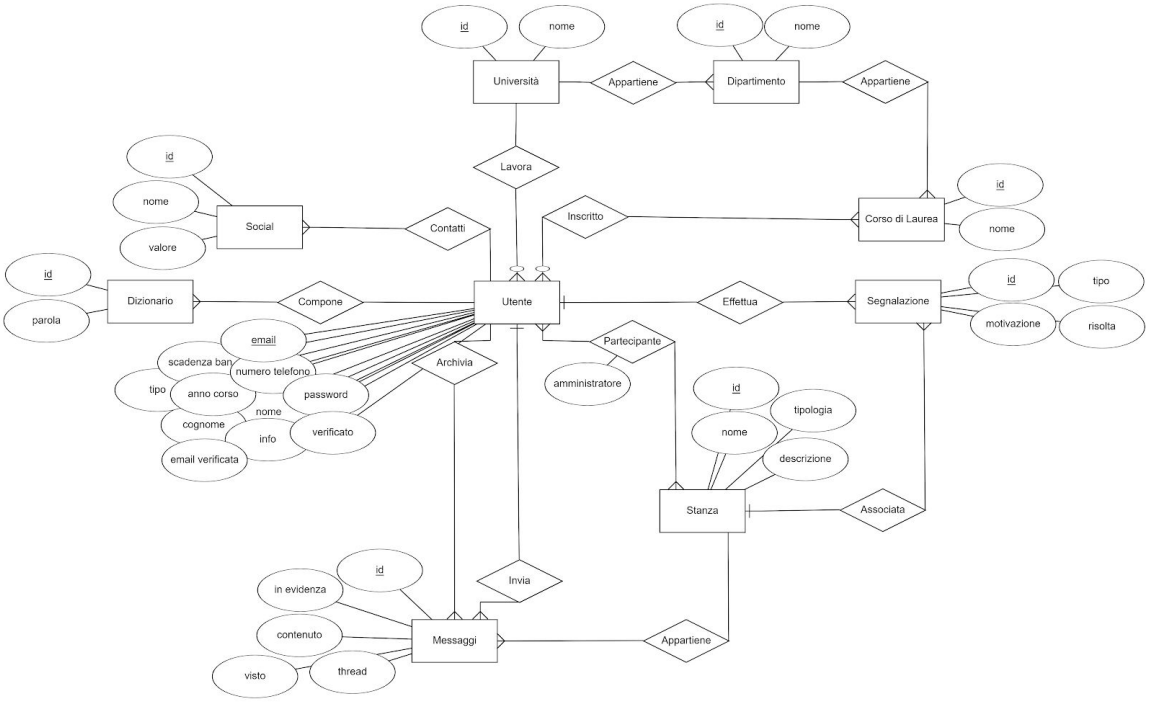
5.3.4 Invio di un messaggio



Viene inviata la richiesta dall'Interfaccia. Il Routing richiama il Controller relativo ai messaggi. Viene effettuata una verifica dei permessi, e successivamente viene chiamata la query. I dati vengono poi modellati dal Controller che li invia al Responser. Il Responser ritorna una risposta in formato JSON all'interfaccia.

6. Dati e loro modellazione

Chiave primaria *Chiave esterna*



Dizionario (ID, parola)

Università (ID, nome)
Dipartimento (ID, nome, *università_id*)
Facoltà (ID, nome, *dipartimento_id*)

Utente (ID, nome, cognome, email, password, tipo, verificato, numero_di_telefono, scadenza_ban, email _verificata, *facoltà_id*, *università_id*)
L'utente, nel caso sia uno studente, è associato al suo corso di laurea, mentre se è un docente o fa parte del personale amministrativo, è associato all'università.
Tipo: studente=1 docente=2 personale amministrativo=3 sviluppatore=10

Social (ID, nome, valore, *utente_id*)
Viene data totale autonomia all'utente dei canali da mostrare all'interno della piattaforma

Stanza (ID, tipo, nome, descrizione, *università_id*)
Tipo: chat diretta = 1 /gruppo = 2/feed = 3

Messaggi (ID, contenuto, importante,thread, *stanza_id*, *utente_id*)
Partecipante(*utente_id*, *stanza_id*,)

Segnalazione (ID, motivazione, risolta,tipo, *utente_id*, *messaggio_id*)

7. Design Decisions

1. Gestione di chat diretta, gruppi e feed tramite un sistema di gestione a stanze. Questo permette di generalizzare i tre concetti in un'unica entità, riducendo le tempistiche

