UNIVERSITÁ DEL SALENTO

Anno Accademico 2017/2018

FACOLTÁ DI INGEGNERIA
Corso di Laurea in Computer Engineering

SOFTWARE ENGINEERING

Progettazione e sviluppo di un sistema per la gestione di un dipartimento Universitario

Studente:

Niko Gentile
Christian Libetta

INDICE:

Analisi dei requisiti del Sistema

Traccia dell'elaborato

Il coordinatore della didattica di un dipartimento universitario vuole realizzare un sistema software a supporto dello svolgimento delle lezioni, utilizzabile anche da smartphone. Per questo motivo decide di introdurre una famiglia di applicazioni composta da tre esemplari: (1) una web app per la segreteria didattica che consenta alla medesima di popolare tutti i dati di dominio relativi ai differenti corsi di studio erogati dal dipartimento, gli insegnamenti per ogni corso di studio, i docenti, le lezioni, le aule atte a ospitare le lezioni in un ipotetico calendario didattico – associando alle aule anche informazioni di geo-localizzazione così da poterle visualizzare su una mappa, oltre che gli strumenti di supporto alla didattica ivi contenuti (wifi, proiettore, smartboad, ecc.), gli esami di profitto; (2) una web app per i docenti che consenta loro di distribuire materiale didattico di varia tipologia (diapositive pdf, immagini, link interessanti, bibliografia da consultare, ecc.), oppure di segnalare problematiche alla segreteria (del tipo, l'aula è troppo piccola, il proiettore non funziona, la wifi perde spesso la connessione, ecc.); (3) un'app mobile per gli studenti e per i docenti che consenta loro di interagire tramite una "stanza virtuale di discussione" (tipo Telegram), che il sistema supporta per ogni differente insegnamento. La web app deve consentire ai docenti di prendere visione dello stato di lavorazione delle segnalazioni che loro stessi hanno fatto sulle aule. Eventualmente il docente può decidere di prendere visione di tutte le segnalazioni di un'aula specifica, vedendo anche quelle eventualmente fatte da colleghi. La web app, dall'altra parte, deve consentire alla segreteria didattica di prendere in carico le segnalazioni e di gestirne il ciclo di vita fino alla risoluzione del problema (segnalazione presa in carico, segnalazione in lavorazione, segnalazione risolta, segnalazione rifiutata). All'atto della soluzione di una segnalazione, la segreteria didattica può specificare una nota; all'atto del rifiuto dell'esecuzione di una segnalazione la segreteria didattica deve fornire una motivazione. L'app mobile deve consentire agli studenti di scrivere messaggi per uno qualsiasi degli insegnamenti ai quali sono iscritti; i messaggi possono essere pubblici (cioè visibili a tutti gli iscritti al corso oltre che al docente) oppure privati (cioè diretti solo a uno specifico iscritto oppure al docente). L'app mobile deve consentire agli studenti di scaricare materiale didattico fornito dai docenti per ogni lezione. L'app mobile deve consentire agli studenti di fornire in forma anonima il proprio gradimento per ogni lezione specificando un valore da 1 a 5 e una nota, oltre che il proprio gradimento per ogni materiale didattico. L'app mobile deve consentire a docenti e studenti di prendere visione delle lezioni del giorno, accedendo interattivamente alla mappa delle aule. L'app mobile deve consentire ai docenti di ricevere notifiche ogniqualvolta la segreteria didattica faccia uno spostamento di aula oppure modifichi lo stato di una delle segnalazioni (ogni docente riceve le notifiche relative alle proprie segnalazioni o alle proprie aule), oppure di ricevere notifiche ogniqualvolta gli studenti aggiungano messaggi o gradimenti a insegnamenti e lezioni (ogni docente riceve le notifiche relative ai propri insegnamenti e alle proprie lezioni). L'app mobile deve consentire agli studenti di ricevere notifiche ogniqualvolta la segreteria didattica faccia uno spostamento di aula (ogni studente riceve le notifiche relative alle aule degli insegnamenti ai quali è iscritto), oppure di ricevere notifiche ogniqualvolta i docenti aggiungano messaggi o materiali didattici a insegnamenti e lezioni (ogni studente riceve le notifiche relative agli insegnamenti ai quali è iscritto).

Elenco dei casi d'uso

La realizzazione del sistema software richiede il coinvolgimento di quattro attori chiave :

- *Segretario*: è colui che attraverso una web app, popola tutti i dati di dominio relativi ai differenti corsi di studio erogati dal dipartimento, gli insegnamenti per ogni corso di studio, i docenti, le lezioni, le aule atte a ospitare le lezioni in un ipotetico calendario didattico – associando alle aule anche informazioni di geo-localizzazione così da poterle visualizzare su una mappa, oltre che gli strumenti di supporto alla didattica (wifi, proiettore, smartboad, ecc.), gli esami di profitto.
  Inoltre, il segretario prende in carico le segnalazioni e ne gestisce il ciclo di vita fino alla risoluzione del problema (segnalazione presa in carico, segnalazione in lavorazione, segnalazione risolta, segnalazione rifiutata). All'atto della soluzione di una segnalazione, la segreteria didattica può specificare una nota, mentre all'atto del rifiuto dell'esecuzione di una segnalazione la segreteria didattica deve fornire una motivazione.

- *Docente*: attraverso l'utilizzo di una web app può caricare materiale didattico di varia tipologia, oppure di segnalare problematiche alla segreteria aprendo un ticket per una segnalazione e controllare lo stato di lavorazione.
  Il docente prende visione di tutte le segnalazioni di un'aula specifica, vedendo anche quelle eventualmente fatte da colleghi. Inoltre, ha la possibilità di utilizzare un app mobile, che gli consente di visualizzare le lezioni del giorno attraverso un calendario interattivo, di accedere alle mappe per trovare l'aula e di chattare con gli studenti del proprio insegnamento in modo sia privato che pubblico attraverso chat di gruppo.
  Il docente ricevere attraverso l'app mobile notifiche se ci sono aggiornamenti sullo stato di lavorazione di una segnalazione fatta alla segreteria o un qualsiasi cambiamento che riguardi le lezioni, l'esame, un aula del suo insegnamento e ogni qual vota uno studente esprima un indice di gradimento sulle lezioni o sul materiale didattico da lui fornito.

- *Studente*: ha a disposizione un app mobile per visualizzare le lezioni del giorno attraverso un calendario interattivo accedendo alle mappe per trovare l'aula e chattare con gli studenti/docenti del proprio insegnamento in modo sia privato che pubblico attraverso chat di gruppo.
  Lo studente può inoltre, con una recensione, esprimere il suo voto di gradimento con una nota sul materiale messo a disposizione dal docente per il download o sulla lezione svolta, totalmente in modo anonimo.

Il sistema fornisce la possibilità ad ogni attore di compiere delle azioni ben precise, in modo da soddisfare le richieste del committente. Senza fornirne un'analisi dettagliata, si fornisce un semplice elenco dei casi d'uso di ogni attore:

APPs MOBILE

- ***Studente***:
    - Login;
    - Visualizza aule;
    - Visualizza calendario;
    - Visualizza materiale didattico;
    - Download materiale didattico;
    - Recensisce il materiale didattico;
    - Visualizza le lezioni;
    - Recensisce le lezioni;
    - Chat;
    - New chat;

- ***Docente***:
    - Login;
    - Visualizza aule;
    - Visualizza calendario;
    - Visualizza materiale didattico;
    - Visualizza le recensioni sul didattico;
    - Visualizza le recensioni sulle lezioni;
    - Visualizza le lezioni;
    - Chat;
    - New chat;

WEB APPs

- ***Segreteria***:
    - Login;
    - Nuovo calendario;
    - Elimina calendario;
    - Aggiunge lezione/esame;
    - Modifica aula lezione/esame;
    - Visualizza lezione/esame;
    - Crea aula;
    - Crea tool;
    - Aggiungi attrezzatura;
    - Elimina attrezzatura;
    - Modifica aula;
    - Elimina aula;
    - Aggiunge corso di studio;
    - Modifica corso di studio;
    - Aggiunge insegnamento;
    - Modifica insegnamento;
    - Aggiunge docente;
    - Modifica docente;
    - Gestisce segnalazione;
    - Elimina lezione/esame;

- ***Docente***:
    - Login;
    - Upload materiale didattico;
    - Crea nuova segnalazione;
    - Visualizza segnalazione;

SOTTO CASI D'USO

- **LOGIN**

Attori: Segretario, Docente, Studente

1. Il sistema fornisce all'utente l'interfaccia per inserire le proprie credenziali;
2. L'attore inserisce matricola e password;
3. Il sistema visualizza la pagina dopo aver effettuato il login in base al tipo di attore;

Estensione

2a. Il sistema invia un messaggio di "alert" perché la matricola e/o la password sono errate. L'attore deve inserire nuovamente le credenziali per poter accedere al sistema.
Si riparte dal punto 2.

CASI D'USO COMPLETI: APPs MOBILE

- **VISUALIZZA AULE**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule";
3. Il sistema presenta un elenco di tutte le aule disponibili;
4. L'attore seleziona l'aula dall'elenco o cerca l'aula per nome con la barra di ricerca disponibile nella pagina;
5. Il sistema carica i dettagli relativi all'aula selezionata con la possibilità di cliccare su "naviga";
6. L'attore selezione il bottone "naviga" e viene guidato verso l'aula attraverso l'utilizzo di Google Maps.

- **VISUALIZZA CALENDARIO**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendario";
3. Il sistema presenta un calendario del giorno suddiviso per ore con la lezione del giorno e la possibilità di selezione una nuova vista delle lezioni per giorno, mese o settimana;
4. L'attore seleziona la lezione del calendario;
5. Il sistema apre la pagina dei dettagli relativa alla lezione selezionata, con la mappa dell'aula e la possibilità di cliccare su "naviga".
6. L'attore selezione il bottone "naviga" e viene guidato verso l'aula attraverso l'utilizzo di Google Maps.

- **VISUALIZZA MATERIALE DIDATTICO**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "materiale didattico";
3. Il sistema presenta un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta un elenco di tutto il materiale didattico disponibile;
6. L'attore seleziona il materiale scelto;
7. Il sistema presenta la pagina del materiale didattico;

- **VISUALIZZA LEZIONI**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "lezioni";
3. Il sistema presenta un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta un elenco di tutte le lezioni svolte;

- **VISUALIZZA RECENSIONE MATERIALE DIDATTICO**

Attori: Docente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "recensioni materiale didattico";
3. Il sistema presenta un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta un elenco di tutto il materiale didattico disponibile;
6. L'attore seleziona il materiale scelto;
7. Il sistema presenta la pagina delle recensioni sul materiale didattico fornite dagli studenti in modo anonimo fornendo solo un voto ed una nota;

- **VISUALIZZA RECENSIONE LEZIONI**

Attori: Docente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "recensioni lezioni";
3. Il sistema presenta un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema fornisce un elenco di tutto le lezioni svolte;
6. L'attore seleziona la lezione interessata;
7. Il sistema presenta la pagina delle recensioni sulla lezione fornite dagli studenti in modo anonimo fornendo solo un voto ed una nota;

- **DOWLOAD MATERIALE DIDATTICO**

Attori: Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "materiale didattico";
3. Il sistema presenta un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta un elenco di tutto il materiale didattico disponibile;
6. L'attore seleziona il materiale scelto;
7. Il sistema presenta la pagina del materiale didattico con il bottone per il download;
8. L'attore selezione il bottone per il download;
9. Il sistema effettua il download del materiale didattico e ritorna al punto 7.

- **NEW CHAT**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "chat";
3. Il sistema presenta un elenco di tutte le chat disponibili che possono essere sia di tipo privata che di gruppo, in base all'insegnamenti e agli studenti del proprio corso di studio;
4. L'attore selezione seleziona il bottone per una nuova chat;
5. Il sistema presenta un elenco dei docenti e degli studenti del proprio corso di studio;
6. L'attore seleziona la persona con cui vuole iniziare una nuova chat;
7. Il sistema apre la room per la chat;

- **RECENSIONE MATERIALE DIDATTICO**

Attori: Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "materiale didattico";
3. Il sistema un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema fornisce un elenco di tutto il materiale didattico disponibile;
6. L'attore seleziona il materiale scelto;
7. Il sistema presenta la pagina del materiale didattico con il form per la recensione con voto e nota;
8. L'attore inserisce la nota e seleziona il voto per inviare la propria recensione cliccando su invia;
9. Il sistema apre un "alert" per chiedere conferma sull'invio della recensione;
10. L'attore seleziona conferma per inviare la propria recensione;
11. Il sistema presenta un "alert" di buona riuscita della recensione e ritorna al punto 6;

Estensione

8a. Il sistema invia un messaggio di "alert" perché il form della recensione nota o voto non sono stati riempiti.
Si riparte dal punto 7.

