“Software Engineering”
Course
a.a. 2019-2020

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

<PROGETTO 3>

Date	<05/12/2019>
Deliverable	Documento Finale – D3
Deliverable	The rolling seals.

TEAM MEMBERS:

NAME & SURNAME	MATRICOLA	EMAIL
NOEMI SURRICCHIO	254437	SURRICCHIONOEMI@GMAIL.COM
SAMUEL FINOCCHIO	255380	SAMUEL.FINOCCHIO@STUDENT.UNIVAQ.IT
SIMONE BELLI	254437	BELLISIMONE9@GMAIL.COM

Table of Contents of this deliverable

List of Challenging/Risky Requirements or Task

CHALLENGING TASK	DATE THE TASK IS IDENTIFIED	DATE THE CHALLENGE IS RESOLVED	EXPLANATION ON HOW THE CHALLENGE HAS BEEN MANAGED
Authentication: Implementation of token based authentication and verification of the api requests using a middleware.	27/11/2019	N.D.	Research on the topic, study of existing open source implementations .
Managing "big" queries on a large dataset. Retrieving an accurate result set with low delay.	21/11/2019	22/11/2019	Succession of reductions on the dataset domain during the user interation with the search filters. Possible implementation of a caching system (unlikely).

A. Stato dell'Arte

Come primo approccio, per lo sviluppo del nostro sistema, abbiamo studiato e analizzato servizi già esistenti all'interno dei quali sono state selezionate le funzionalità che si avvicinano maggiormente alla nostra idea di progetto con l'intenzione di migliorarle. Abbiamo inoltre scartato alcune implementazioni ritenute poco efficienti o lontane dalla specifica di partenza. I servizi che abbiamo analizzato sono i siti web riportati qui sotto:

-DOCSITY (www.docsity.com):

è uno dei siti analizzati che si avvicina maggiormente alla nostra idea di progetto. Le funzionalità che abbiamo deciso di prendere in considerazione per migliorarle sono:

- Gestione dei punti per il download dei file caricati da altri utenti
- Metodologia per guadagnare punti:

  registrazione, completamento profilo, recensione dell'università di appartenenza, recensione esame, numero di download ricevuti.
- Migliore gestione dei file e della loro ricerca:

  ogni volta che un utente entrerà sul sito appariranno sempre in primo piano i documenti che riguardano il suo corso di laurea della sua università.

Le funzionalità che abbiamo ritenuto essere poco inerenti alle specifiche del progetto sono:

- Acquisto dei punti per il download dei file:

  abbonamenti settimanali, mensili o annuali.
- Post riguardanti lavoro, news, ecc.. (pubblicità).

-APPUNTI CONDIVISI (www.appunticondivisi.com):

sito web dal quale non abbiamo voluto prendere spunto perché risulta poco efficiente.

-STUDOCU (www.studocu.com):

è uno dei siti analizzati che si avvicina alla nostra idea di progetto. Le funzionalità che abbiamo deciso di prendere in considerazione per migliorarle sono:

- Gestione del profilo utente:

  livello utente che crescerà in base ai download che riceve, alle recensioni che vengono fatte sui suoi documenti e alla valutazione che gli altri utenti faranno su di esso.

- Aggiunta di alcuni feedback sui documenti per classificare la loro utilità.

-TESIONLINE (www.tesionline.it):

sito web dal quale non abbiamo voluto prendere spunto perché risulta poco efficiente.

Questi sono i siti più “importanti” che offrono a grandi linee lo stesso tipo di servizio. Abbiamo deciso di prendere spunto da alcune delle loro funzionalità per fari si che il nostro sistema sia facile da usare, funzioni nel migliore dei modi e offra un servizio utile all’utente.

B. Raffinamento dei Requisiti

A.1 Servizi (con prioritizzazione)

Possiamo dividere i servizi con prioritizzazione in 4 categorie: servizi sui documenti, operazioni di ricerca dei documenti, login e servizi di gamification. Qui sotto riportiamo in modo dettagliato la lista dei servizi offerti dal nostro sistema:

Operazioni sui documenti:

- Caricamento di link e/o documenti.
- Storage di documenti interno.
- Storage di documenti esterno.
- Download dei documenti.
- Decremento dei punti dell'utente dopo che ha scaricato un documento.
- Valutazione del documento caricato dall'utente.
- Preview dei documenti.
- Categorizzazione dei documenti caricati.
- Versioning dei documenti.
- Mostrare differenze tra le versioni dei documenti
- Ordinamento e classificazione delle versioni del documento stesso.
- Implementazione o integrazione di software esterni allo scopo di fornire tecniche di versioning.
- Storage di collegamenti ipertestuali.
- Assegnamento di licenze agli appunti e gli allegati.

- In caso di allegati eseguibili, verifica di attendibilità e sicurezza.

Operazioni di ricerca dei documenti:

- Ricerca intelligente in grado di effettuare ricerca su tag, titolo, categorie, ateneo, corso di laurea, anno di corso, docente, parole chiave ed eventualmente la ricerca all'interno del file.

- La ricerca deve dipendere da: l'utente registrato, l'ateneo di appartenenza e l'anno di corso a cui è iscritto.

Servizi di login:

- Autenticazione.

- Autenticazione indiretta.

- Autenticazione basata su token e bearer oppure autenticazione diretta, tramite query a base di dati, con session storage, cookie e crittografia asimmetrica.

Servizi di Gamification:

- Assegnazione di un punteggio premio alle operazioni effettuabili dagli utenti e sulle ripercussioni che hanno sulla piattaforma: registrazione utente, completamento profilo utente, recensione università di appartenenza, recensione corso e numero di download che vengono effettuati su un documento.

- Aumento del livello utente in base alle sue attività sulla community. I download ricevuti,

- Realizzazione automatica e dinamica di una classifica degli utenti in base ai punteggi ponderati con i feedback relativi ai loro contenuti.

- Assegnazione feedback agli utenti al fine di mostrare agli utilizzatori una stima della qualità media che ci si può aspettare dal materiale di un definito utente.

A.2 Requisiti non Funzionali

- Autenticazione plug in: permettere di delegare a terzi il servizio di integrazione e in alternativa fornire un sistema interno di autenticazione.

- Capacità di memorizzare un numero elevato di dati. (I dati da memorizzare all'interno del nostro db sono nell'ordine di 1E+10).

- Verifica dell'esistenza e della sicurezza dei riferimenti (link) esterni.

- Tempi di esecuzione non troppo lunghi per poter permettere un servizio tanto veloce quanto efficiente.