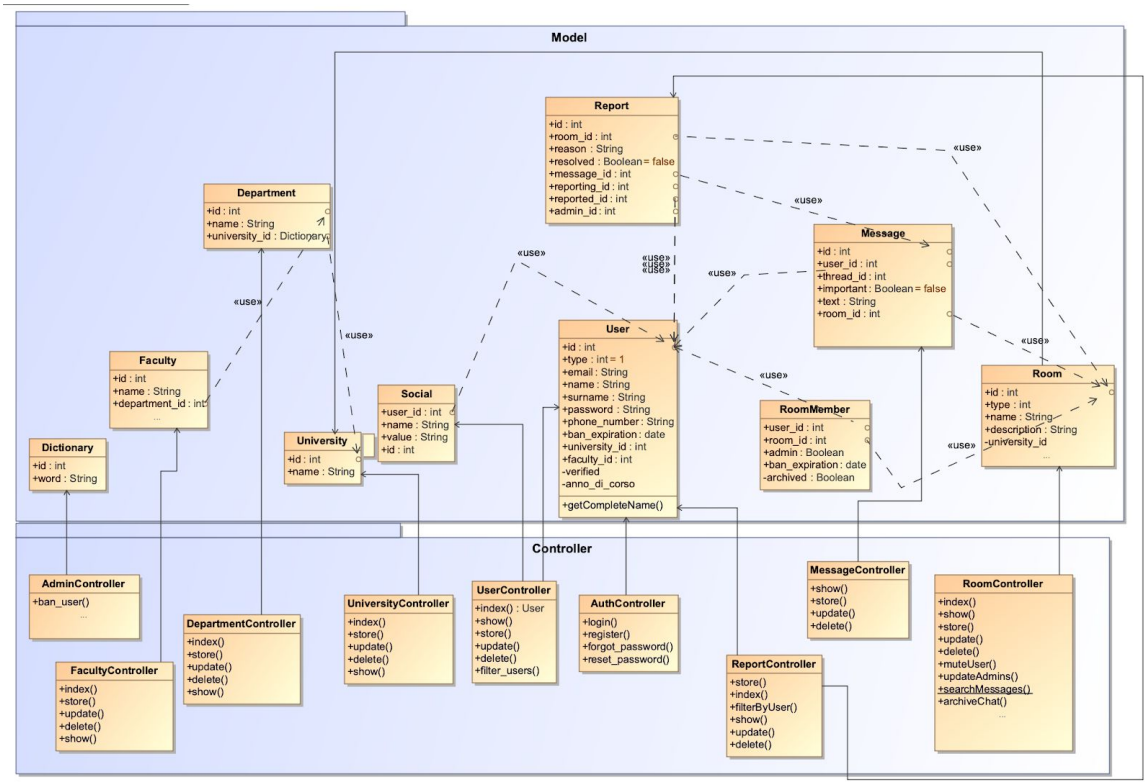
2. Autenticazione nel sistema tramite credenziali salvate nel database di UniChat, anziché l'utilizzo di servizi esterni come SPID o credenziali universitarie. Questo perché la componente dell'autenticazione è facilmente sostituibile, e l'autenticazione tramite credenziali richiede meno tempo rispetto all'integrazione di servizi esterni. Nella prima versione sarà impiegato un sistema di sicurezza tramite JWT.

3. La verifica degli utenti è manuale. In questo modo la verifica degli utenti è responsabilità del personale amministrativo, che si presume conosca la struttura universitaria, eliminando le problematiche legate ad una verifica di tipo automatica, che potrebbe verificare utenti malintenzionati.

4. Il ban automatico si basa sul dizionario, poiché è una soluzione efficace e non richiede molto tempo per la sua implementazione, mentre il ban manuale è un compito del personale amministrativo, il quale, avendo l'elenco di utenti, con eventualmente il numero di ban ottenuti, si occuperà del ban alla piattaforma

5. Per limitare lo spam esiste un sistema automatico, l'antiflood, che blocca l'invio dello stesso messaggio per più di tre volte nell'arco di un minuto. Si suppone che nessun utente possa avere realmente intenzione di scrivere lo stesso messaggio nell'arco di un minuto, quindi un sistema di questo tipo va potenzialmente a ridurre il numero di messaggi inutili.

8. Design di Basso Livello



Il Class Diagram mostra la strutturazione di UniChat, evidenziando un pattern basato su modelli e controller. I modelli si occupano di definire le entità presenti nel sistema, mentre i controller gestiscono le operazioni effettuabili.

Le operazioni principali sono di tipo CRUD: create, read, update, delete, poiché sono le operazioni basilari per il funzionamento del software.

9. Explain how the FRs and the NFRs are satisfied by design

Requisiti funzionali

Gestione delle stanze
La gestione delle stanze avviene tramite un room system, che astrae il concetto di chat diretta, gruppo e feed ad un unico concetto, poiché le operazioni effettuabili sono molto simili. Nonostante questa generalizzazione, viene comunque mantenuta l'informazione riguardante il tipo di stanza, in modo da poter gestire correttamente le operazioni in comune, che però richiedono implementazioni diverse, come la gestione dei partecipanti, che in gruppi e feed può maggiore di 2, mentre nelle chat dirette è il numero massimo di partecipanti

Visualizzazione degli utenti
Riguardo gli utenti vengono salvate tutte le informazioni relative ai contatti tramite i social (esclusa l'email e il numero telefonico), in modo tale che si possa anche condividere il link ad un account di un social network o IM. Viene effettuato un controllo sul tipo dell'utente se si accede alla pagina di un utente docente o del personale amministrativo, poiché, se studente, non può visualizzare il numero telefonico.

Gestione dello SPAM
Il controller relativo all'invio di messaggi si occupa di effettuare le dovute verifiche relative al controllo dei messaggi inviati.

Requisiti non funzionali

Interoperability
UniChat contiene tutte le funzionalità necessarie al funzionamento del sistema. Partendo dal sistema di autenticazione, in cui è presente un sistema di autenticazione username/password, basato su JSON Web Token, fino ad arrivare alla gestione delle stanze, gestite dal relativo controller senza l'utilizzo di servizi esterni, arrivando allo scambio di messaggi,.

Usabilità
UniChat offre un sistema con funzionalità che possono essere utilizzate da qualsiasi utente. Infatti, le schermate dell'interfaccia utente sono molto pulite e mostrano solamente i dati richiesti e le operazioni effettuabili in un determinato contesto.

Reliability
I messaggi vengono salvati sul server, in modo tale che nessun utente possa perdere le proprie chat. Inoltre, il backup giornaliero può essere effettuato creando un apposito cron job che fa un dump del database.