- **RECENSIONE LEZIONE**

Attori: Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "lezioni";
3. Il sistema un elenco di tutti gli insegnamenti del corso di studio dell'attore;
4. L'attore seleziona l'insegnamento dall'elenco o cerca per nome insegnamento con la barra di ricerca disponibile nella pagina;
5. Il sistema fornisce un elenco di tutte le lezioni svolte;
6. L'attore selezionala lezione;
7. Il sistema presenta la pagina della lezione con il form per la recensione con voto e nota;
8. L'attore inserisce la nota e seleziona il voto per inviare la propria recensione cliccando su invia;
9. Il sistema apre un "alert" per chiedere conferma sull'invio della recensione;
10. L'attore seleziona conferma per inviare la propria recensione;
11. Il sistema presenta un "alert" di buona riuscita della recensione e ritorna al punto 6;

Estensione

8a. Il sistema invia un messaggio di "alert" perché il form della recensione nota o voto non sono stati riempiti.
Si riparte dal punto 7.

- **CHAT**

Attori: Docente, Studente

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "chat";
3. Il sistema presenta un elenco di tutte le chat private già create e delle chat pubbliche disponibili, in base agli insegnamenti del proprio corso di studio;
4. L'attore selezione la chat dall'elenco;
5. Il sistema apre la room per la chat;

CASI D'USO COMPLETI: WEB APPs

- **NUOVO CALENDARIO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. L'attore seleziona il bottone "nuovo calendario";
5. Il sistema presenta il form di compilazione per il nuovo calendario;
6. L'attore compila il form con i dati richiesti e comferma;
7. Il sistema presenta un "alert" di calendario "creato con successo" e ritorna al punto 3.

- **ELIMINA CALENDARIO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. L'attore selezione il calendario o i calendari da voler eliminare e selezione il bottone "elimina";
5. Il sistema presenta un "alert" di successo calendario eliminato e ritorna al punto 3;

- **AGGIUNGE LEZIONE/ESAME**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. L'attore seleziona il calendario interessato;
5. Il sistema presenta il calendario scelto;
6. L'attore seleziona il giorno dal calendario cliccando su un punto vuoto della cella;
7. Il sistema presenta il form di compilazione della nuova lezione/esame;
8. L'attore compila il form e conferma la nuova lezione/esame;
9. Il sistema presenta un "alert" di conferma e ritorna al punto 5.

Estensione

8a. Il sistema invia un messaggio di "alert", perché gli orari inseriti della lezione/esame sono errati o la data selezionata non rientra nel semestre;
Si riparte dal punto 7.

8b. L'attore seleziona la spunta "inserisci per tutte le settimane del calendario";
8b.1 Il sistema aggiunge la lezione/esame per tutte le settimane del semestre;
Si riparte dal punto 9.

- **MODIFICA AULA LEZIONE/ESAME**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. L'attore seleziona il calendario interessato;
5. Il sistema presenta il calendario scelto;
6. L'attore seleziona la lezione/esame dal calendario da voler modificare;
7. Il sistema presenta i dettagli della lezione/esame attraverso un "alert" per continuare o annullare;
8. L'attore seleziona "modifica";
9. Il sistema presenta il form con i relativi dati;
10. L'attore modifica l'aula e conferma;
11. Il sistema presenta un "alert" di conferma e ritorna al punto 5.

- **ELIMINA LEZIONE/ESAME**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. L'attoreseleziona il calendario interessato;
5. Il sistema presenta il calendario scelto;
6. L'attore seleziona la lezione/esame dal calendario da voler modificare;
7. Il sistema presenta i dettagli della lezione/esame attraverso un "alert" per continuare o annullare;
8. L'attore seleziona "modifica";
9. Il sistema presenta il form con i relativi dati;
10. L'attore elimina la lezione/esame scelto;
11. Il sistema presenta un "alert" di conferma con lezione/esame eliminato e ritorna al punto 5.

- **VISUALIZZA LEZIONE/ESAME**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "calendari" utilizzando il menù;
3. Il sistema presenta la lista dei calendari disponibili;
4. Lattore'seleziona il calendario interessato;
5. Il sistema presenta il calendario scelto;
6. L'attore seleziona la lezione/esame dal calendario da voler visualizzare;
7. Il sistema presenta un "pop-up" con i dettagli della lezione/esame;

- **CREA AULA**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona il bottone "nuova aula";
5. Il sistema presenta il form da compilare per la nuova aula;
6. L'attore compila il form e conferma la nuova aula creata;
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **CREA TOOL**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona il bottone "nuovo tool";
5. Il sistema presenta il form da compilare per il nuovo tool;
6. L'attore compila il form e conferma;
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **MODIFICA AULA**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona l'aula dall'elenco o cerca l'aula per nome con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta il form con irelativi dati dell'aula da modificare;
6. L'attore modifica i dati dell'aula compilando il form e seleziona il bottore "aggiona";
7. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **AGGIUNGI ATTREZZATURE**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona l'aula dall'elenco o cerca l'aula per nome con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta il form dell'aula e della sua attrezzatura;
6. L'attore seleziona il bottone "scegli tool";
7. Il sistema presenta un "pop-up" con i tools disponibili da aggiungere;
8. L'attore seleziona il tool interessato;
9. Il sistema ritorna al punto 5;
10. L'attore clicca "aggiungi";
11. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **ELIMINA ATTREZZATURE**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona l'aula dall'elenco o cerca l'aula per nome con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta il form dell'aula e della sua attrezzatura;
6. L'attore seleziona l'attrezzatura da rimuovere dall'aula e clicca su "rimuovi";
7. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **ELIMINA AULA**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "aule" utilizzando il menù;
3. Il sistema presenta la lista delle aule e dei tools;
4. L'attore seleziona l'aula o le aula dall'elenco e clicca "elimina";
5. Il sistema presenta un "alert" di successo e ritorna al punto 2.

Estensione

4a. Il sistema invia un messaggio di "alert" con un messaggio di avviso prima della conferma.

4a.1 L'attore conferma l'eliminazione dell'aula;

Si riparte dal punto 9.

- **AGGIUNGI CORSO DI STUDIO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "corsi di studio" utilizzando il menù;
3. Il sistema presenta una lista di tutti i corsi di studio disponibili;
4. L'attore clicca su "nuovo corso";
5. Il sistema presenta un form con i relativi dati del corso;
6. L'attore compila il form con il nuovo corso conferma il nuovo corso;
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **MODIFICA CORSO DI STUDIO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "corsi di studio" utilizzando il menù;
3. Il sistema presenta una lista di tutti i corsi di studio disponibili;
4. L'attore seleziona il corso di studio interessato;
5. Il sistema presenta un form con i relativi dati del corso da modifcare;
6. L'attore modifica i campi del form interessati e clicca su "invio";
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **AGGIUNGI INSEGNAMENTO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "corsi di studio" utilizzando il menù;
3. Il sistema presenta una lista di tutti i corsi di studio disponibili;
4. L'attore seleziona il corso di studio interessato;
5. Il sistema presenta un form con i relativi dati del corso e la lista degli insegnamenti del corso;
6. L'attore seleziona "nuovo insegnamento";
7. Il sistema presenta un form da compilare con i dati del nuovo insegnamento;
8. L'attore clicca su "invio";
9. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **MODIFICA INSEGNAMENTO**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "corsi di studio" utilizzando il menù;
3. Il sistema presenta una lista di tutti i corsi di studio disponibili;
4. L'attore seleziona il corso di studio interessato;
5. Il sistema presenta un form con i relativi dati del corso e la lista degli insegnamenti;
6. L'attore seleziona l'insegnamento interessato;
7. Il sistema presenta un form con i relativi dati dell'insegnamento da modifcare;
8. L'attore modifica i campi del form interessati e clicca su "invio";
9. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **AGGIUNGI DOCENTE**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "docenti" utilizzando il menù;
3. Il sistema presenta una lista di tutti i docenti disponibili;
4. L'attore clicca su "nuovo docente";
5. Il sistema presenta un form con i relativi campi del nuovo docente;
6. L'attore compila il form e clicca "invio";
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **MODIFICA DOCENTE**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "docenti" utilizzando il menù;
3. Il sistema presenta una lista di tutti i docenti disponibili;
4. L'attore seleziona il docente dall'elenco o cerca il docente per nome e cognome con la barra di ricerca disponibile nella pagina;
5. Il sistema presenta un form con i relativi dati del docente da modificare;
6. L'attore modifica i campi del form interessati e clicca su "invio";
7. Il sistema presenta un "alert" di successo e ritorna al punto 5.

- **GESTIONE SEGNALAZIONE**

Attori: Segreteria

1. L'attore effettua il LOGIN;
2. L'attore accede alla pagina "segnalazioni" utilizzando il menù;
3. Il sistema presenta una lista di tutte le segnalazioni prese in gestione;
4. L'attore seleziona la segnalazione interessata;
5. Il sistema mostra i dettagli relativi alla segnalazione scelta;
6. L'attore gestisce la segnalazione, aggiornando lo stato con le opzioni di:
   -in lavorazione
   -risolta
   -presa in carico
   -rifiutata.
   Aggiungendo se necessario una nota alla segnalazione e clicca su "invio";
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

Estensione

6a. Il sistema invia un "alert" di errore con un messaggio dove è necessario specificare con una nota nel caso lo stato della segnalazione è stato impostato su "rifiutata".
Si riparte dal punto 5.

- **UPLOAD MATERIALE DIDATTICO**

Attori: Docente

1. L'attore effettua il LOGIN;
2. L'attore seleziona l'insegnamento interessato;
3. Il sistema presenta la lista del materiale didattico relativo all'insegnamento;
4. L'attore clicca su "nuovo materiale";
5. Il sistema presente la pagina per caricare il nuovo materiale;
6. L'attore seleziona "choose file" e seleziona il file da caricare;
7. Il sistema presenta il form dei dati relativi al materiale aggiunto con una barra di caricamento relativa all'upload del file;
8. L'attore clicca su "invio" al termine dell'upload;
9. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **CREA NUOVA SEGNALAZIONE**

Attori: Docente

1. L'attore effettua il LOGIN;
2. L'attore seleziona il bottone visualizza segnalazioni;
3. Il sistema presenta una lista delle segnalazioni;
4. L'attore clicca su "nuova segnalazione";
5. Il sistema presenta il form con i campi della segnalazione da inviare;
6. L'attore compila il form e clicca su "invio" per inviare la nuova segnalazione alla segreteria;
7. Il sistema presenta un "alert" di successo e ritorna al punto 3.