A.3 Scenari d'uso dettagliati

Esempi scenari d'uso:

1) Marco nuovo studente dell'UNIVAQ, venuto a conoscenza grazie ai suoi amici di corso dell'esistenza del nostro servizio, decide di dare un occhiata al sito per valutare l'idea di iscriversi e iniziare a scaricare file e condividere appunti e documenti con la community. Prima di registrarsi decide di cercare il corso di laurea della sua università e gli esami relativi ad esso per avere conferma dell'esistenza di materiale da scaricare. Una volta che ha visto il materiale (tramite le preview sui documenti, gli utenti non registrati possono fare uso di questa implementazione) e ha valutato il livello dei documenti decide di iscriversi.

2) Una volta che Marco si è iscritto decide di completare il suo profilo (ad esempio inserendo una sua foto, i suoi hobbies, sport praticati exc…), recensire la propria università e il corso appena frequentato per guadagnare dei punti che utilizzerà per scaricare appunti o altri file.

3) Seguendo le lezioni Marco decide di scrivere degli appunti che caricherà sulla community con i dati relativi a quel tipo di

documento. Ad esempio caricherà “Appunti del corso di Laurea in informatica, UNIVAQ - esame: ingegneria del software - docente: Henry Muccini - anno accademico: 2018-2019” e deciderà il costo, in punti, del documento. Marco non potrà scegliere un valore casuale ma avrà a disposizione un range di valori che varierà con l’aumentare del suo livello utente.

4) Il livello utente aumenta in base ai documenti caricati, alle recensioni effettuate, ai download (fatti e ricevuti) e al numero di stelle, assegnate ai propri documenti, ricevute dagli altri utenti. I valori che saranno fondamentali per l’aumento del livello sono: i download che vengono effettuati sui documenti caricati dall’utente e il numero di recensioni che egli riceve. Questo implica che (nella maggior parte dei casi) se il livello di un utente è alto allora gli appunti e i documenti che condivide sono di buona qualità. Ovviamente l’assegnazione dei punti per il download non sarà per forza alta ma sarà l’utente stesso a stabilire il costo del documento in base ad una sua autovalutazione. Più il livello utente sarà alto più il range di valori di assegnazione punti diventerà più grande. Ad esempio tizio_x ha un livello utente = 1 potrà assegnare un punteggio al file caricato che varia da 50 a 55 mentre tizio_y che ha un livello utente=10 ha a disposizione un range che va da 55 a 90 (ad esempio).

5) Dopo alcuni mesi Marco controlla sul suo profilo quanti download e quante recensioni ha ricevuto il documento che aveva caricato. Marco guadagnerà dei punti per ogni download effettuato su quel documento e potrà rispondere ad eventuali domande che gli verranno fatte nelle recensioni.

6) Marco non può frequentare le lezioni relative al corso di sistemi operativi e decide di cercare appunti. Entra nel sito e decide di filtrare i documenti scegliendo anno, corso di laurea, esame e

docente. Facendo ciò avrà come risultato due appunti di studenti diversi e decide di valutare quale dei due scaricare. Come farà Marco a valutare gli appunti? Utilizzerà sicuramente la preview del documento ma andrà anche a controllare il profilo dell'utente che ha caricato il file vedendo il suo livello utente, la media di stelle generale degli appunti caricati ed il numero di like che ha il documento. Marco noterà anche che il costo per il download dei 2 appunti è diverso. Questo è dovuto al fatto che l'appunto con il costo più alto viene valutato come un "appunto di ottimo livello" dal sistema e dall'utente che lo carica. Una volta scelto il file da scaricare Marco perderà un numero di punti pari a quelli relativi al costo di download del documento scelto.

7) Marco cerca degli appunti filtrando la sua università e il suo corso di laurea ma non è soddisfatto della qualità dei risultati quindi cerca altri appunti di altri utenti che studiano in altre Università. Dopo aver analizzato i risultati decide di scaricare documenti provenienti dall'Università di Milano.

8) Una volta superato l'esame o una volta che Marco avrà finito di studiare sul documento scaricato potrà inserire una recensione.

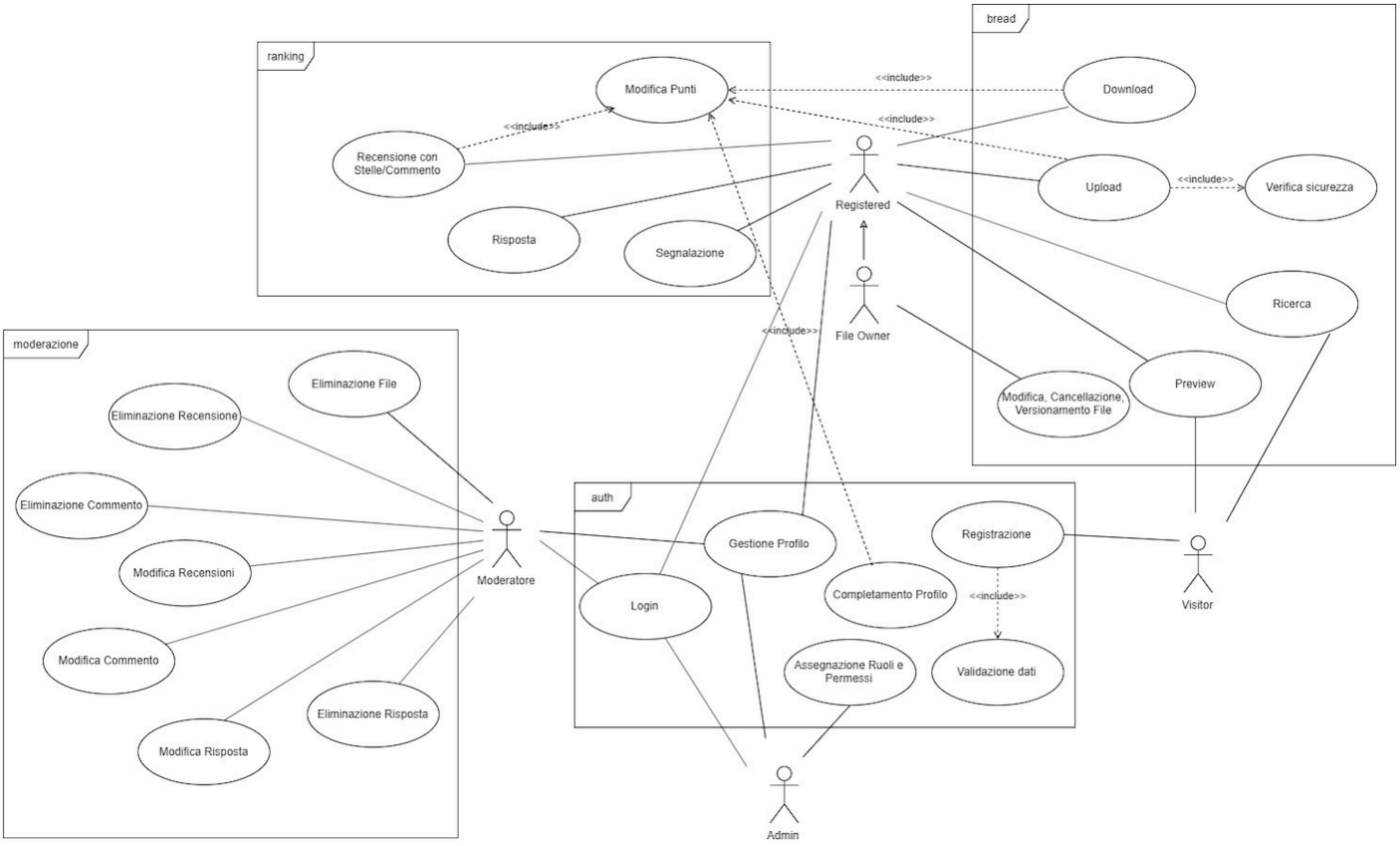
9) Inoltre Marco potrà confrontare il suo livello con quello degli altri tramite una classifica degli utenti più collaborativi.