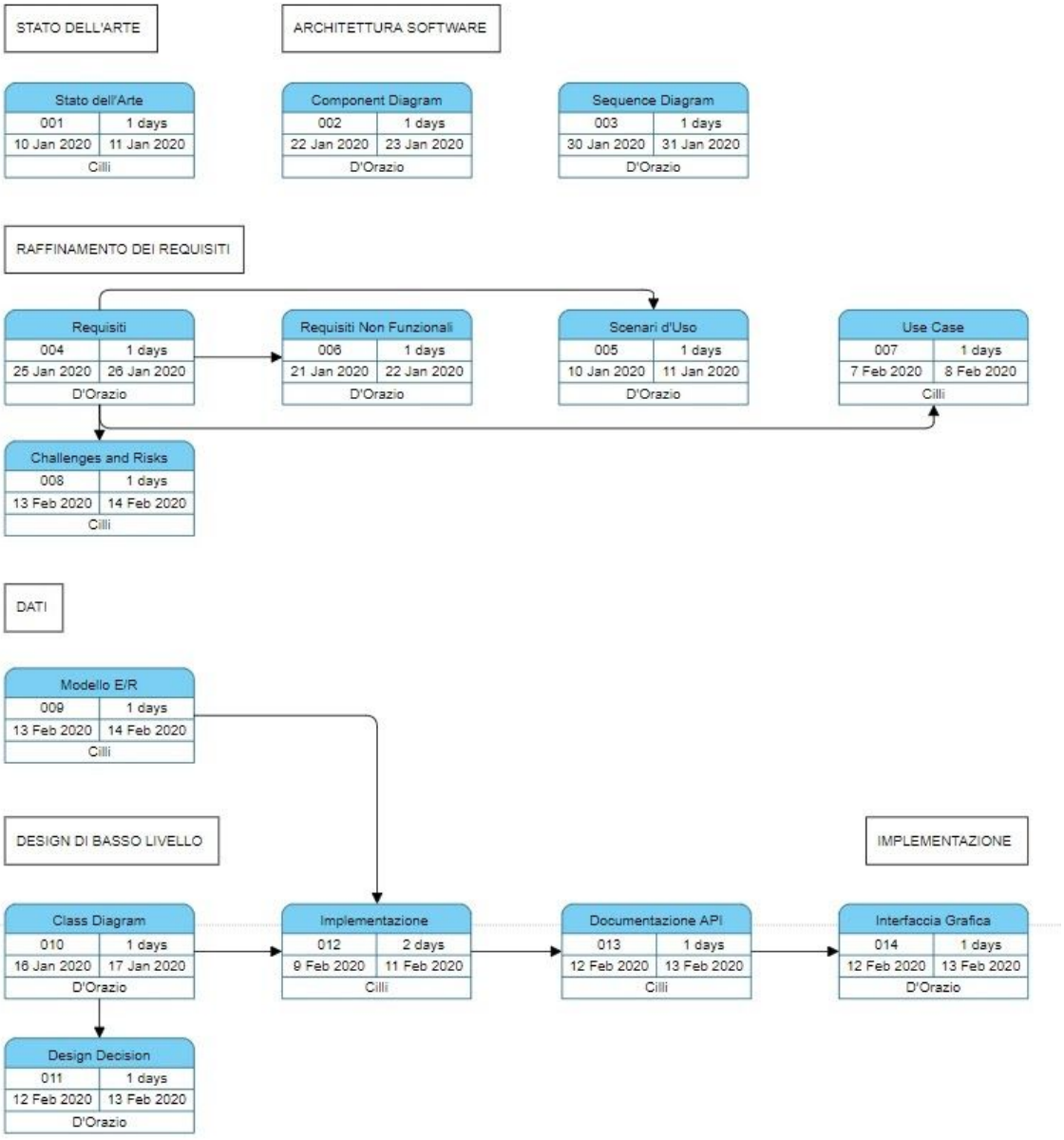
Sicurezza
La sicurezza viene garantita tramite il sistema di JSON Web Token, e l'autenticità tramite la verifica dei profili più importanti (docenti e personale amministrativo)

Manutenibilità
Il software sfrutta un pattern MVC, non dipende da nessuna università e il sistema di autenticazione può essere facilmente sostituito modificando il controller relativo all'autenticazione