- **VISUALIZZA SEGNALAZIONE**

Attori: Docente

1. L'attore effettua il LOGIN;
2. L'attore seleziona il bottone visualizza segnalazioni;
3. Il sistema presenta una lista delle segnalazioni;
4. L'attore seleziona la segnalazione interessata;
5. Il sistema presenta il form con i dati relativi alla segnalazione;

Estensione

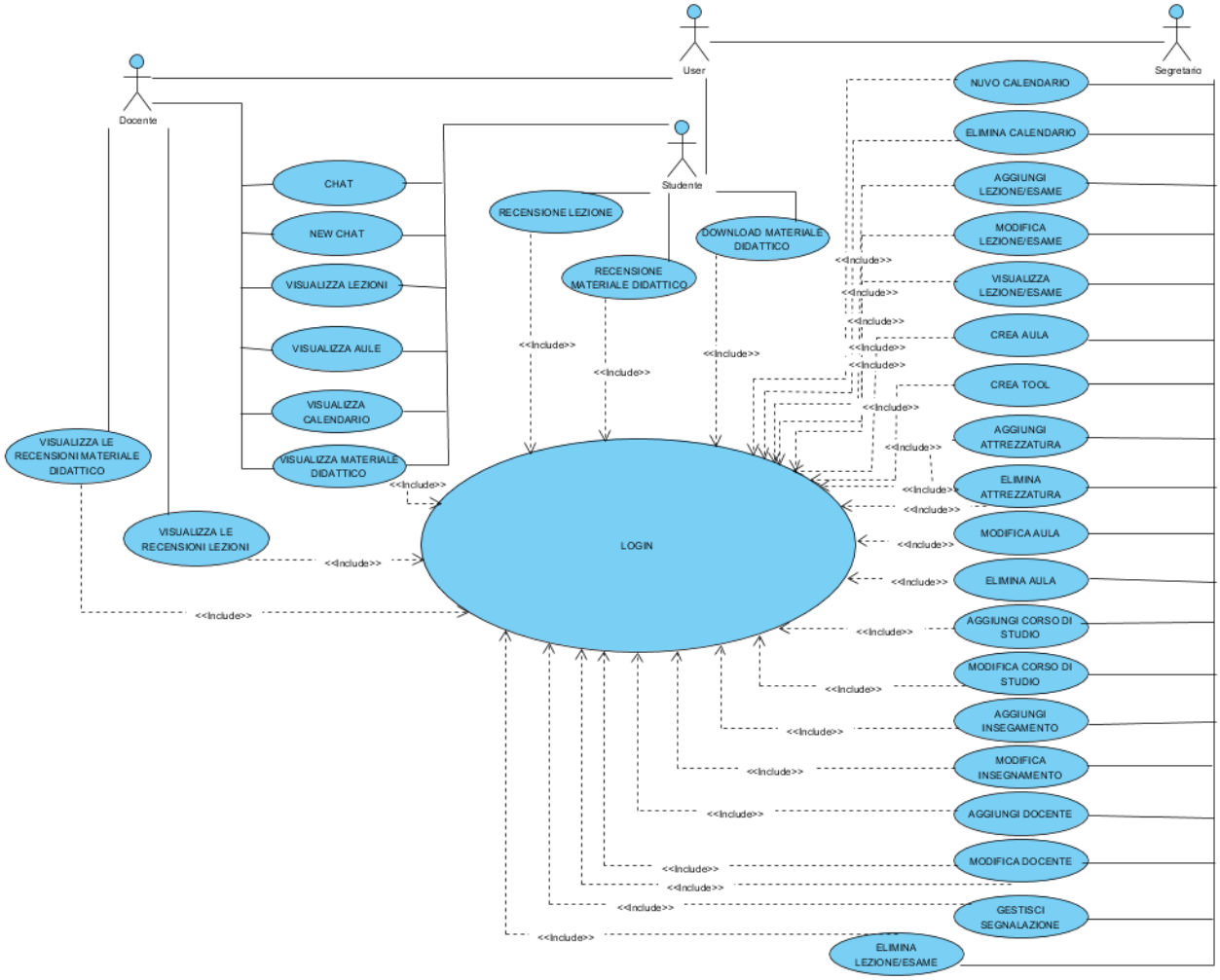
5a. L'attore selezione "vedi segnalazioni aule";

5a.1 Il sistema presenta una lista delle segnalazioni riguardanti la stessa aula;

5a.2 L'attore seleziona la segnalazione interessata;

5a.3 Il sistema presenta i dettagli della segnalazione selezionata;

Diagramma dei Casi d'Uso



Architettura del Sistema

Descrizione del sistema

Il sistema realizzato si basa sul paradigma Client-Server, ed è implementato secondo un'architettura Three-tier. Risulta essere composto da:

- **Web Server;**
- **Web App;**
- **Mobile App;**
- **Database relazionale;**

Il Web Server costituisce il lato Back-End del sistema, il quale elabora le richieste e i dati provenienti dalle App, permettendo il corretto funzionamento delle interazioni con l'opportuno attore. Si basa sull'architettura REST, in modo da massimizzare la flessibilità del server e da garantire compatibilità con le future tecnologie. Il Web Server quindi espone dei Web Service di tipo REST che sono identificati da un URI, e sono necessari al funzionamento di tutte le applicazioni Front-End realizzate. Le web responses dei vari Web Services esposti dal server sono codificate utilizzando il formato JSON attraverso l'utilizzo dei DTO.
Il Web Server svolge il ruolo di ponte tra il client e il database, in quanto dovrà prendere in carico le richieste dei client restituendo le opportune informazioni richieste in base a quanto contenuto nel database.

La Web App per il Docente e la Segreteria con le due mobile apps, una per lo Studente e una per Docente, costituiscono il Front-End del sistema. Permettono al client di interagire con il server, per salvare e/o ottenere dati dal database.

È possibile sintetizzare il flusso dei dati all'interno del sistema nel seguente modo:

- Il client richiede l'accesso ad un contenuto tramite Web App o Mobile Apps;
- Si invia una richiesta verso uno o più Web Services esposti dal Web Server per ottenere i dati necessari;
- Il Web Server preleva dal database le informazioni richieste e le codifica in oggetti JSON con i DTO prima di rispondere alla richiesta;
- Il contenuto è inviato all'opportuna App usata dal client.

Design Pattern

Per evitare la realizzazione di un sistema funzionante ma incomprensibile e non propenso ad aggiornamenti e lavori di manutenzione, si è deciso di utilizzare il concetto dei Design Pattern con l'opportuna implementazione, sia dal lato Front-End che dal lato Back-end. In questo modo si è realizzato un sistema chiaro e flessibile, andando ad usare ed implementare semplicemente delle soluzioni progettuali generali a problemi ricorrenti.
Ne distinguiamo 3 famiglie, che sono rispettivamente:

- **PATTERN CREAZIONALI**: riguardano il processo di crea- zione degli oggetti;
- **PATTERN STRUTTURALI**: riguardano la composizione di classi e di oggetti;
- **PATTERN COMPORTAMENTALI**: si occupano di come oggetti interagiscono reciprocamente e di come vengono distribuite fra loro le responsabilità;

Web Server

Per la realizzazione del Web Server sono stati usati i seguenti Design Patterns:

- **DAO**: acronimo di Data Access Object, è un pattern architetturale che ha consentito di accedere ai dati presenti nel database e di mapparli sulle corrispondenti classi di model in Java, disaccoppiando i due concetti. Tale pattern si compone di 3 partecipanti: Data Access Object Interface, Data Access Object Concrete Class, Model Object. Essi, nel nostro caso, sono rappresentati rispettivamente dalle classi DAO Interfaces, DAO Implementation e Domain.

- **DTO**: acronimo di Data Transfer Object, è un pattern che viene usato quando vogliamo passare dati con più attributi in un unico blocco da Client al Server.
  I Transfers Objects non sono altro che semplici POJO Class, con metodi get/set che vengono serializzati in modo che possano essere trasferiti sulla rete. Essi, nel nostro caso, sono stati utilizzati nell'inviare i dati del nostro DB, settati in un determinato modo, alle nostre rispettive App e Web attraverso i servizi Rest.

- **ITERATOR**: è un pattern comportamentale che permette di analizzare gli elementi di una collezione di oggetti indipendentemente dalla struttura dati usata. Nei servizi REST, precisamente nella maggior parte dei metodi GET, sono state create delle liste in cui contenere diversi oggetti della stessa classe estrapolate da una determinata query fatta al database. L'uso di tale pattern ha permesso di evitare l'utilizzo di strutture cicliche come i for, per poi estrapolare i dati fornendo la posizione nella lista.

- **COMPOSITE**: un pattern strutturale basato su oggetti che viene utilizzato quando si ha la necessità di realizzare una gerarchia di oggetti in cui l'oggetto contenitore può detenere oggetti elementari e/o oggetti contenitori. L'obiettivo è di permettere al Client che deve navigare la gerarchia, di comportarsi sempre nello stesso modo sia verso gli oggetti elementari e sia verso gli oggetti contenitori. Tale pattern è stato utilizzato per la realizzazione del calendario per poter avere una gestione delle lezioni e degli esami in esso contenute.

- **STRATEGY**: Si tratta di un pattern comportamentale basato su oggetti e viene utilizzato per definire una famiglia di algoritmi, incapsularli e renderli intercambiabili. Il client definisce l'algoritmo da utilizzare, incapsulandolo in un contesto, il quale verrà utilizzato nella fase di elaborazione. Il contesto detiene i puntamenti alle informazioni necessarie al fine della elaborazione, e nel nostro caso il pattern Strategy è stato utilizzato per implementare una famiglia di algoritmi di ordinamento dati.

- **FACTORY METHOD**: Si tratta di un pattern basato su classi e viene utilizzato per creare degli oggetti senza conoscerne i dettagli ma delegando un Creator che, in base alle informazioni ricevute, saprà quale oggetto restituire. Questo pattern consente di separare il Client dal Framework permettendo di modificare i dettagli implementativi senza dovere modificare il Client.

Web App e Mobile Apps

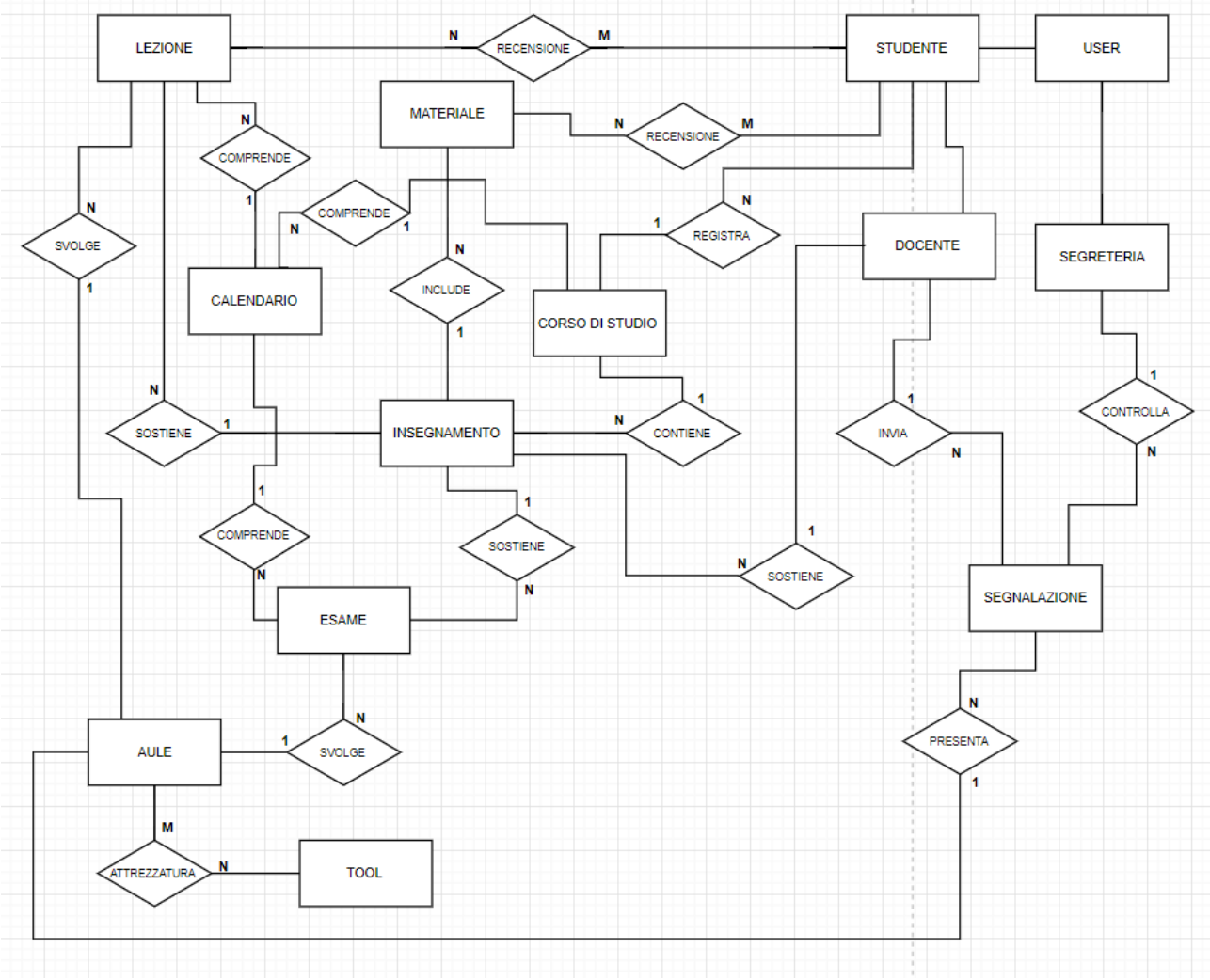
- **MVC (Model View Controller):** le applicazioni front-end sono state sviluppane con framework (Angular2 e Ionic2) che hanno alla base il pattern architetturale MVC.
  Questo permette loro di separare la logica di presentazione (View) dalla logica di business (Controller) e dalle strutture dati utilizzate (Model).

- **OBSERVER**: tutte le applicazioni fanno uso di questo pattern comportamentale, ogni qual volta si effettua una chiamata asincrona verso uno dei Web Services del nostro sistema, o esterno. Quindi, quando viene effettuata una chiamata HTTP il controller che la effettua esegue una sottoscrizione (.subscribe()) al metodo, in modo che una volta arrivata la risposta, gli sia notificato il suo contenuto.