A.4 Excluded Requirements

I requisiti che abbiamo deciso di escludere dal nostro progetto sono:

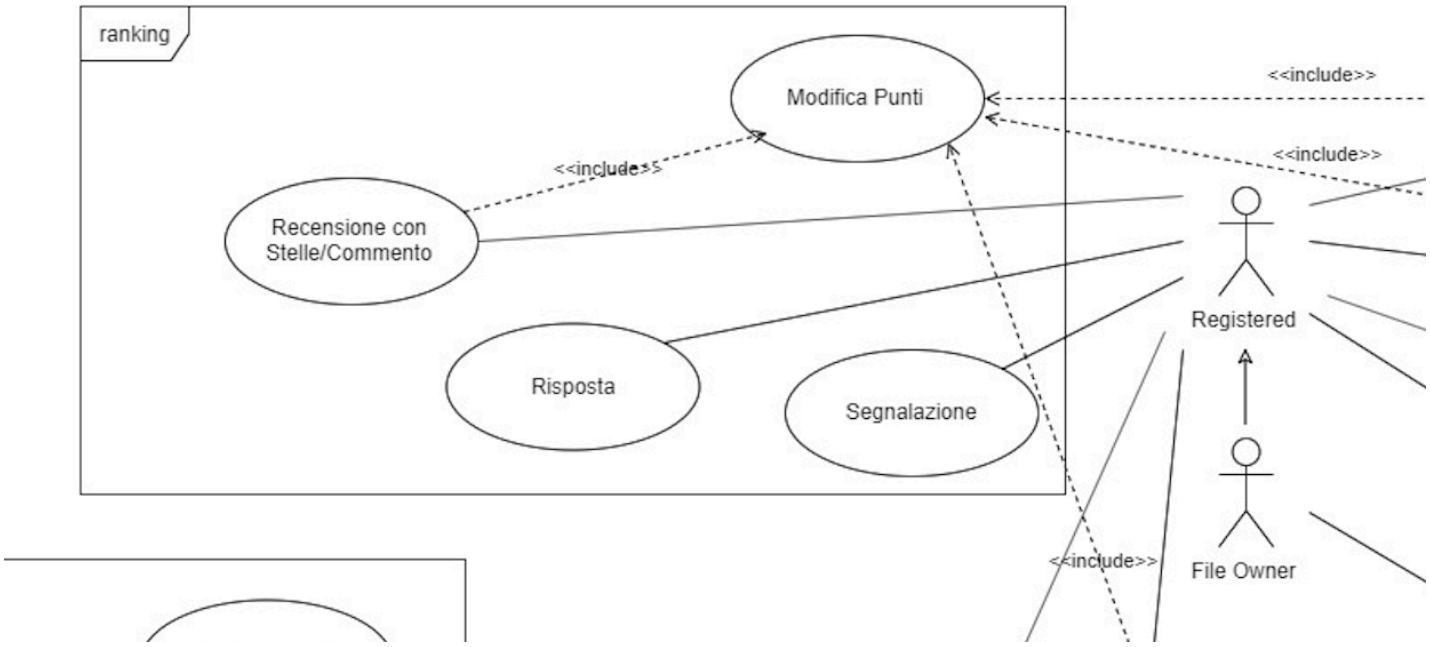
- Statistica descrittiva a posteriori.
- Analisi dei dati.
- Non è possibile acquistare punti tramite abbonamenti mensili/settimanali/annuali.