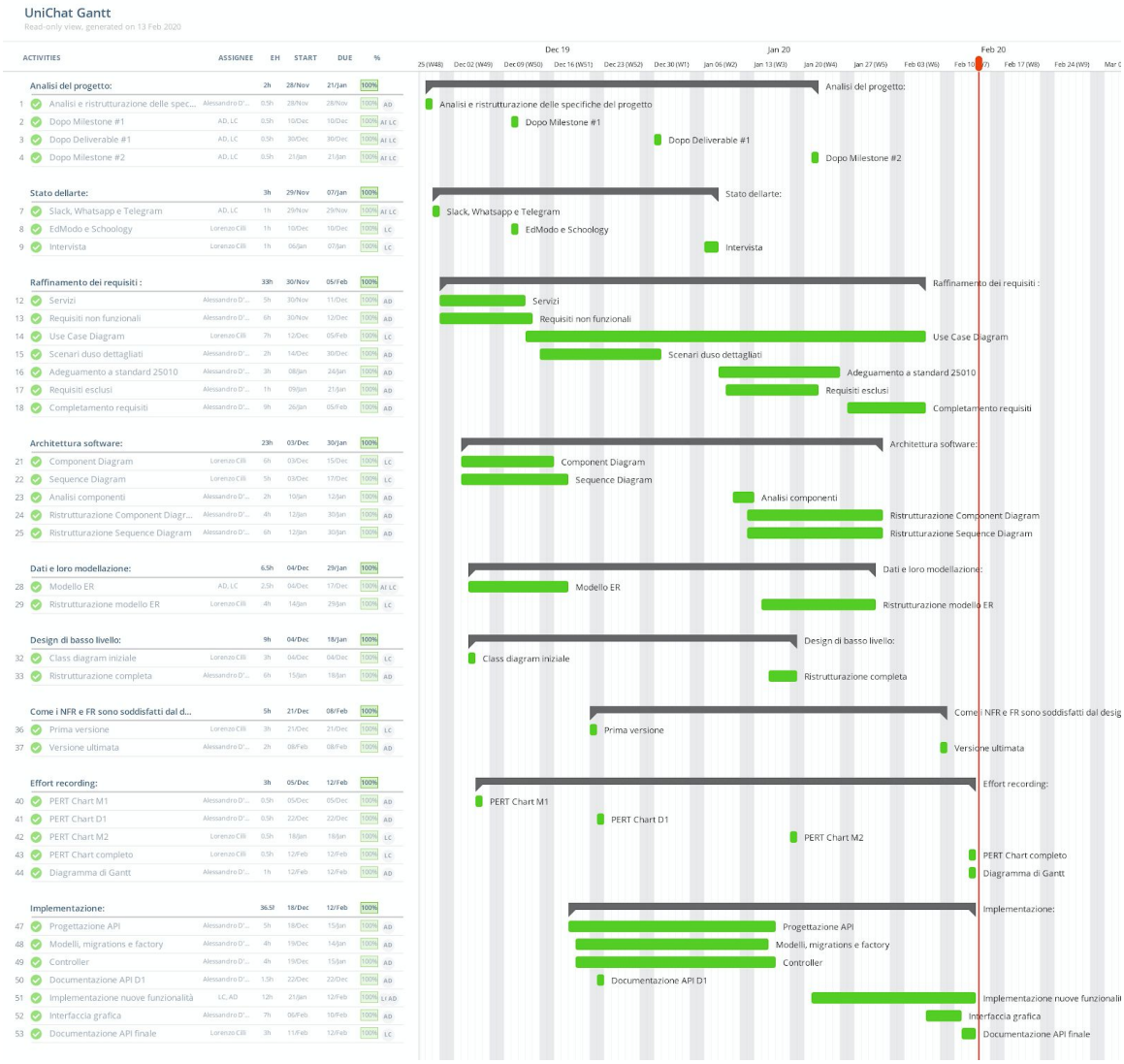
Portabilità
UniChat può essere installato su un'altra macchina contenente uno stack LAMP. E' possibile realizzare una nuova installazione creando una cartella con tutti i file presenti su GIT, installando tramite composer i pacchetti necessari e configurando il database.