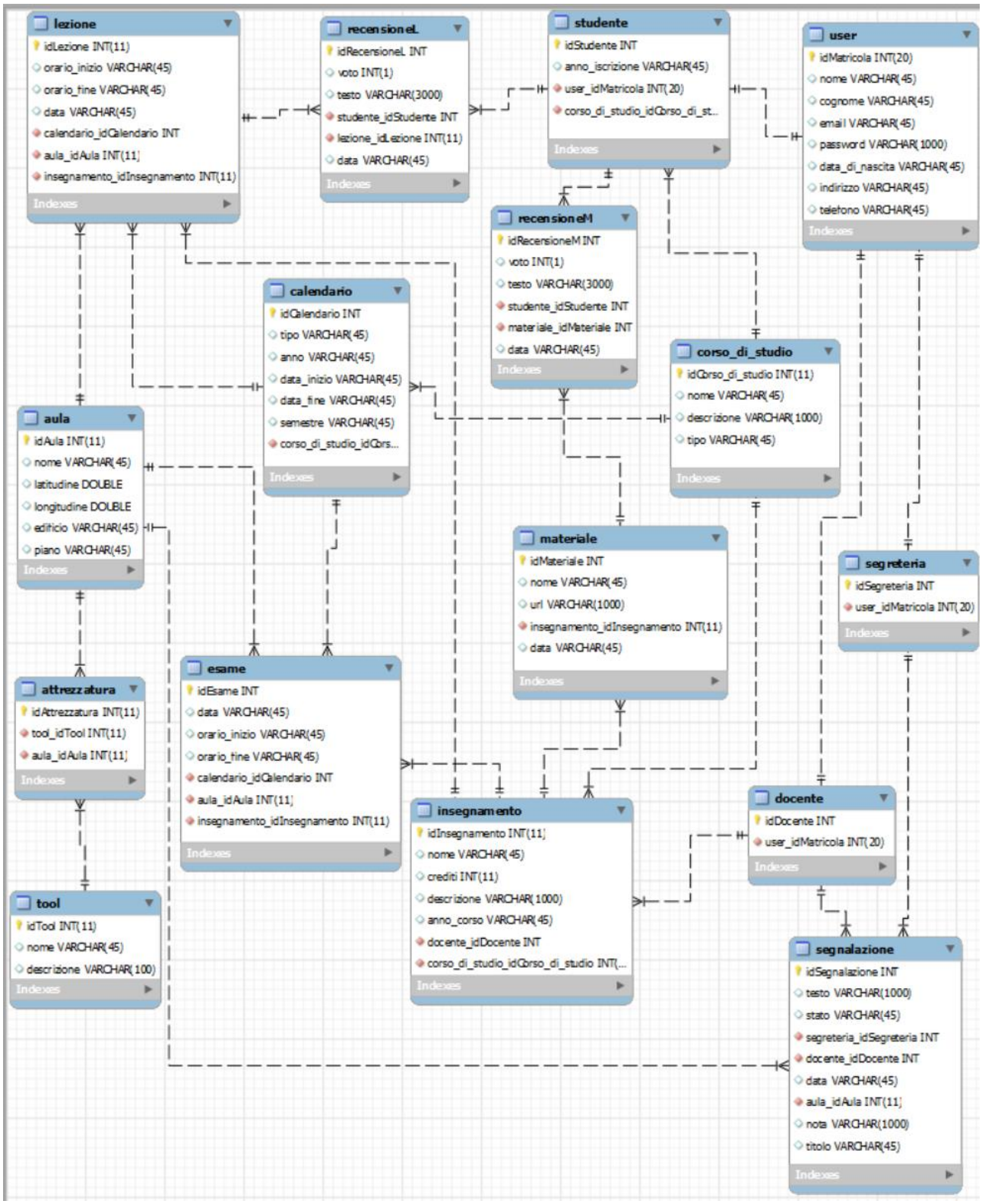
Descrizione Base Dati

Modello E-R

La gestione dei dati è permessa tramite l'utilizzo di un database di tipo relazionale, dove si usa MySQL come RDBMS. Il Web Server può interagire con il database per la memorizzazione dei dati attraverso l'integrazione di Hibernate-Tool in un progetto Spring-Gradle. Esso è un tool ORM (Object-Relational Mapping) che permette di gestire la persistenza dei dati sul database, e di effettuare un mapping fra classi Java di model e dati presenti sulle tabelle del database, attraverso la generazione delle classi "domain".
Il progetto software presenta il seguente **Modello E-ER**:

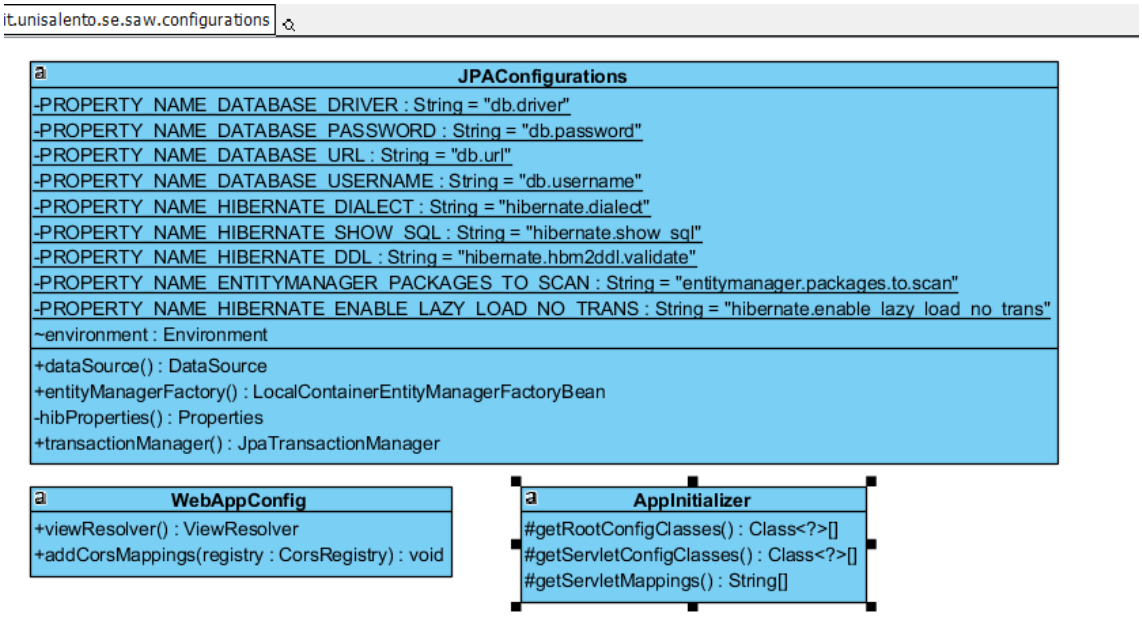


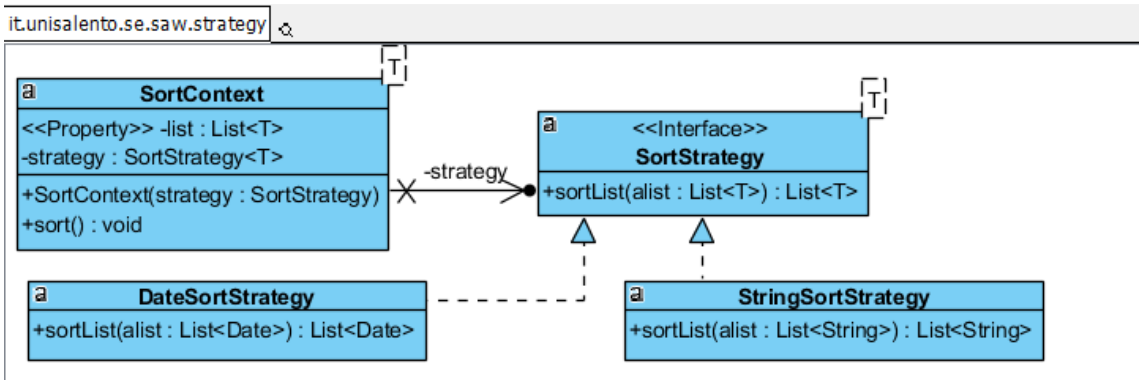
Modello Logico:

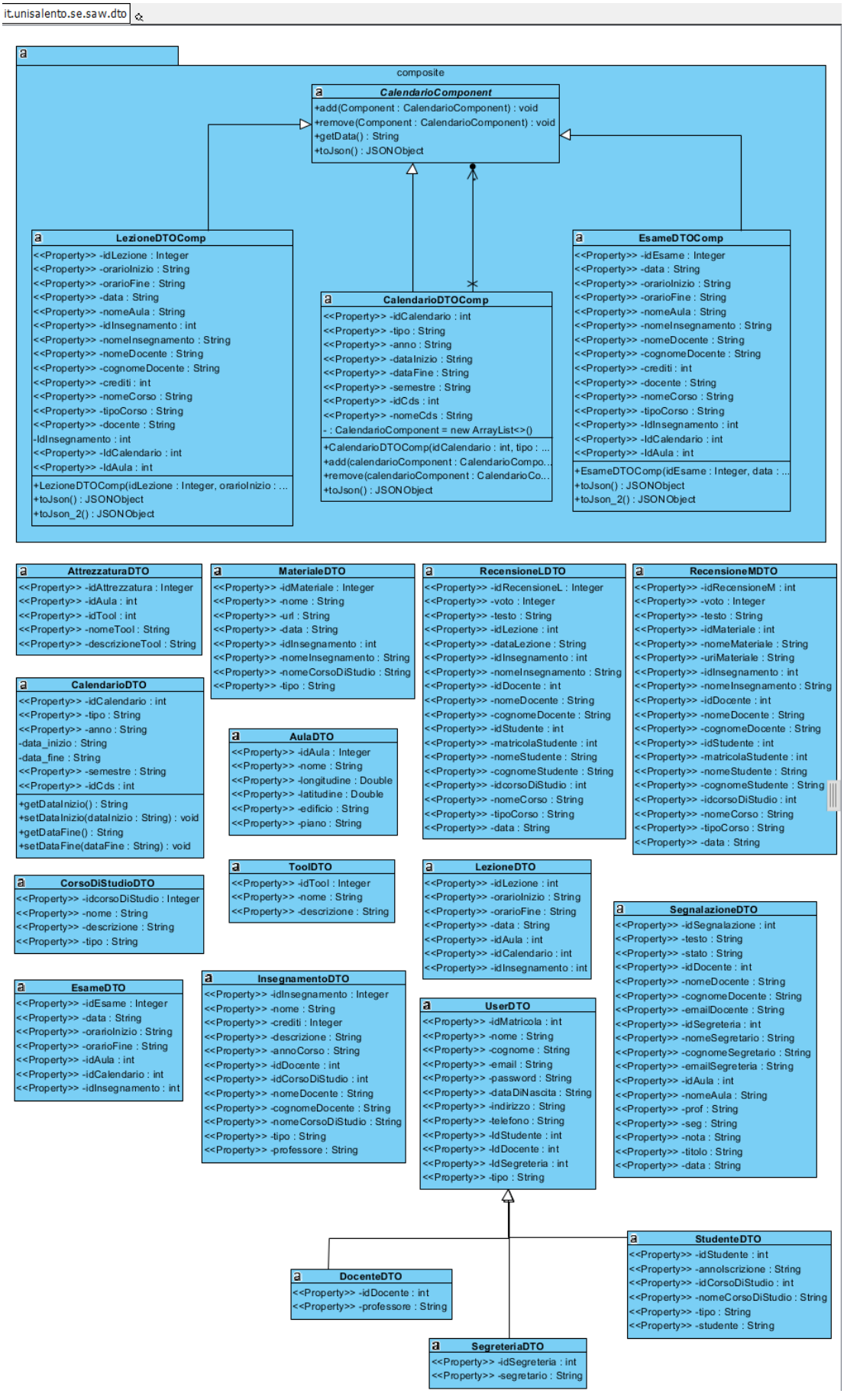


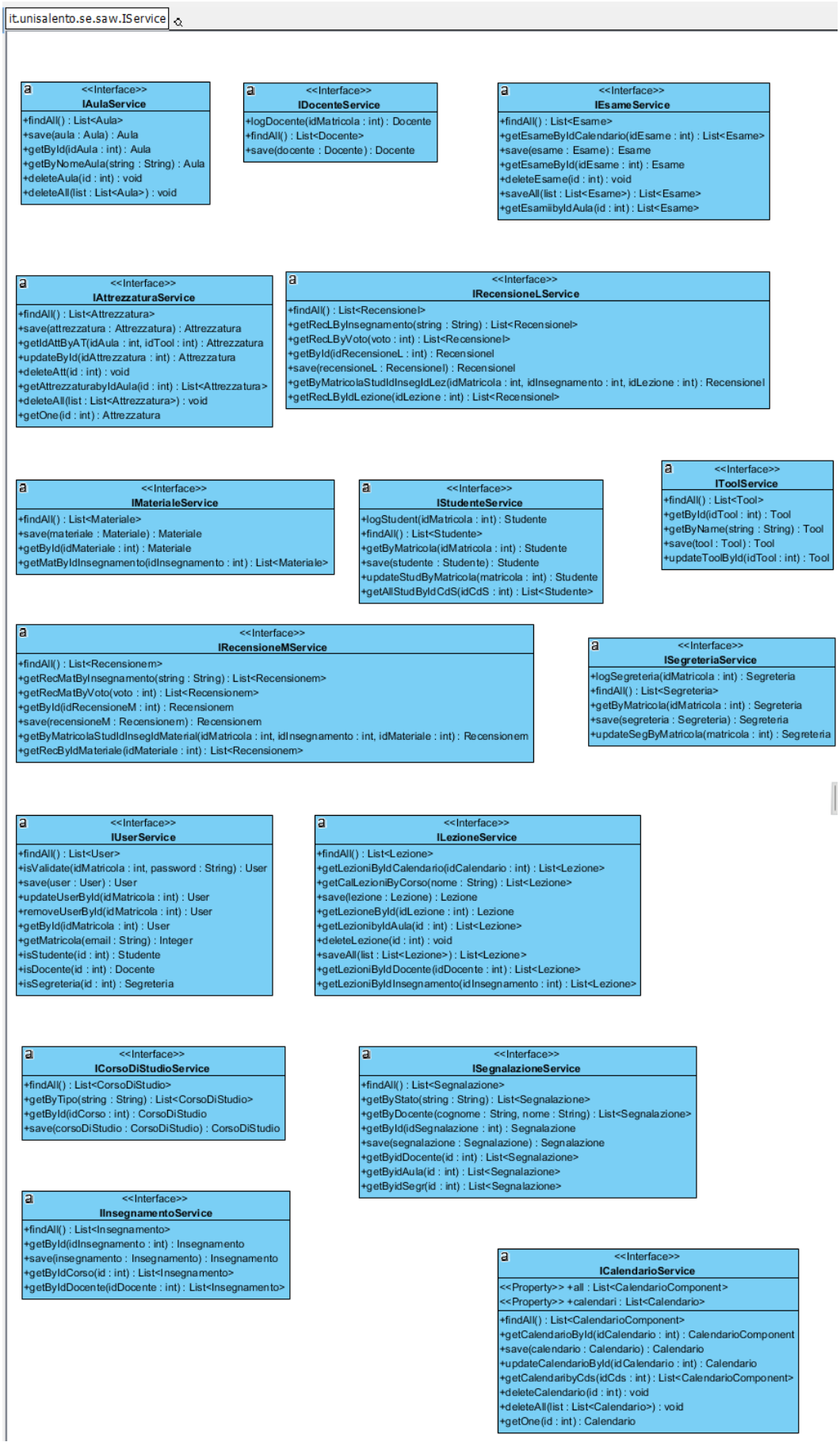
Diagrammi delle Classi

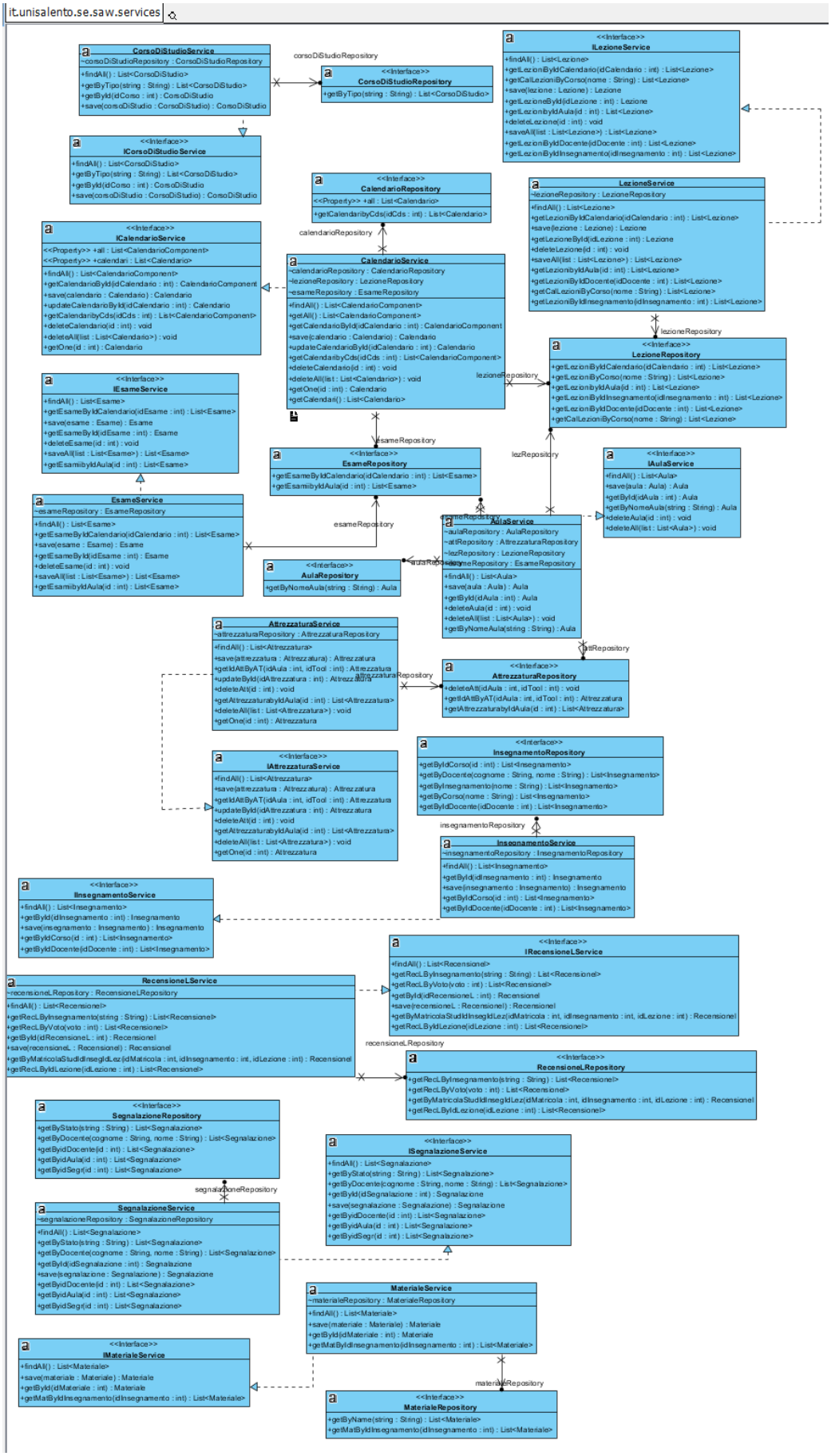
Poiché ci si è resi conto della eccessiva complessità di un diagramma delle classi generale si è preferito creare diagrammi delle classi che si riferiscono solo alle classi relative ad un determinato aspetto del programma. In seguito si sono comunque inseriti i diagrammi di tutte le classi, raggruppati per package, privati di dipendenze e associazioni.