A.6 Use Case Diagrams

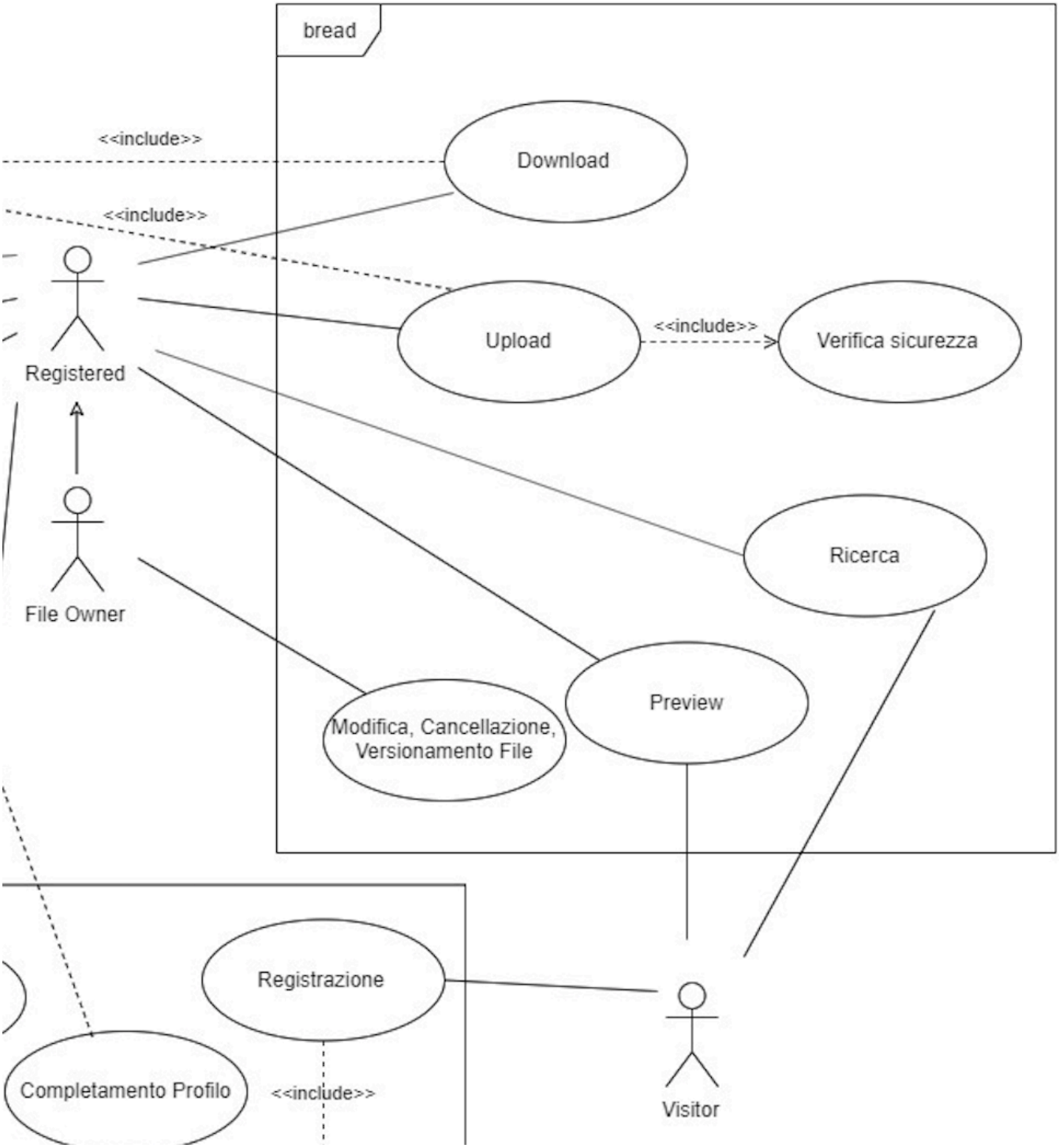


Attori coinvolti: Registered(utente registrato), Visitor(utente non registrato), Moderatore(Utente registrato con poteri descritti nello use case), Admin(Utente registrato con poteri descritti nello use case).

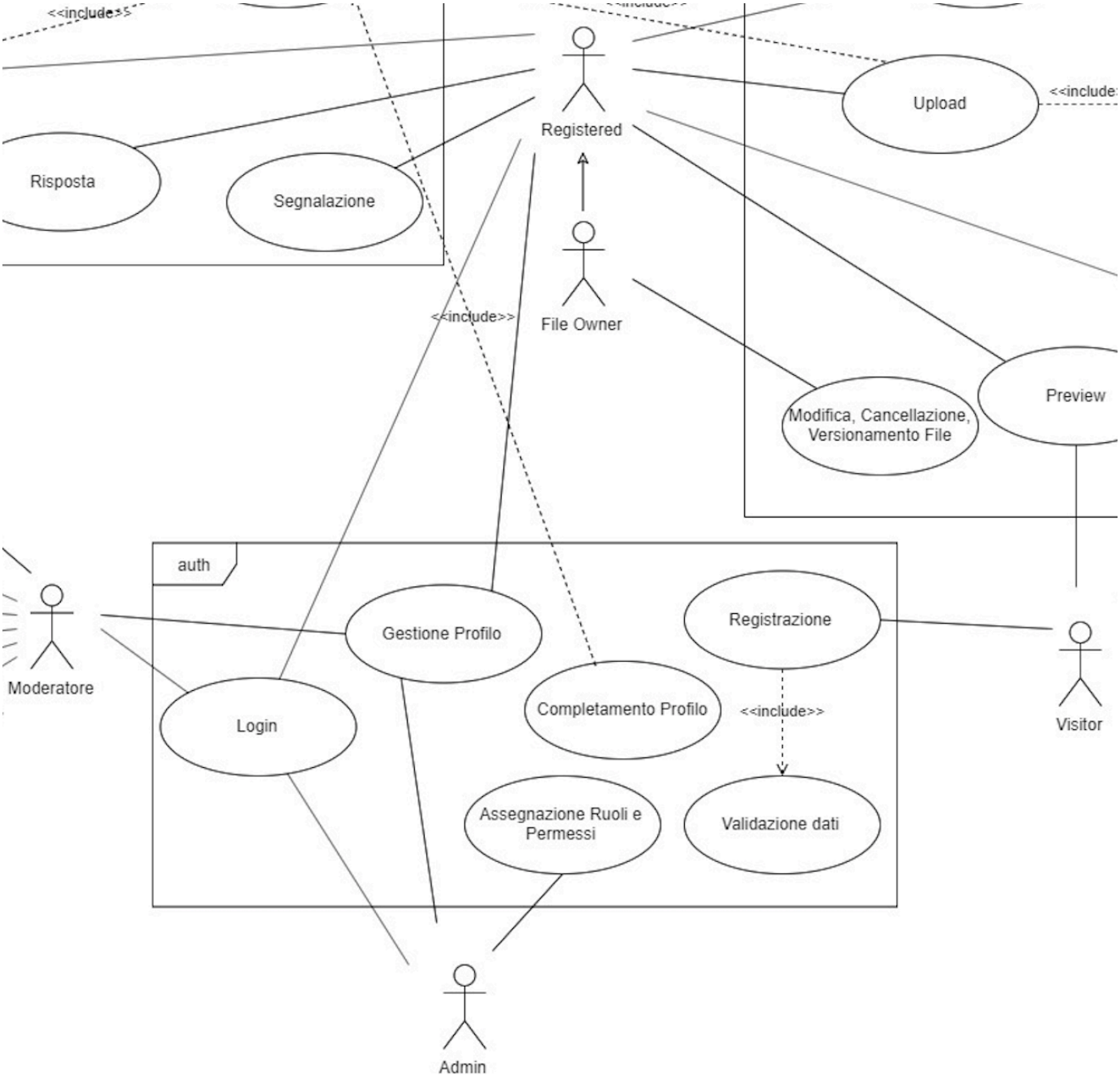
Zoom modulo “ranking” per l’attore registered:



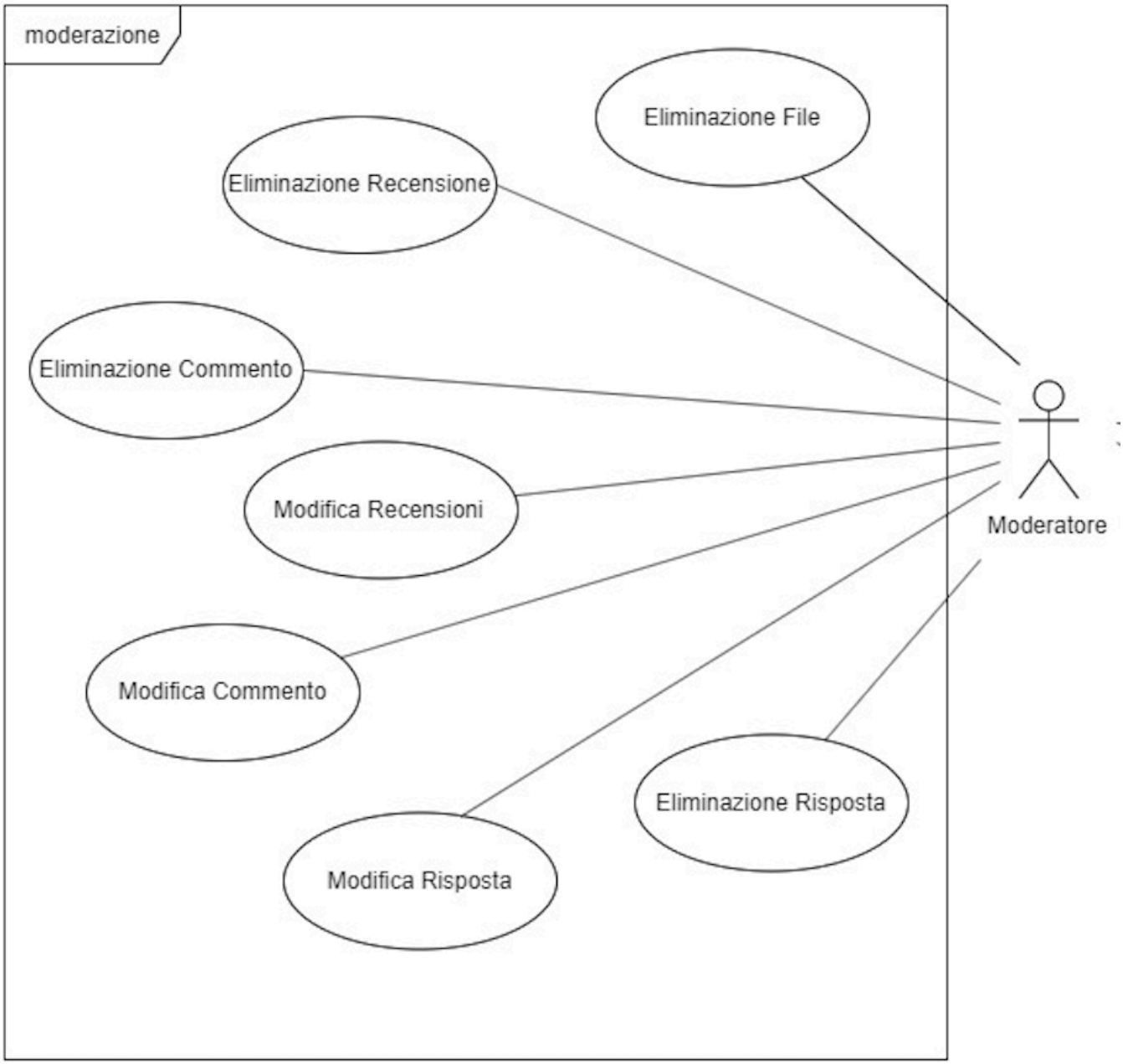
Zoom modulo “bread” per l’attore registered e visitor:



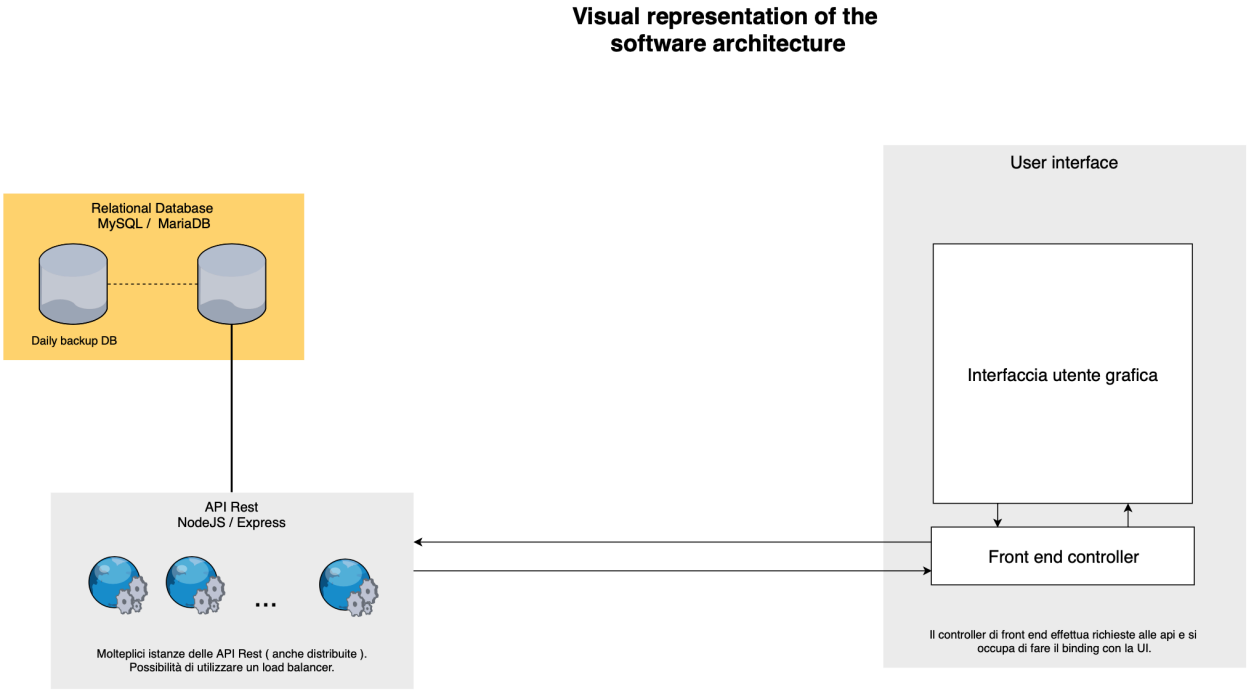
Zoom modulo “auth” per tutti gli attori coinvolti:



Zoom use case diagram “Moderazione” per l’attore moderatore:

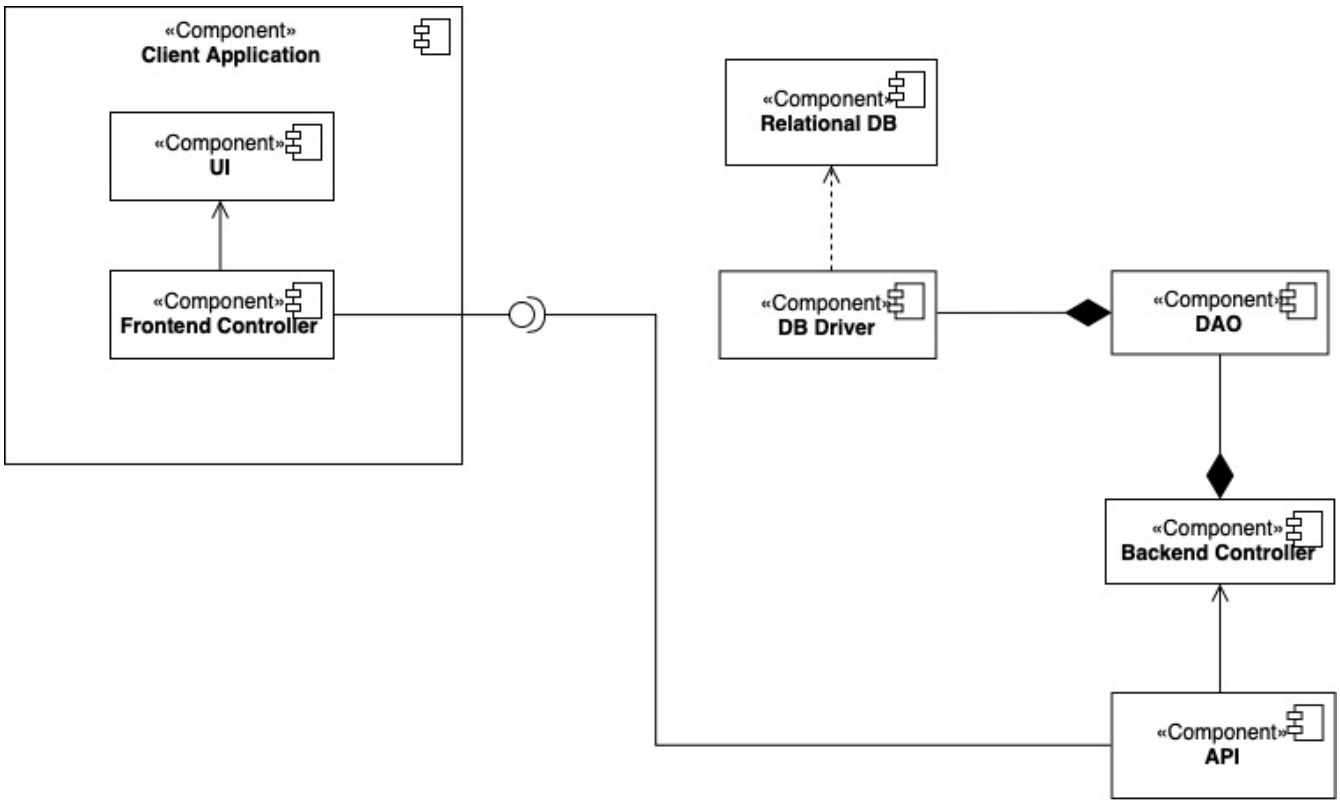


C. Architettura Software

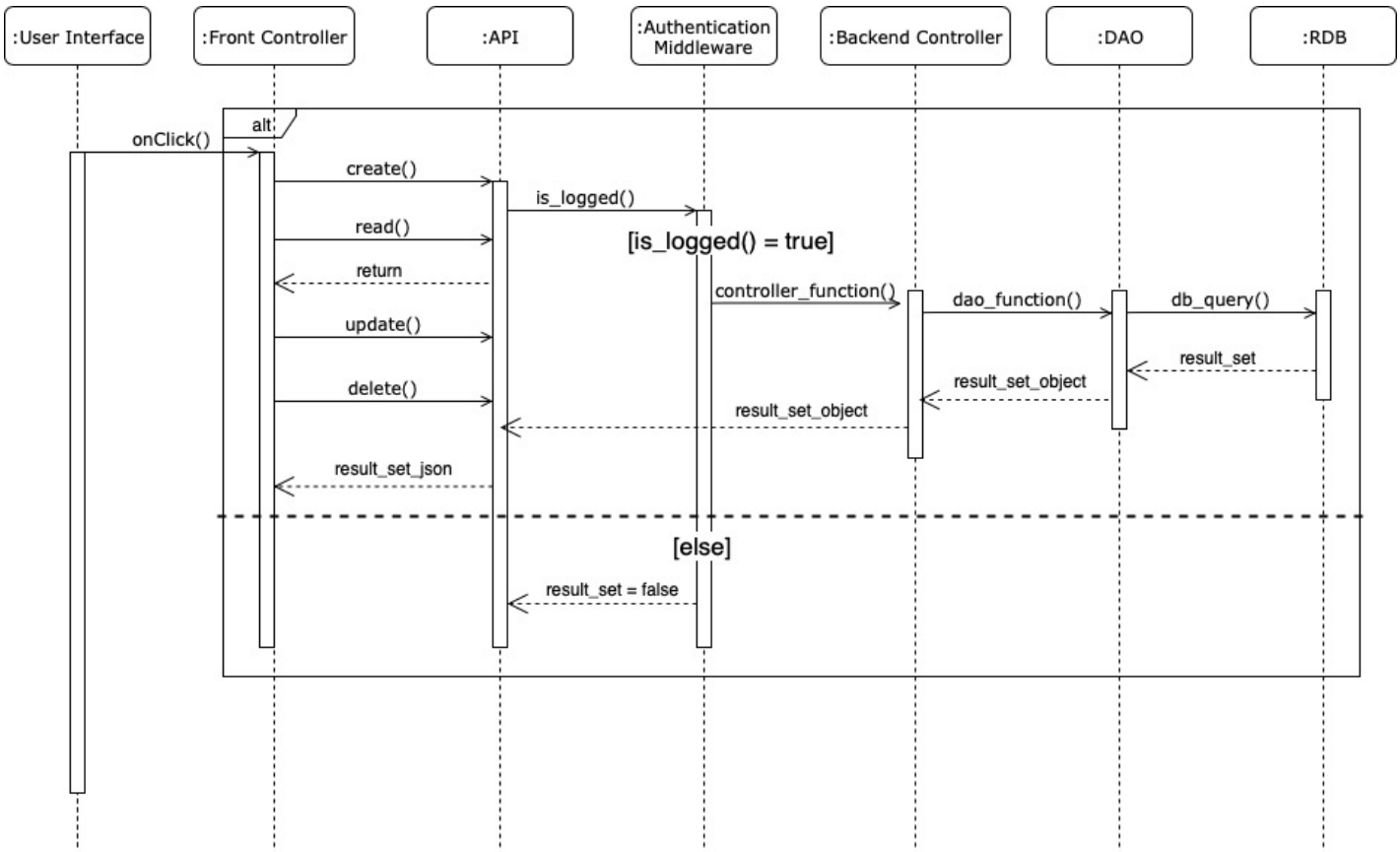


La separazione del software allo scopo di ottenere una riusabilità ottimale ci ha portato ad effettuare una classica separazione db e api dalla logica di presentazione. A tale scopo colleghiamo ad una base di dati relazionale