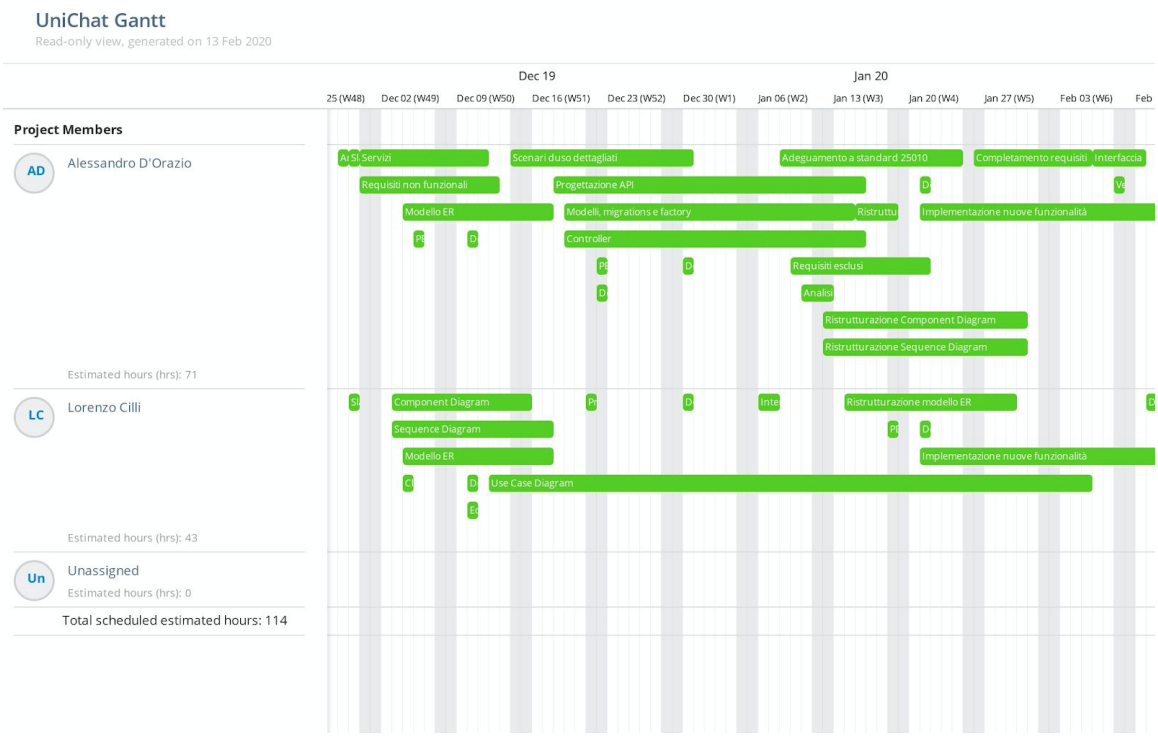
10. Effort Recording

10.1 PERT



10.2 GANTT


UniChat Gantt
Read-only view, generated on 13 Feb 2020
ACTIVITIES
ASSIGNEE
EH
START
DUE
%
Dec 19
Jan 20
Feb 20
Analisi del progetto:
Analisi e ristrutturazione delle specifiche del progetto
Dopo Milestone #1
Dopo Deliverable #1
Dopo Milestone #2
Stato dellarte:
Slack, Whatsapp e Telegram
EdModo e Schoology
Intervista
Raffinamento dei requisiti :
Servizi
Requisiti non funzionali
Use Case Diagram
Scenari duso dettagliati
Adeguamento a standard 25010
Requisiti esclusi
Completamento requisiti
Architettura software:
Component Diagram
Sequence Diagram
Analisi componenti
Ristrutturazione Component Diagram
Ristrutturazione Sequence Diagram
Dati e loro modellazione:
Modello ER
Ristrutturazione modello ER
Design di basso livello:
Class diagram iniziale
Ristrutturazione completa
Come i NFR e FR sono soddisfatti dal desig
Prima versione
Versione ultimata
Effort recording:
PERT Chart M1
PERT Chart D1
PERT Chart M2
PERT Chart completo
Diagramma di Gantt
Implementazione:
Progettazione API
Modelli, migrations e factory
Controller
Documentazione API D1
Implementazione nuove funzionalità
Interfaccia grafica
Documentazione API finale



10.2 Logging

Ore Alessandro: 71
Ore Lorenzo: 41

Ore in comune: 4

Ore Totali: 120

Personal Journal

Team (number and name): Flop team
Student name: Alessandro D'Orazio, Lorenzo Cilli
Student number: 251811, 253121
Email: alessandro.dorazio2@student.univaq.it, lorenzo.cilli@student.univaq.it

When		Time spent (min)	Partners	Brief Description of the performed task	Category	Sub-Category
11	28	90	A	Analisi e ristrutturazione delle specifiche del progetto	Learning	Studio del progetto
11	28	30	AL	Analisi progetto	Doing	Studio del progetto
11	29	50	AL	Stato dell'arte	Doing	Stato dell'arte
11	30	150	A	Raffinamento dei requisiti	Learning	Raffinamento dei requisiti
12	1	30	A	Use Case Diagrams	Doing	Raffinamento dei requisiti
12	2	45	L	Use Case Diagram	Doing	Raffinamento dei requisiti
12	2	180	L	Raffinamento dei requisiti	Learning	Raffinamento dei requisiti
12	3	45	A	Studio Sequence Diagram	Learning	Studio individuale
12	3	30	A	Studio tool MagicDraw	Learning	Studio software
12	3	150	L	Component Diagram	Doing	Architettura software
12	3	45	A	Sequence Diagram	Doing	Architettura software
12	4	90	L	Studio Component Diagram	Learning	Studio individuale
12	4	45	L	Studio Class Diagram	Learning	Studio individuale
12	4	45	L	Studio tool MagicDraw	Learning	Studio software
12	4	150	A	Modello E/R	Doing	Dati e loro modellazione
12	4	60	L	Class Diagram	Doing	Design di basso livello
12	4	90	A	Scrittura documento	Doing	Rifiniture finali
12	5	30	A	Effort Recording	Doing	Effort Recording
12	5	45	A	Rifinitura requisiti	Doing	Raffinamento dei requisiti
12	6			Termine Milestone #1		
12	10	90	L	Stato dell'arte	Doing	Stato dell'arte
12	11	120	A	Servizi	Doing	Raffinamento dei requisiti
12	12	180	A	Requisiti non funzionali	Doing	Raffinamento dei requisiti
12	13	180	L	Use Case Diagrams	Doing	Raffinamento dei requisiti
12	14	150	A	Scenari d'uso dettagliati	Doing	Raffinamento dei requisiti
12	15	270	L	Component Diagram	Doing	Architettura software
12	16	120	L	Studio Sequence Diagram	Learning	Studio individuale
12	17	300	L	Sequence Diagram	Doing	Architettura software
12	17	150	A	Modello E/R	Doing	Dati e loro modellazione
12	17	90	AL	Class Diagram	Doing	Design di basso livello
12	18	120	A	Configurazione progetto	Doing	Implementazione
12	19	105	A	Progettazione API e routes	Doing	Implementazione
12	20	180	A	Modelli, migrations e factory	Doing	Implementazione
12	21	240	A	Controller	Doing	Implementazione
12	21	60	A	Studio Postman	Learning	Studio individuale
12	21	120	L	Explain how the FRs and the NFRs are satisfied by design	Doing	Raffinamento dei requisiti
12	22	90	A	Documentazione API con Postman	Doing	Implementazione
12	22	30	A	Appendyx Prototype	Doing	Implementazione
12	22	60	A	Pert Chart	Doing	Effort Recording
12	22	60	A	Studio Pert Chart	Learning	Effort Recording
12	23			Termine Deliverable #1		
12	30	15	AL	Call di allineamento	Doing	Studio del progetto

Ore Alessandro	38,1
Ore Lorenzo	28,3
Ore in comune	3,1
Ore totali	**72,5**
Ore Learning	15
Ore Doing	54