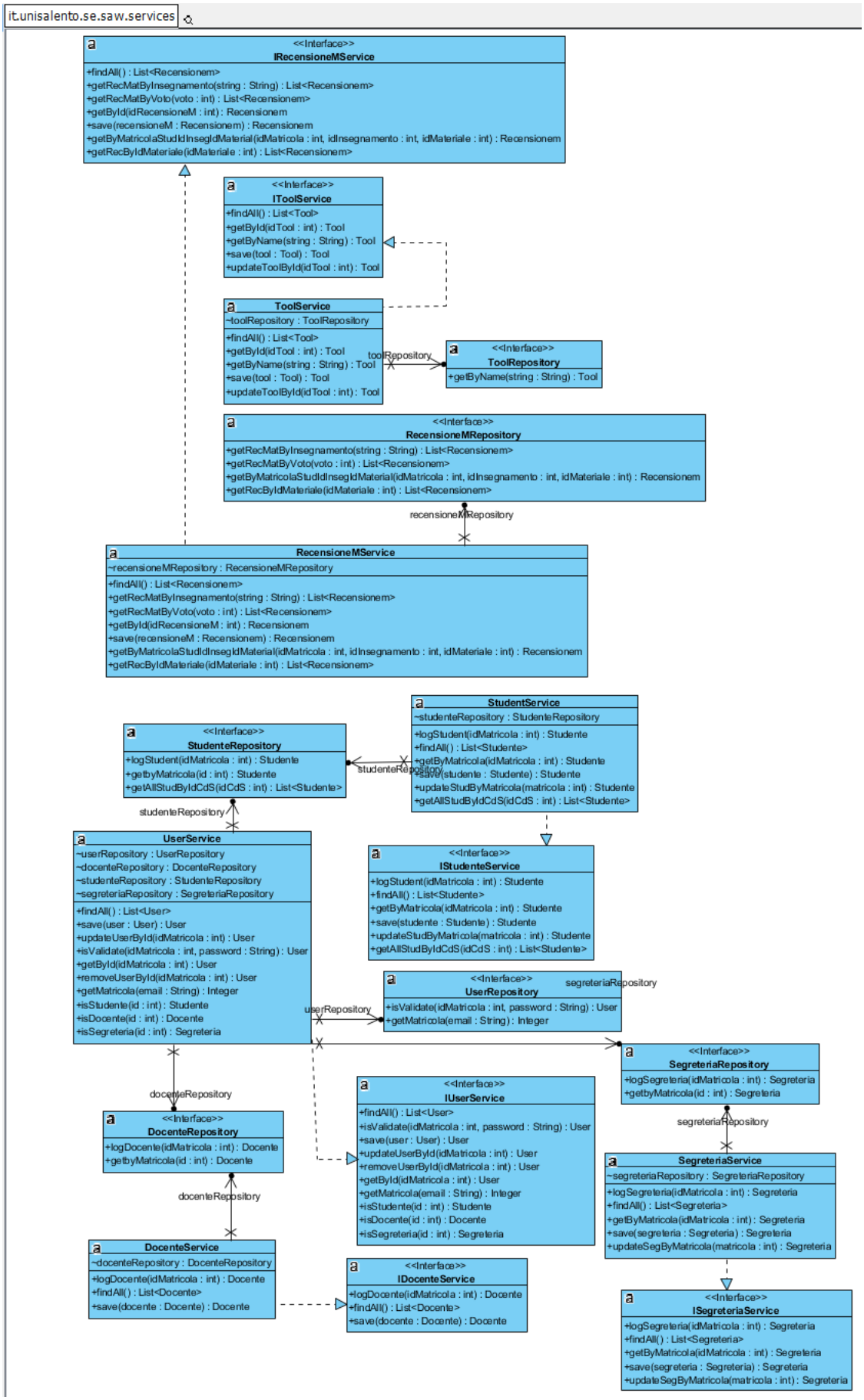


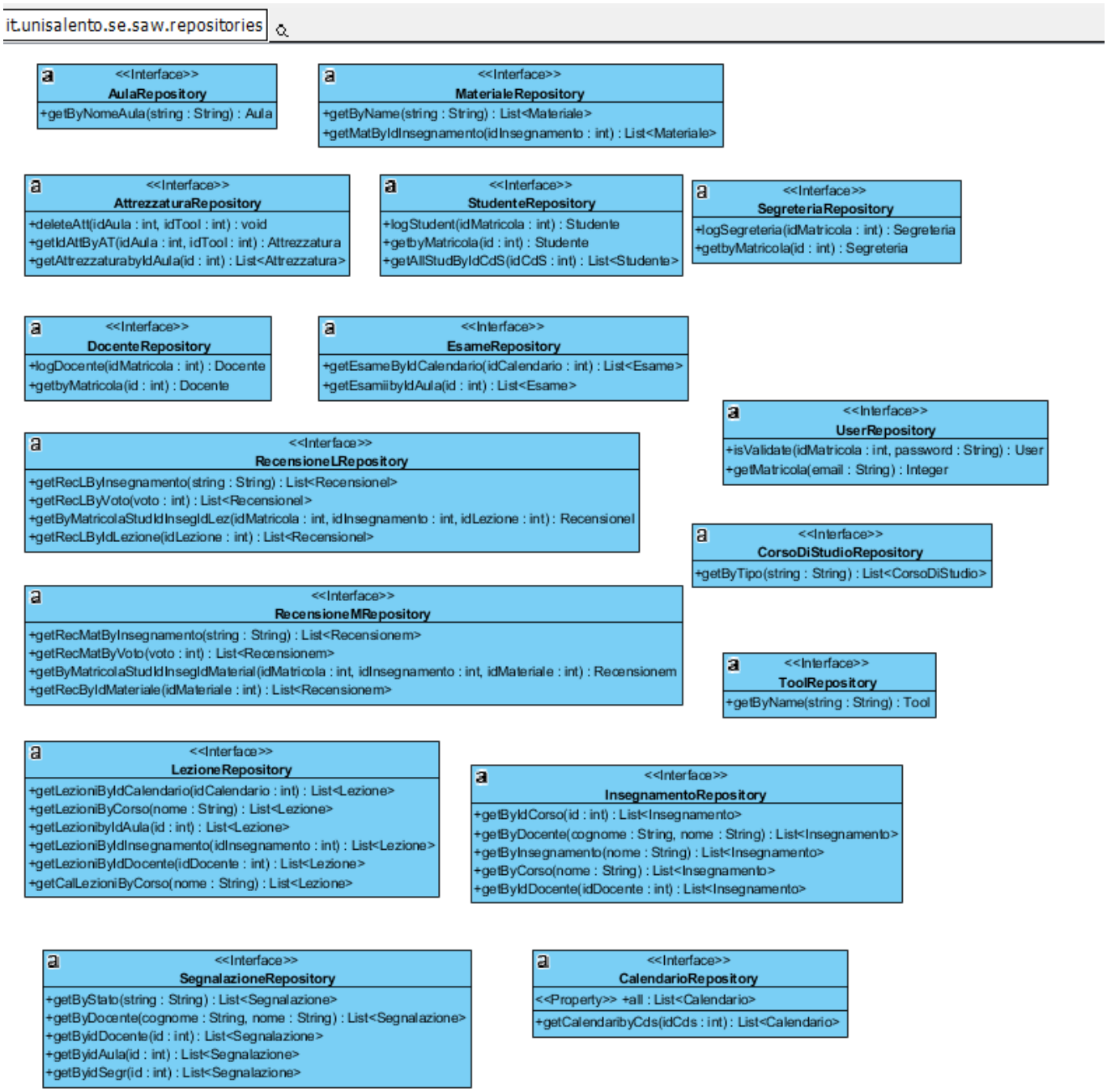


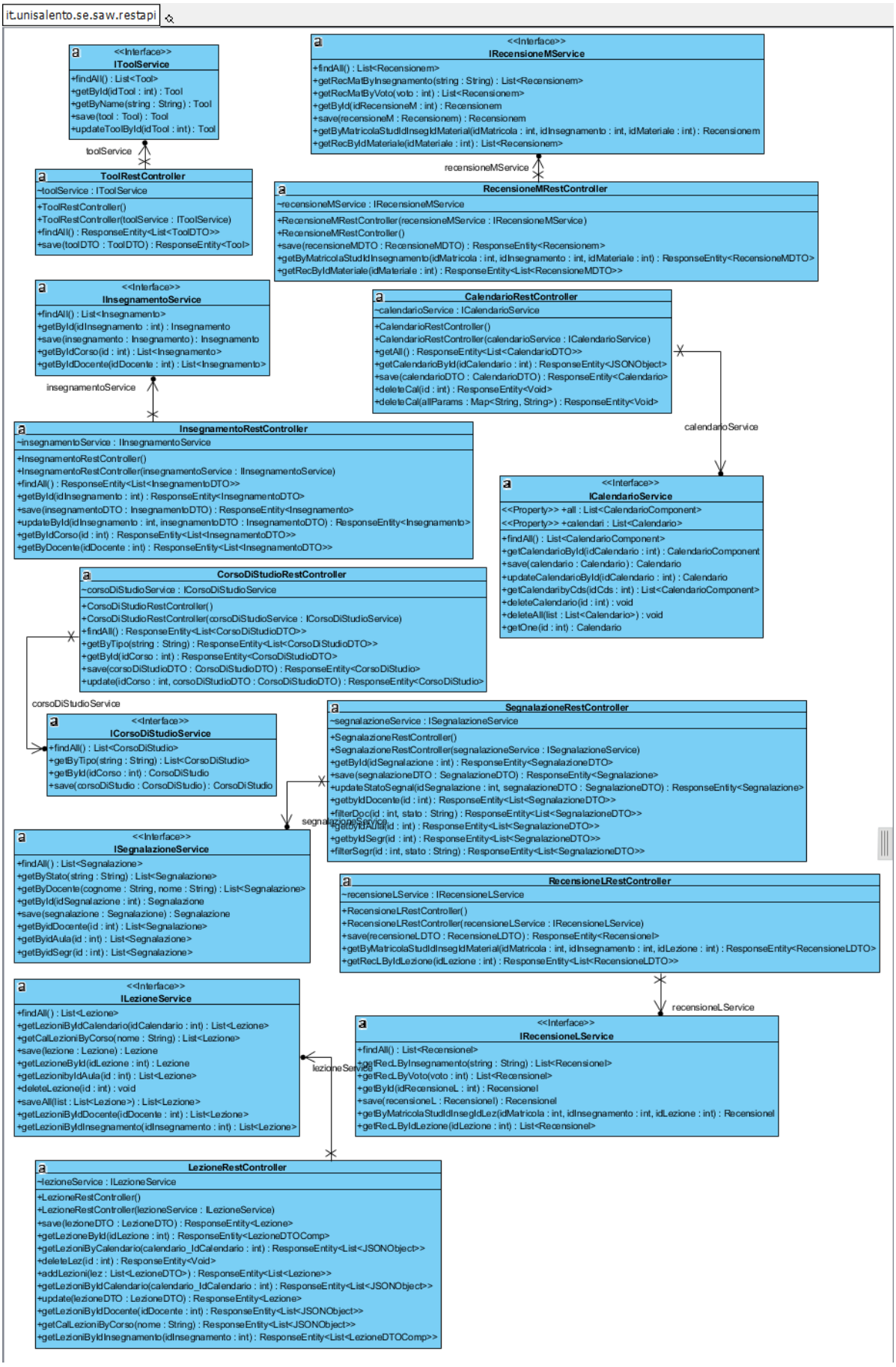

it.unisalento.se.saw.dto
composite
CalendarioComponent
+add(Component : CalendarioComponent) : void
+remove(Component : CalendarioComponent) : void
+getData() : String
+toJson() : JSONObject
LezioneDTOComp
<<Property>> -idLezione : Integer
<<Property>> -orarioInizio : String
<<Property>> -orarioFine : String
<<Property>> -data : String
<<Property>> -nomeAula : String
<<Property>> -idInsegnamento : int
<<Property>> -nomeInsegnamento : String
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -crediti : int
<<Property>> -nomeCorso : String
<<Property>> -tipoCorso : String
<<Property>> -docente : String
-IdInsegnamento : int
<<Property>> -IdCalendario : int
<<Property>> -IdAula : int
+LezioneDTOComp(idLezione : Integer, orarioInizio : ...
+toJson() : JSONObject
+toJson_2() : JSONObject
CalendarioDTOComp
<<Property>> -idCalendario : int
<<Property>> -tipo : String
<<Property>> -anno : String
<<Property>> -dataInizio : String
<<Property>> -dataFine : String
<<Property>> -semestre : String
<<Property>> -idCds : int
<<Property>> -nomeCds : String
- : CalendarioComponent = new ArrayList<>()
+CalendarioDTOComp(idCalendario : int, tipo : ...
+add(calendarioComponent : CalendarioCompo...
+remove(calendarioComponent : CalendarioCo...
+toJson() : JSONObject
EsameDTOComp
<<Property>> -idEsame : Integer
<<Property>> -data : String
<<Property>> -orarioInizio : String
<<Property>> -orarioFine : String
<<Property>> -nomeAula : String
<<Property>> -nomeInsegnamento : String
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -crediti : int
<<Property>> -docente : String
<<Property>> -nomeCorso : String
<<Property>> -tipoCorso : String
<<Property>> -IdInsegnamento : int
<<Property>> -IdCalendario : int
<<Property>> -IdAula : int
+EsameDTOComp(idEsame : Integer, data : ...
+toJson() : JSONObject
+toJson_2() : JSONObject
AttrezzaturaDTO
<<Property>> -idAttrezzatura : Integer
<<Property>> -idAula : int
<<Property>> -idTool : int
<<Property>> -nomeTool : String
<<Property>> -descrizioneTool : String
MaterialeDTO
<<Property>> -idMateriale : Integer
<<Property>> -nome : String
<<Property>> -url : String
<<Property>> -data : String
<<Property>> -idInsegnamento : int
<<Property>> -nomeInsegnamento : String
<<Property>> -nomeCorsoDiStudio : String
<<Property>> -tipo : String
RecensioneLDTO
<<Property>> -idRecensioneL : Integer
<<Property>> -voto : Integer
<<Property>> -testo : String
<<Property>> -idLezione : int
<<Property>> -dataLezione : String
<<Property>> -idInsegnamento : int
<<Property>> -nomeInsegnamento : String
<<Property>> -idDocente : int
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -idStudente : int
<<Property>> -matricolaStudente : int
<<Property>> -nomeStudente : String
<<Property>> -cognomeStudente : String
<<Property>> -idcorsoDiStudio : int
<<Property>> -nomeCorso : String
<<Property>> -tipoCorso : String
<<Property>> -data : String
RecensioneMDTO
<<Property>> -idRecensioneM : int
<<Property>> -voto : Integer
<<Property>> -testo : String
<<Property>> -idMateriale : int
<<Property>> -nomeMateriale : String
<<Property>> -uriMateriale : String
<<Property>> -idInsegnamento : int
<<Property>> -nomeInsegnamento : String
<<Property>> -idDocente : int
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -idStudente : int
<<Property>> -matricolaStudente : int
<<Property>> -nomeStudente : String
<<Property>> -cognomeStudente : String
<<Property>> -idcorsoDiStudio : int
<<Property>> -nomeCorso : String
<<Property>> -tipoCorso : String
<<Property>> -data : String
CalendarioDTO
<<Property>> -idCalendario : int
<<Property>> -tipo : String
<<Property>> -anno : String
-data_inizio : String
-data_fine : String
<<Property>> -semestre : String
<<Property>> -idCds : int
+getDataInizio() : String
+setDataInizio(dataInizio : String) : void
+getDataFine() : String
+setDataFine(dataFine : String) : void
AulaDTO
<<Property>> -idAula : Integer
<<Property>> -nome : String
<<Property>> -longitudine : Double
<<Property>> -latitudine : Double
<<Property>> -edificio : String
<<Property>> -piano : String
ToolDTO
<<Property>> -idTool : Integer
<<Property>> -nome : String
<<Property>> -descrizione : String
LezioneDTO
<<Property>> -idLezione : int
<<Property>> -orarioInizio : String
<<Property>> -orarioFine : String
<<Property>> -data : String
<<Property>> -idAula : int
<<Property>> -idCalendario : int
<<Property>> -idInsegnamento : int
SegnalazioneDTO
<<Property>> -idSegnalazione : int
<<Property>> -testo : String
<<Property>> -stato : String
<<Property>> -idDocente : int
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -emailDocente : String
<<Property>> -idSegreteria : int
<<Property>> -nomeSegretario : String
<<Property>> -cognomeSegretario : String
<<Property>> -emailSegreteria : String
<<Property>> -idAula : int
<<Property>> -nomeAula : String
<<Property>> -prof : String
<<Property>> -seg : String
<<Property>> -nota : String
<<Property>> -titolo : String
<<Property>> -data : String
CorsoDiStudioDTO
<<Property>> -idcorsoDiStudio : Integer
<<Property>> -nome : String
<<Property>> -descrizione : String
<<Property>> -tipo : String
EsameDTO
<<Property>> -idEsame : Integer
<<Property>> -data : String
<<Property>> -orarioInizio : String
<<Property>> -orarioFine : String
<<Property>> -idAula : int
<<Property>> -idCalendario : int
<<Property>> -idInsegnamento : int
InsegnamentoDTO
<<Property>> -idInsegnamento : Integer
<<Property>> -nome : String
<<Property>> -crediti : Integer
<<Property>> -descrizione : String
<<Property>> -annoCorso : String
<<Property>> -idDocente : int
<<Property>> -idCorsoDiStudio : int
<<Property>> -nomeDocente : String
<<Property>> -cognomeDocente : String
<<Property>> -nomeCorsoDiStudio : String
<<Property>> -tipo : String
<<Property>> -professore : String
UserDTO
<<Property>> -idMatricola : int
<<Property>> -nome : String
<<Property>> -cognome : String
<<Property>> -email : String
<<Property>> -password : String
<<Property>> -dataDiNascita : String
<<Property>> -indirizzo : String
<<Property>> -telefono : String
<<Property>> -IdStudente : int
<<Property>> -IdDocente : int
<<Property>> -IdSegreteria : int
<<Property>> -tipo : String
StudenteDTO
<<Property>> -idStudente : int
<<Property>> -annoIscrizione : String
<<Property>> -idCorsoDiStudio : int
<<Property>> -nomeCorsoDiStudio : String
<<Property>> -tipo : String
<<Property>> -studente : String
DocenteDTO
<<Property>> -idDocente : int
<<Property>> -professore : String
SegreteriaDTO
<<Property>> -idSegreteria : int
<<Property>> -segretario : String