MySQL/MariaDB* ad una API rest. La API rest idealmente stateless lavora su protocollo HTTP fornendoci una facile interazione per il frontend. Il frontend verrà realizzato utilizzando un framework basato sulla moderna tecnologia ECMAScript (js): Vue.js*.

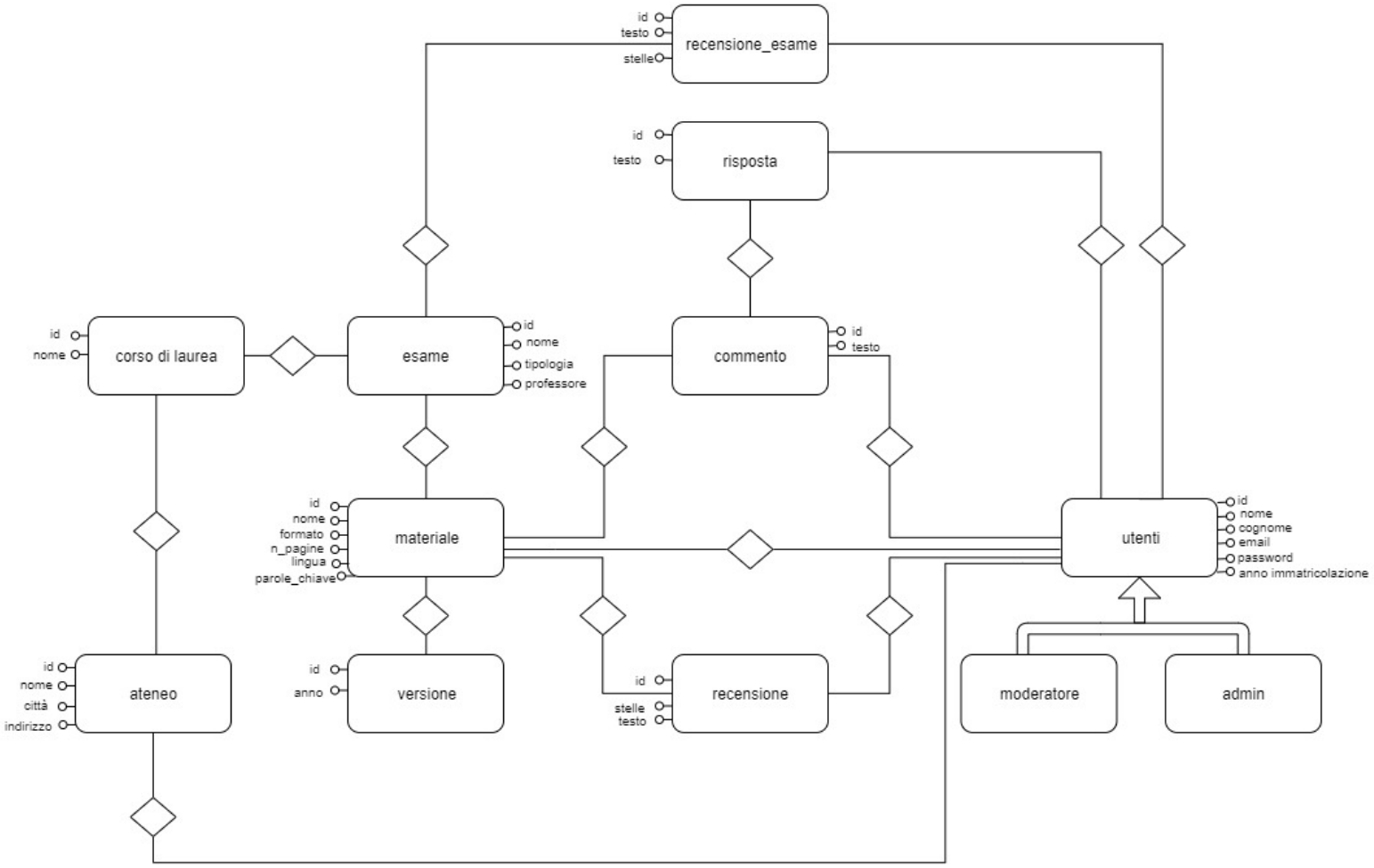
C.1 The static view of the system: Component Diagram