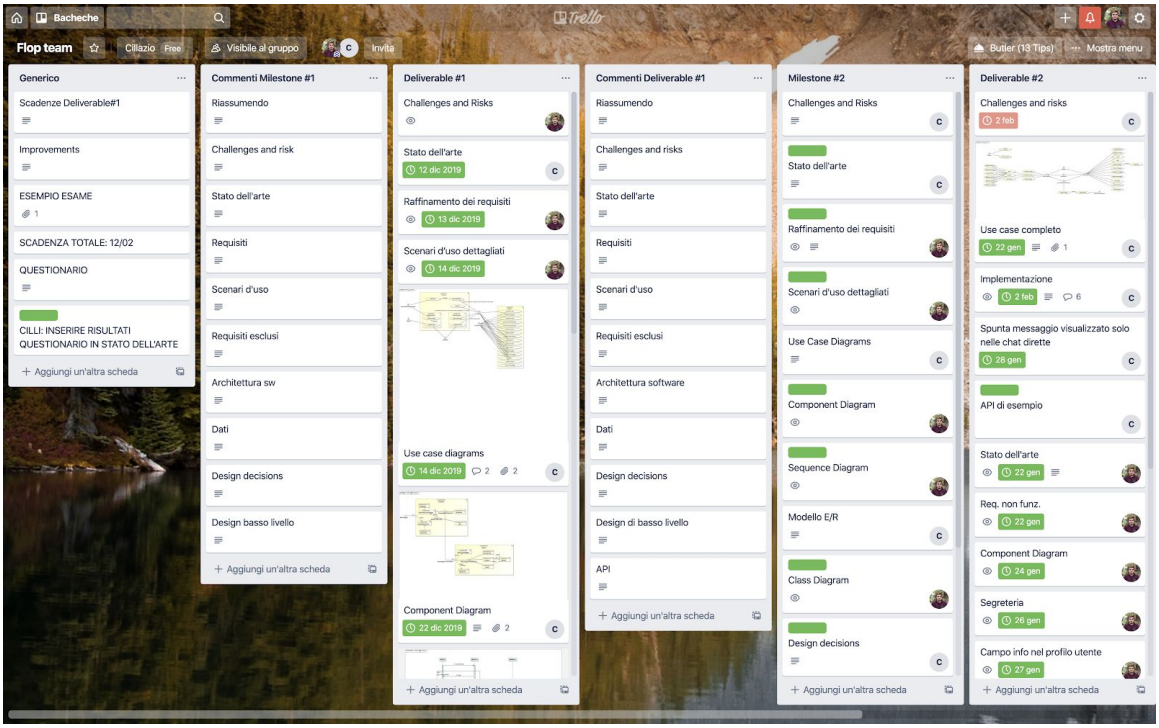
LogTemplate Milestone #2

Team (number and name): Flop team
Student name: Alessandro D'Orazio, Lorenzo Cilli
Student number: 251811, 253121
Email: alessandro.dorazio2@student.univaq.it, lorenzo.cilli@student.univaq.it

When		Time spent (min)	Partners	Brief Description of the performed task	Category	Sub-Category
1	1	15	AL	Call di allineamento	Doing	Studio del progetto
1	1	30	A	Organizzazione task per Milestone #2	Doing	Studio del progetto
1	5	15	AL	Challenging/Risky Requirements or Tasks	Doing	Challenging/Risky Requirements or Tasks
1	6	45	L	Stato dell'Arte	Doing	Stato dell'arte
1	7	45	A	Servizi	Doing	Raffinamento dei requisiti
1	8	120	A	Requisiti non funzionali	Doing	Raffinamento dei requisiti
1	9	15	A	Requisiti esclusi e assunzioni	Doing	Raffinamento dei requisiti
1	10	15	A	Scenari d'uso dettagliato	Doing	Raffinamento dei requisiti
1	11	120	A	Component Diagram	Doing	Architettura software
1	12	150	A	Sequence Diagram	Doing	Architettura software
1	13	90	L	Use-Case Diagram	Doing	Raffinamento dei requisiti
1	14	60	L	Modello ER	Doing	Dati e loro modellazione
1	15	150	A	Class Diagram	Doing	Design di basso livello
1	17	45	A	Modifiche API e lista routes	Doing	Implementazione
1	17	90	L	Design Decision	Doing	Design di basso livello
1	18	60	L	Studio Pert	Learning	Effort Recording
1	18	60	L	Pert Chart	Doing	Effort Recording
1	18	30	AL	Rifinitura documento	Doing	
1	21	15	AL	Call di allineamento	Doing	Studio del progetto
1	21	45	A	Inserimento risposte questionario e perfezionamento stato dell'arte	Doing	Stato dell'arte
1	21	90	A	Requisiti non funzionali	Doing	Raffinamento dei requisiti
1	22	75	A	Component Diagram	Doing	Architettura software
1	25	30	A	Requisiti esclusi e assunzioni		
1	27	15	A	Anno di corso -> immatricolazione	Doing	Raffinamento dei requisiti
1	30	150	A	Sequence Diagram	Doing	Architettura software
2	6	480	A	Interfaccia grafica	Doing	Implementazione
2	7	60	L	Use-Case Diagram	Doing	Raffinamento dei requisiti
2	8	90	A	Come i requisiti funzionali e non funzionali sono soddisfatti dal design	Doing	NFR e FR soddisfatti dal design
2	9	210	L	Implementazione (model)	Doing	Implementazione
2	9	15	L	Modello ER	Doing	Dati e loro modellazione
2	10	120	A	Modifiche all'implementazione	Doing	Implementazione
2	11	210	L	Implementazione (controller)	Doing	Implementazione
2	11	45	L	Appendyx prototype	Doing	
2	12	60	L	Documentazione API	Doing	Implementazione
2	12	30	A	Design Decisions		
2	12	30	A	Revisione documento		
2	12	30	L	Pert Chart	Doing	Effort Recording
2	12	60	L	Implementazione	Doing	Implementazione
2	13	60	A	Studio Gantt	Learning	Effort Recording
2	13	60	A	Effort Recording	Doing	Effort Recording
2	13	15	L	Challenging/Risky Requirements or Tasks	Doing	Challengin/Risky Requirements or Tasks

Ore Alessandro	32,8
Ore Lorenzo	18,5
Ore in comune	1,3
Ore totali	**53,8**
Ore Learning	2
Ore Doing	49

L'organizzazione è avvenuta tramite chiamate Skype, incontri e la gestione di una bacheca su Trello.