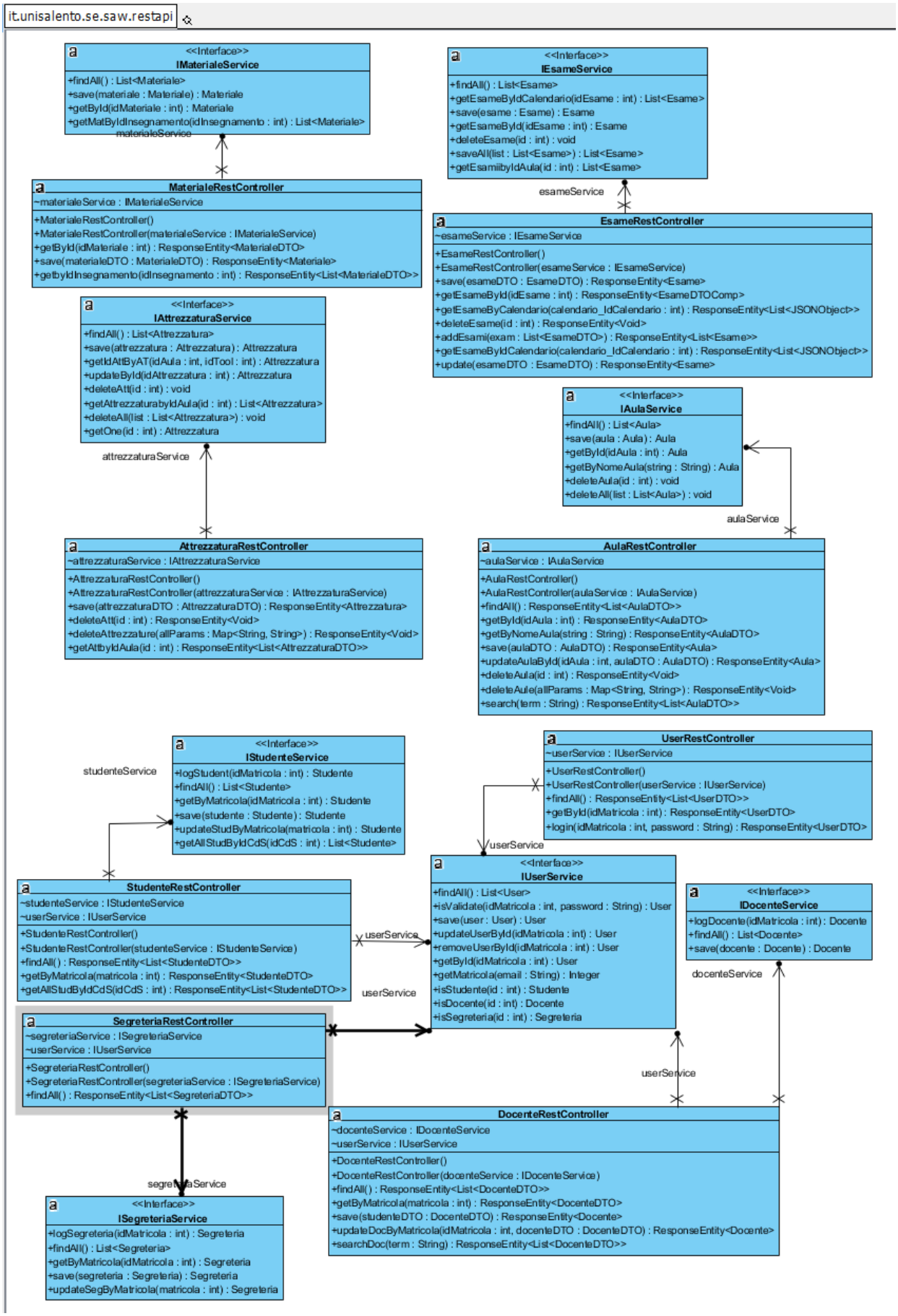

it.unisalento.se.saw.IService
<<Interface>>
IAulaService
+findAll() : List<Aula>
+save(aula : Aula) : Aula
+getById(idAula : int) : Aula
+getByNomeAula(string : String) : Aula
+deleteAula(id : int) : void
+deleteAll(list : List<Aula>) : void
<<Interface>>
IDocenteService
+logDocente(idMatricola : int) : Docente
+findAll() : List<Docente>
+save(docente : Docente) : Docente
<<Interface>>
IEsameService
+findAll() : List<Esame>
+getEsameByIdCalendario(idEsame : int) : List<Esame>
+save(esame : Esame) : Esame
+getEsameById(idEsame : int) : Esame
+deleteEsame(id : int) : void
+saveAll(list : List<Esame>) : List<Esame>
+getEsamiibyIdAula(id : int) : List<Esame>
<<Interface>>
IAttrezzaturaService
+findAll() : List<Attrezzatura>
+save(attrezzatura : Attrezzatura) : Attrezzatura
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+updateById(idAttrezzatura : int) : Attrezzatura
+deleteAtt(id : int) : void
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
+deleteAll(list : List<Attrezzatura>) : void
+getOne(id : int) : Attrezzatura
<<Interface>>
IRecensioneLService
+findAll() : List<RecensioneI>
+getRecLByInsegnamento(string : String) : List<RecensioneI>
+getRecLByVoto(voto : int) : List<RecensioneI>
+getById(idRecensioneL : int) : RecensioneI
+save(recensioneL : RecensioneI) : RecensioneI
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : RecensioneI
+getRecLByIdLezione(idLezione : int) : List<RecensioneI>
<<Interface>>
IMaterialeService
+findAll() : List<Materiale>
+save(materiale : Materiale) : Materiale
+getById(idMateriale : int) : Materiale
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>
<<Interface>>
IStudenteService
+logStudent(idMatricola : int) : Studente
+findAll() : List<Studente>
+getByMatricola(idMatricola : int) : Studente
+save(studente : Studente) : Studente
+updateStudByMatricola(matricola : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
<<Interface>>
IToolService
+findAll() : List<Tool>
+getById(idTool : int) : Tool
+getByName(string : String) : Tool
+save(tool : Tool) : Tool
+updateToolById(idTool : int) : Tool
<<Interface>>
IRecensioneMService
+findAll() : List<RecensioneM>
+getRecMatByInsegnamento(string : String) : List<RecensioneM>
+getRecMatByVoto(voto : int) : List<RecensioneM>
+getById(idRecensioneM : int) : RecensioneM
+save(recensioneM : RecensioneM) : RecensioneM
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : RecensioneM
+getRecByIdMateriale(idMateriale : int) : List<RecensioneM>
<<Interface>>
ISegreteriaService
+logSegreteria(idMatricola : int) : Segreteria
+findAll() : List<Segreteria>
+getByMatricola(idMatricola : int) : Segreteria
+save(segreteria : Segreteria) : Segreteria
+updateSegByMatricola(matricola : int) : Segreteria
<<Interface>>
IUserService
+findAll() : List<User>
+isValidate(idMatricola : int, password : String) : User
+save(user : User) : User
+updateUserById(idMatricola : int) : User
+removeUserById(idMatricola : int) : User
+getById(idMatricola : int) : User
+getMatricola(email : String) : Integer
+isStudente(id : int) : Studente
+isDocente(id : int) : Docente
+isSegreteria(id : int) : Segreteria
<<Interface>>
ILezioneService
+findAll() : List<Lezione>
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
+save(lezione : Lezione) : Lezione
+getLezioneById(idLezione : int) : Lezione
+getLezionibyIdAula(id : int) : List<Lezione>
+deleteLezione(id : int) : void
+saveAll(list : List<Lezione>) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
<<Interface>>
ICorsoDiStudioService
+findAll() : List<CorsoDiStudio>
+getByTipo(string : String) : List<CorsoDiStudio>
+getById(idCorso : int) : CorsoDiStudio
+save(corsoDiStudio : CorsoDiStudio) : CorsoDiStudio
<<Interface>>
ISegnalazioneService
+findAll() : List<Segnalazione>
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getById(idSegnalazione : int) : Segnalazione
+save(segnalazione : Segnalazione) : Segnalazione
+getByidDocente(id : int) : List<Segnalazione>
+getByidAula(id : int) : List<Segnalazione>
+getByidSegr(id : int) : List<Segnalazione>
<<Interface>>
IInsegnamentoService
+findAll() : List<Insegnamento>
+getById(idInsegnamento : int) : Insegnamento
+save(insegnamento : Insegnamento) : Insegnamento
+getByIdCorso(id : int) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
<<Interface>>
ICalendarioService
<<Property>> +all : List<CalendarioComponent>
<<Property>> +calendari : List<Calendario>
+findAll() : List<CalendarioComponent>
+getCalendarioById(idCalendario : int) : CalendarioComponent
+save(calendario : Calendario) : Calendario
+updateCalendarioById(idCalendario : int) : Calendario
+getCalendaribyCds(idCds : int) : List<CalendarioComponent>
+deleteCalendario(id : int) : void
+deleteAll(list : List<Calendario>) : void
+getOne(id : int) : Calendario