C.2 The dynamic view of the software architecture: Sequence Diagram



D. Dati e loro modellazione

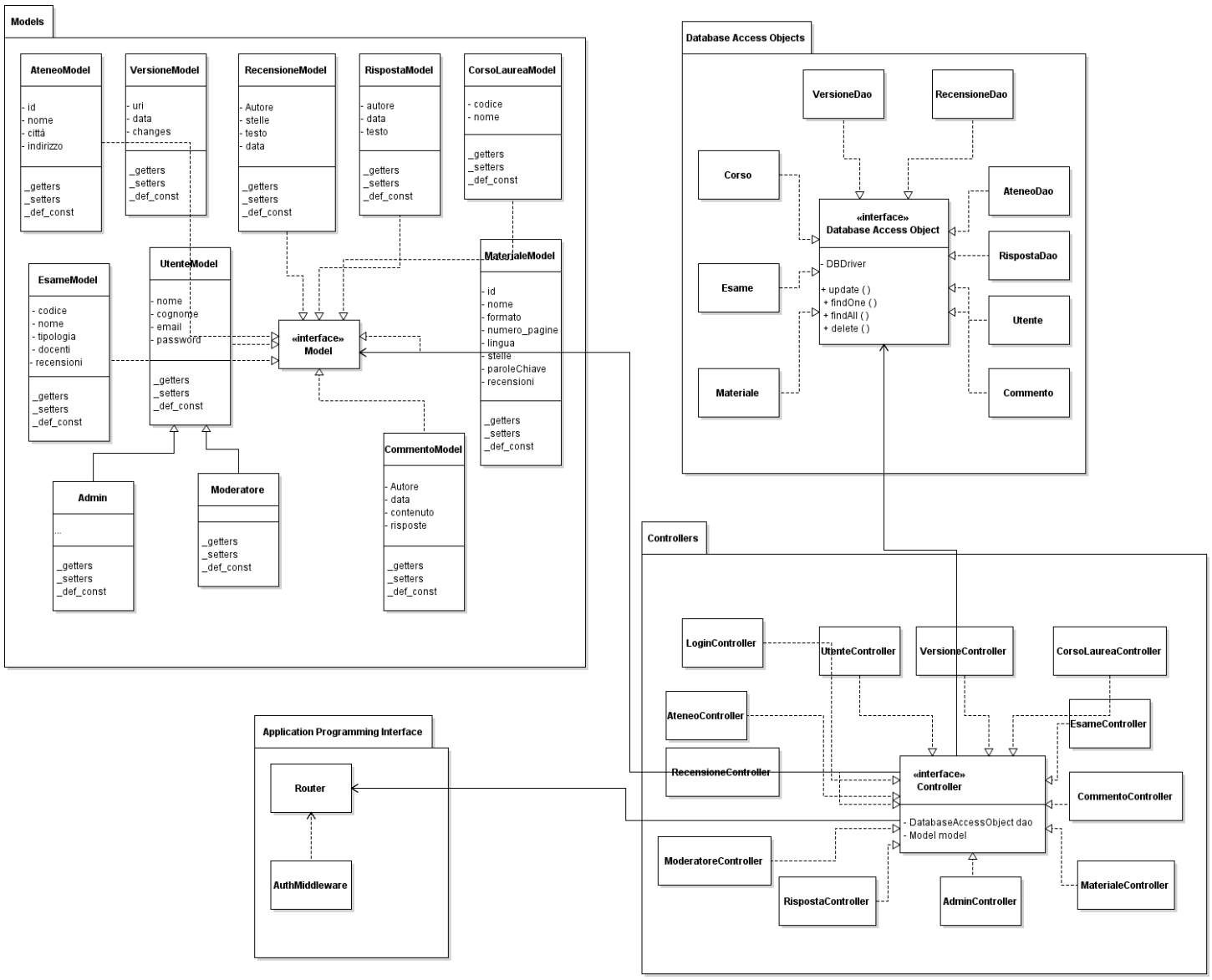


E. Design Decisions

Allo scopo di ottenere codice riutilizzabile è stato necessario uno studio sulla componentizzazione del codice al fine di ottenere un numero più alto possibile di componenti atomici indipendenti dal resto del sistema. I dati su cui il software dovrà lavorare sono fortemente relazionati, abbiamo inoltre verificato che gran parte delle funzionalità del software richiedono un collegamento forte tra le varie entità. Questo ci ha portato a ragionare con un modello relazionale dei dati escludendo quindi l'utilizzo di una base di dati non tradizionale (RDB). Non avendo il controllo sull'implementazione completa del sistema, è stato deciso di separare completamente il backend incapsulandolo in una API al fine di ottenere un accoppiamento lasco tra backend e frontend. Questa scelta permetterà di integrare la restante parte del software rimpiazzando il frontend. Non avendo certezze sulle tecnologie utilizzate per l'autenticazione, assegneremo ad un middleware residente sulle API un controller di autenticazione, il quale potrà essere modificato per soddisfare qualunque necessità tecnologica riguardante l'autenticazione. La API rest verrà implementata in NodeJS.
NodeJS permette di realizzare una architettura event driven con I/O non bloccante, il che è molto efficiente nella gestione di carichi elevati.
Per NodeJS esistono diversi sistemi esistenti che permettono un monitoraggio molto dettagliato delle richieste. Sono inoltre di facile implementazione api rest grazie ad expressjs, il quale supporta implementazioni di autenticazioni token based a noi necessarie. La maturità della tecnologie e il suo largo impiego inoltre ci assicurano di trovare librerie implementate per i più disparati impieghi, supporto tecnico sulle più utilizzate piattaforme (stack overflow) e server pensati per ospitare applicazioni node (Azure, digital ocean e AWS). VueJS è un framework javascript, leggero, veloce e di semplice implementazione e utilizzo. La sua modularità ci consente di non aggiungere inutile complessità alla realizzazione della logica di frondend la quale è secondaria alla realizzazione dello stack di backend. Fattore determinante alla scelta di Vuejs è stato anche il livello di dettaglio con la quale la documentazione ufficiale è scritta.

F. Design di Basso Livello

CLASS DIAGRAM



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

Il requisito da noi messo al primo posto è un requisito non funzionale: la riusabilità del codice. Durante tutte le fasi di progettazione è stato messo in primo piano il focus della separazione delle componenti a livello strutturale. La base di dati NON deve assolutamente vincolare la portabilità del progetto, deve essere possibile fare un porting del progetto su una tecnologia completamente differente di base di dati, anche non relazionale come a grafo, a documento o ad oggetti. A tale scopo il software fa accesso alla base dei dati esclusivamente tramite componenti DAO (database access object). Ciò permette in caso di cambio di tecnologia di base di dati la sola sostituzione dei componenti DAO. Ciò inoltre ci permette di cambiare tecnologia in fase di testing in caso la attuale tecnologia non sia soddisfacente a livello prestazionale essendo la base di dati la fonte principale di bottleneck in un sistema data centrico come il nostro. I requisiti funzionali sono soddisfatti dai metodi implementati nei controller. Elencarli singolarmente sarebbe esageratamente prolisso e scontato. Il path della risoluzione di un requisito funzionale, il che consiste nella totalità di casi ad accesso ad una porzione di dati segue il seguente percorso come descritto dal sequence diagram:

- Utente fa accesso alla UI
- UI esegue una funzione nel frontend controller
- Il frontend controller effettua una chiamata alle API accompagnata dal token di autenticazione.
- Le api fanno una chiamata ad un metodo del controller
- Il controller legge e scrive i dati utilizzando i componenti DAO* instanziando i model
- Il DAO accede al DBMS tramite un driver il quale si occupa di manipolare la base di dati

In caso di richieste complesse potrebbe essere necessario l'utilizzo di più DAO contemporaneamente in metodi presenti nei controller di backend.

L'autenticazione è realizzata dal LoginController e la verifica dell'autenticazione ad ogni richiesta alle API è effettuata dall'AuthMiddleware Il middleware è un intermediarlo tra la risoluzione della richiesta delle API e i controller di backend. Prima di effettuare una chiamata ad un metodo di un backend controller verrà verificata la autenticità e validità del token.

H. GANTT DIAGRAM