Bacheche
Flop team
Cillazio Free
Visibile al gruppo
Invita
Butler (13 Tips)
Mostra menu
Generico
Scadenze Deliverable#1
Improvements
ESEMPIO ESAME
SCADENZA TOTALE: 12/02
QUESTIONARIO
CILLI: INSERIRE RISULTATI QUESTIONARIO IN STATO DELL'ARTE
Aggiungi un'altra scheda
Commenti Milestone #1
Riassumendo
Challenges and risk
Stato dell'arte
Requisiti
Scenari d'uso
Requisiti esclusi
Architettura sw
Dati
Design decisions
Design basso livello
Aggiungi un'altra scheda
Deliverable #1
Challenges and Risks
Stato dell'arte
12 dic 2019
Raffinamento dei requisiti
13 dic 2019
Scenari d'uso dettagliati
14 dic 2019
Use case diagrams
14 dic 2019
Component Diagram
22 dic 2019
Aggiungi un'altra scheda
Commenti Deliverable #1
Riassumendo
Challenges and risks
Stato dell'arte
Requisiti
Scenari d'uso
Requisiti esclusi
Architettura software
Dati
Design decisions
Design di basso livello
API
Aggiungi un'altra scheda
Milestone #2
Challenges and Risks
Stato dell'arte
Raffinamento dei requisiti
Scenari d'uso dettagliati
Use Case Diagrams
Component Diagram
Sequence Diagram
Modello E/R
Class Diagram
Design decisions
Aggiungi un'altra scheda
Deliverable #2
Challenges and risks
2 feb
Use case completo
22 gen
Implementazione
2 feb
Spunta messaggio visualizzato solo nelle chat dirette
28 gen
API di esempio
Stato dell'arte
22 gen
Req. non funz.
22 gen
Component Diagram
24 gen
Segreteria
26 gen
Campo info nel profilo utente
27 gen
Aggiungi un'altra scheda

11. Appendix. Prototype

Il prototipo, sviluppato con il framework Laravel, si basa su un pattern MVC-like. È presente sia la componente dei modelli che quella dei controller che quella delle viste.

Il prototipo segue la definizione dello schema del Class Diagram, includendo una classe PHP (sviluppata da Alessandro D'Orazio in un contesto lavorativo) che gestisce il ritorno dei dati in JSON, ottimizzando la scrittura del codice e le tempistiche grazie al riuso del codice.

Le operazioni permesse sono le CRUD (create, read, update, delete) per le seguenti entità: utenti, università, dipartimenti, facoltà, stanze e messaggi. I social vengono inseriti tramite l'update dell'utente.

La sicurezza viene garantita tramite JSON Web Token con scadenza di 1 ora e possibilità di refresh del token avendo il token precedente. Questo consente di non utilizzare le sessioni, ed offre un'elasticità maggiore rispetto ai metodi tradizionali per la gestione dell'autenticazione.

Quando si invia un messaggio, avviene un controllo dal server che, prima di inserire il messaggio nel db, verifica se l'utente è silenziato, oppure se ha utiliizzato una parola vietata.

Le API sono strutturate seguendo le linee guida di Laravel, utilizzando quindi un pattern {url}/api/{entity}/{action}

La fase di testing viene effettuata tramite le factory di Laravel, e una breve sessione di testing con casi limite. Abbiamo provato a caricare centinaia di migliaia di messaggi, contenuti in migliaia di stanze, senza ottenere problematiche di performance.

Nota per la documentazione mediante Postman
Poiché la preferenza del team è di sviluppare un prodotto creando i componenti, testandoli ed estendendoli, alcune funzionalità, come l'implementazione del Web Sockets non sono state introdotte. Riteniamo che la soluzione migliore sia di fornire dei componenti funzionanti al 100%, piuttosto che un componente sviluppato in poco tempo ma non funzionante.
Le API di un utente autenticato possiedono nell'header una regola "X-Authorization" contenente il token dell'utente autenticato. In questo modo è possibile sapere le informazioni dell'utente e gestire correttamente i permessi.

Sono presenti tutte le chiamate fornite da Laravel come resource (index, show, create, update, store, delete) per università, dipartimenti, corsi di laurea, utenti e stanze.
Sono stati oscurati gli id.
Oltre alle API già discusse sono state implementate quelle ancora inserite come ricerca stanze secondo un determinato dato che viene inserito quale nome stanza.
È possibile anche ricercare un utente utilizzando il suo nome, cognome, facoltà o anno di corso.
Un altro esempio può essere l'archiviazione di una stanza e la sua dearchiviazione, o la contrassegnazione di un messaggio importante.

Documentazione mediante Postman
https://documenter.getpostman.com/view/10198106/SzKN22aU?ver

L'autorizzazione avviene tramite il Bearer token, impostato nell'header.

Nella documentazione Postman non sono presenti tutte le API presenti nel sistema.

Lista delle routes

Domain	Method	URI	Name	Action	Middleware
	POST	api/admin/ban	admin.ban_user	App\Http\Controllers\AdminController@ban_user	api
	GET|HEAD	api/auth/info	info	App\Http\Controllers\AuthController@me	api
	POST	api/auth/login	login	App\Http\Controllers\AuthController@login	api
	GET|HEAD	api/auth/login	login	App\Http\Controllers\AuthController@login	api
	POST	api/auth/logout	logout	App\Http\Controllers\AuthController@logout	api
	POST	api/auth/register	register	App\Http\Controllers\AuthController@register	api
	POST	api/departments/{department_id}/destroy	department.destroy	App\Http\Controllers\DepartmentController@destroy	api
	POST	api/faculties	faculties.store	App\Http\Controllers\FacultyController@store	api
	GET|HEAD	api/faculties	faculties.index	App\Http\Controllers\FacultyController@index	api
	POST	api/faculties/{faculty_id}/destroy	faculty.destroy	App\Http\Controllers\FacultyController@destroy	api
	GET|HEAD	api/faculties/{faculty}	faculties.show	App\Http\Controllers\FacultyController@show	api
	PUT|PATCH	api/faculties/{faculty}	faculties.update	App\Http\Controllers\FacultyController@update	api
	POST	api/messages/{message_id}/find	message.find	App\Http\Controllers\MessageController@find	api
	POST	api/reports	reports.store	App\Http\Controllers\ReportController@store	api
	GET|HEAD	api/reports	reports.index	App\Http\Controllers\ReportController@index	api
	GET|HEAD	api/reports/user/{user_id}/all	reports.filter_by_user	App\Http\Controllers\ReportController@filterByUser	api
	PUT|PATCH	api/reports/{report}	reports.update	App\Http\Controllers\ReportController@update	api
	GET|HEAD	api/reports/{report}	reports.show	App\Http\Controllers\ReportController@show	api
	POST	api/rooms	rooms.store	App\Http\Controllers\RoomController@store	api
	GET|HEAD	api/rooms	rooms.index	App\Http\Controllers\RoomController@index	api
	GET|HEAD	api/rooms/archived	rooms.archived	App\Http\Controllers\RoomController@archived	api
	POST	api/rooms/search	rooms.find	App\Http\Controllers\RoomController@find	api
	POST	api/rooms/{room_id}/admins/update	rooms.update_admins	App\Http\Controllers\RoomController@updateAdmins	api
	GET|HEAD	api/rooms/{room_id}/archives	rooms.archives	App\Http\Controllers\RoomController@archives	api
	POST	api/rooms/{room_id}/destroy	rooms.destroy	App\Http\Controllers\RoomController@destroy	api
	GET|HEAD	api/rooms/{room_id}/importantMessages	message.importantmessages	App\Http\Controllers\MessageController@importantMessages	api
	POST	api/rooms/{room_id}/messages	messages.store	App\Http\Controllers\MessageController@store	api
	GET|HEAD	api/rooms/{room_id}/messages/{message_id}/important	message.important	App\Http\Controllers\MessageController@important	api
	GET|HEAD	api/rooms/{room_id}/messages/{message}	messages.show	App\Http\Controllers\MessageController@show	api
	PUT|PATCH	api/rooms/{room_id}/messages/{message}	messages.update	App\Http\Controllers\MessageController@update	api
	POST	api/rooms/{room_id}/user/{user_id}/mute	rooms.mute_user	App\Http\Controllers\RoomController@muteUser	api
	PUT|PATCH	api/rooms/{room}	rooms.update	App\Http\Controllers\RoomController@update	api
	GET|HEAD	api/rooms/{room}	rooms.show	App\Http\Controllers\RoomController@show	api
	POST	api/universities	universities.store	App\Http\Controllers\UniversityController@store	api
	GET|HEAD	api/universities	universities.index	App\Http\Controllers\UniversityController@index	api
	GET|HEAD	api/universities/{university_id}/departments	departments.index	App\Http\Controllers\DepartmentController@index	api
	POST	api/universities/{university_id}/departments	departments.store	App\Http\Controllers\DepartmentController@store	api
	PUT|PATCH	api/universities/{university_id}/departments/{department}	departments.update	App\Http\Controllers\DepartmentController@update	api
	GET|HEAD	api/universities/{university_id}/departments/{department}	departments.show	App\Http\Controllers\DepartmentController@show	api
	POST	api/universities/{university_id}/destroy	university.destroy	App\Http\Controllers\UniversityController@destroy	api
	PUT|PATCH	api/universities/{university}	universities.update	App\Http\Controllers\UniversityController@update	api
	GET|HEAD	api/universities/{university}	universities.show	App\Http\Controllers\UniversityController@show	api
	POST	api/users	users.store	App\Http\Controllers\UserController@store	api
	GET|HEAD	api/users	users.index	App\Http\Controllers\UserController@index	api
	POST	api/users/filter		App\Http\Controllers\UserController@filterUsers	api
	POST	api/users/search	user.search	App\Http\Controllers\UserController@search	api
	POST	api/users/social/add	social.store	App\Http\Controllers\SocialController@store	api
	POST	api/users/{user_id}/destroy	user.destroy	App\Http\Controllers\UserController@destroy	api
	GET|HEAD	api/users/{user}	users.show	App\Http\Controllers\UserController@show	api
	PUT|PATCH	api/users/{user}	users.update	App\Http\Controllers\UserController@update	api

Lista delle routes

Link Youtube

https://youtu.be/NVRFJS1NG58