it.unisalento.se.saw.services
CorsoDiStudioService
-corsoDiStudioRepository : CorsoDiStudioRepository
+findAll() : List<CorsoDiStudio>
+getByTipo(string : String) : List<CorsoDiStudio>
+getById(idCorso : int) : CorsoDiStudio
+save(corsoDiStudio : CorsoDiStudio) : CorsoDiStudio
corsoDiStudioRepository
<<Interface>>
CorsoDiStudioRepository
+getByTipo(string : String) : List<CorsoDiStudio>
<<Interface>>
ICorsoDiStudioService
+findAll() : List<CorsoDiStudio>
+getByTipo(string : String) : List<CorsoDiStudio>
+getById(idCorso : int) : CorsoDiStudio
+save(corsoDiStudio : CorsoDiStudio) : CorsoDiStudio
<<Interface>>
ILezioneService
+findAll() : List<Lezione>
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
+save(lezione : Lezione) : Lezione
+getLezioneById(idLezione : int) : Lezione
+getLezioniByIdAula(id : int) : List<Lezione>
+deleteLezione(id : int) : void
+saveAll(list : List<Lezione>) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
<<Interface>>
CalendarioRepository
<<Property>> +all : List<Calendario>
+getCalendariByCds(idCds : int) : List<Calendario>
LezioneService
-lezioneRepository : LezioneRepository
+findAll() : List<Lezione>
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+save(lezione : Lezione) : Lezione
+getLezioneById(idLezione : int) : Lezione
+deleteLezione(id : int) : void
+saveAll(list : List<Lezione>) : List<Lezione>
+getLezioniByIdAula(id : int) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
<<Interface>>
ICalendarioService
<<Property>> +all : List<CalendarioComponent>
<<Property>> +calendari : List<Calendario>
+findAll() : List<CalendarioComponent>
+getCalendarioById(idCalendario : int) : CalendarioComponent
+save(calendario : Calendario) : Calendario
+updateCalendarioById(idCalendario : int) : Calendario
+getCalendariByCds(idCds : int) : List<CalendarioComponent>
+deleteCalendario(id : int) : void
+deleteAll(list : List<Calendario>) : void
+getOne(id : int) : Calendario
calendarioRepository
CalendarioService
~calendarioRepository : CalendarioRepository
~lezioneRepository : LezioneRepository
~esameRepository : EsameRepository
+findAll() : List<CalendarioComponent>
+getAll() : List<CalendarioComponent>
+getCalendarioById(idCalendario : int) : CalendarioComponent
+save(calendario : Calendario) : Calendario
+updateCalendarioById(idCalendario : int) : Calendario
+getCalendariByCds(idCds : int) : List<CalendarioComponent>
+deleteCalendario(id : int) : void
+deleteAll(list : List<Calendario>) : void
+getOne(id : int) : Calendario
+getCalendari() : List<Calendario>
lezioneRepository
<<Interface>>
LezioneRepository
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+getLezioniByCorso(nome : String) : List<Lezione>
+getLezioniByIdAula(id : int) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
<<Interface>>
IEsameService
+findAll() : List<Esame>
+getEsameByIdCalendario(idEsame : int) : List<Esame>
+save(esame : Esame) : Esame
+getEsameById(idEsame : int) : Esame
+deleteEsame(id : int) : void
+saveAll(list : List<Esame>) : List<Esame>
+getEsamibyIdAula(id : int) : List<Esame>
esameRepository
<<Interface>>
EsameRepository
+getEsameByIdCalendario(idCalendario : int) : List<Esame>
+getEsamibyIdAula(id : int) : List<Esame>
lezRepository
<<Interface>>
IAulaService
+findAll() : List<Aula>
+save(aula : Aula) : Aula
+getById(idAula : int) : Aula
+getByNomeAula(string : String) : Aula
+deleteAula(id : int) : void
+deleteAll(list : List<Aula>) : void
EsameService
~esameRepository : EsameRepository
+findAll() : List<Esame>
+getEsameByIdCalendario(idCalendario : int) : List<Esame>
+save(esame : Esame) : Esame
+getEsameById(idEsame : int) : Esame
+deleteEsame(id : int) : void
+saveAll(list : List<Esame>) : List<Esame>
+getEsamibyIdAula(id : int) : List<Esame>
esameRepository
<<Interface>>
AulaRepository
+getByNomeAula(string : String) : Aula
aulaRepository
AulaService
~aulaRepository : AulaRepository
~attRepository : AttrezzaturaRepository
~lezRepository : LezioneRepository
~esameRepository : EsameRepository
+findAll() : List<Aula>
+save(aula : Aula) : Aula
+getById(idAula : int) : Aula
+deleteAula(id : int) : void
+deleteAll(list : List<Aula>) : void
+getByNomeAula(string : String) : Aula
attRepository
AttrezzaturaService
~attrezzaturaRepository : AttrezzaturaRepository
+findAll() : List<Attrezzatura>
+save(attrezzatura : Attrezzatura) : Attrezzatura
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+updateById(idAttrezzatura : int) : Attrezzatura
+deleteAt(id : int) : void
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
+deleteAll(list : List<Attrezzatura>) : void
+getOne(id : int) : Attrezzatura
attrezzaturaRepository
<<Interface>>
AttrezzaturaRepository
+deleteAt(idAula : int, idTool : int) : void
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
<<Interface>>
IAttrezzaturaService
+findAll() : List<Attrezzatura>
+save(attrezzatura : Attrezzatura) : Attrezzatura
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+updateById(idAttrezzatura : int) : Attrezzatura
+deleteAt(id : int) : void
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
+deleteAll(list : List<Attrezzatura>) : void
+getOne(id : int) : Attrezzatura
<<Interface>>
InsegnamentoRepository
+getByIdCorso(id : int) : List<Insegnamento>
+getByDocente(cognome : String, nome : String) : List<Insegnamento>
+getByInsegnamento(nome : String) : List<Insegnamento>
+getByCorso(nome : String) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
insegnamentoRepository
InsegnamentoService
~insegnamentoRepository : InsegnamentoRepository
+findAll() : List<Insegnamento>
+getById(idInsegnamento : int) : Insegnamento
+save(insegnamento : Insegnamento) : Insegnamento
+getByIdCorso(id : int) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
<<Interface>>
IInsegnamentoService
+findAll() : List<Insegnamento>
+getById(idInsegnamento : int) : Insegnamento
+save(insegnamento : Insegnamento) : Insegnamento
+getByIdCorso(id : int) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
<<Interface>>
IRecensioneLService
+findAll() : List<Recensione>
+getRecLByInsegnamento(string : String) : List<Recensione>
+getRecLByVoto(voto : int) : List<Recensione>
+getById(idRecensioneL : int) : Recensione
+save(recensioneL : Recensione) : Recensione
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : Recensione
+getRecLByIdLezione(idLezione : int) : List<Recensione>
RecensioneLService
~recensioneLRepository : RecensioneLRepository
+findAll() : List<Recensione>
+getRecLByInsegnamento(string : String) : List<Recensione>
+getRecLByVoto(voto : int) : List<Recensione>
+getById(idRecensioneL : int) : Recensione
+save(recensioneL : Recensione) : Recensione
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : Recensione
+getRecLByIdLezione(idLezione : int) : List<Recensione>
recensioneLRepository
<<Interface>>
RecensioneLRepository
+getRecLByInsegnamento(string : String) : List<Recensione>
+getRecLByVoto(voto : int) : List<Recensione>
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : Recensione
+getRecLByIdLezione(idLezione : int) : List<Recensione>
<<Interface>>
SegnalazioneRepository
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getByIdDocente(id : int) : List<Segnalazione>
+getByIdAula(id : int) : List<Segnalazione>
+getByIdSegn(id : int) : List<Segnalazione>
<<Interface>>
ISegnalazioneService
+findAll() : List<Segnalazione>
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getById(idSegnalazione : int) : Segnalazione
+save(segnalazione : Segnalazione) : Segnalazione
+getByIdDocente(id : int) : List<Segnalazione>
+getByIdAula(id : int) : List<Segnalazione>
+getByIdSegn(id : int) : List<Segnalazione>
segnalazioneRepository
SegnalazioneService
~segnalazioneRepository : SegnalazioneRepository
+findAll() : List<Segnalazione>
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getById(idSegnalazione : int) : Segnalazione
+save(segnalazione : Segnalazione) : Segnalazione
+getByIdDocente(id : int) : List<Segnalazione>
+getByIdAula(id : int) : List<Segnalazione>
+getByIdSegn(id : int) : List<Segnalazione>
MaterialeService
~materialeRepository : MaterialeRepository
+findAll() : List<Materiale>
+save(materiale : Materiale) : Materiale
+getById(idMateriale : int) : Materiale
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>
<<Interface>>
IMaterialeService
+findAll() : List<Materiale>
+save(materiale : Materiale) : Materiale
+getById(idMateriale : int) : Materiale
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>
materialeRepository
<<Interface>>
MaterialeRepository
+getByName(string : String) : List<Materiale>
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>


it.unisalento.se.saw.services
<<Interface>>
IRecensioneMService
+findAll() : List<Recensionem>
+getRecMatByInsegnamento(string : String) : List<Recensionem>
+getRecMatByVoto(voto : int) : List<Recensionem>
+getById(idRecensioneM : int) : Recensionem
+save(recensioneM : Recensionem) : Recensionem
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : Recensionem
+getRecByIdMateriale(idMateriale : int) : List<Recensionem>
<<Interface>>
IToolService
+findAll() : List<Tool>
+getById(idTool : int) : Tool
+getByName(string : String) : Tool
+save(tool : Tool) : Tool
+updateToolById(idTool : int) : Tool
ToolService
~toolRepository : ToolRepository
+findAll() : List<Tool>
+getById(idTool : int) : Tool
+getByName(string : String) : Tool
+save(tool : Tool) : Tool
+updateToolById(idTool : int) : Tool
toolRepository
<<Interface>>
ToolRepository
+getByName(string : String) : Tool
<<Interface>>
RecensioneMRepository
+getRecMatByInsegnamento(string : String) : List<Recensionem>
+getRecMatByVoto(voto : int) : List<Recensionem>
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : Recensionem
+getRecByIdMateriale(idMateriale : int) : List<Recensionem>
recensioneMRepository
RecensioneMService
~recensioneMRepository : RecensioneMRepository
+findAll() : List<Recensionem>
+getRecMatByInsegnamento(string : String) : List<Recensionem>
+getRecMatByVoto(voto : int) : List<Recensionem>
+getById(idRecensioneM : int) : Recensionem
+save(recensioneM : Recensionem) : Recensionem
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : Recensionem
+getRecByIdMateriale(idMateriale : int) : List<Recensionem>
StudentService
~studenteRepository : StudenteRepository
+logStudent(idMatricola : int) : Studente
+findAll() : List<Studente>
+getByMatricola(idMatricola : int) : Studente
+save(studente : Studente) : Studente
+updateStudByMatricola(matricola : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
<<Interface>>
StudenteRepository
+logStudent(idMatricola : int) : Studente
+getbyMatricola(id : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
studenteRepository
studenteRepository
UserService
~userRepository : UserRepository
~docenteRepository : DocenteRepository
~studenteRepository : StudenteRepository
~segreteriaRepository : SegreteriaRepository
+findAll() : List<User>
+save(user : User) : User
+updateUserById(idMatricola : int) : User
+isValidate(idMatricola : int, password : String) : User
+getById(idMatricola : int) : User
+removeUserById(idMatricola : int) : User
+getMatricola(email : String) : Integer
+isStudente(id : int) : Studente
+isDocente(id : int) : Docente
+isSegreteria(id : int) : Segreteria
<<Interface>>
IStudenteService
+logStudent(idMatricola : int) : Studente
+findAll() : List<Studente>
+getByMatricola(idMatricola : int) : Studente
+save(studente : Studente) : Studente
+updateStudByMatricola(matricola : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
<<Interface>>
UserRepository
+isValidate(idMatricola : int, password : String) : User
+getMatricola(email : String) : Integer
segreteriaRepository
userRepository
<<Interface>>
SegreteriaRepository
+logSegreteria(idMatricola : int) : Segreteria
+getbyMatricola(id : int) : Segreteria
docenteRepository
<<Interface>>
DocenteRepository
+logDocente(idMatricola : int) : Docente
+getbyMatricola(id : int) : Docente
<<Interface>>
IUserService
+findAll() : List<User>
+isValidate(idMatricola : int, password : String) : User
+save(user : User) : User
+updateUserById(idMatricola : int) : User
+removeUserById(idMatricola : int) : User
+getById(idMatricola : int) : User
+getMatricola(email : String) : Integer
+isStudente(id : int) : Studente
+isDocente(id : int) : Docente
+isSegreteria(id : int) : Segreteria
segreteriaRepository
SegreteriaService
~segreteriaRepository : SegreteriaRepository
+logSegreteria(idMatricola : int) : Segreteria
+findAll() : List<Segreteria>
+getByMatricola(idMatricola : int) : Segreteria
+save(segreteria : Segreteria) : Segreteria
+updateSegByMatricola(matricola : int) : Segreteria
docenteRepository
DocenteService
~docenteRepository : DocenteRepository
+logDocente(idMatricola : int) : Docente
+findAll() : List<Docente>
+save(docente : Docente) : Docente
<<Interface>>
IDocenteService
+logDocente(idMatricola : int) : Docente
+findAll() : List<Docente>
+save(docente : Docente) : Docente
<<Interface>>
ISegreteriaService
+logSegreteria(idMatricola : int) : Segreteria
+findAll() : List<Segreteria>
+getByMatricola(idMatricola : int) : Segreteria
+save(segreteria : Segreteria) : Segreteria
+updateSegByMatricola(matricola : int) : Segreteria


it.unisalento.se.saw.repositories
<<Interface>>
AulaRepository
+getByNomeAula(string : String) : Aula
<<Interface>>
MaterialeRepository
+getByName(string : String) : List<Materiale>
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>
<<Interface>>
AttrezzaturaRepository
+deleteAtt(idAula : int, idTool : int) : void
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
<<Interface>>
StudenteRepository
+logStudent(idMatricola : int) : Studente
+getbyMatricola(id : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
<<Interface>>
SegreteriaRepository
+logSegreteria(idMatricola : int) : Segreteria
+getbyMatricola(id : int) : Segreteria
<<Interface>>
DocenteRepository
+logDocente(idMatricola : int) : Docente
+getbyMatricola(id : int) : Docente
<<Interface>>
EsameRepository
+getEsameByIdCalendario(idCalendario : int) : List<Esame>
+getEsamiibyIdAula(id : int) : List<Esame>
<<Interface>>
UserRepository
+isValidate(idMatricola : int, password : String) : User
+getMatricola(email : String) : Integer
<<Interface>>
RecensioneLRepository
+getRecLByInsegnamento(string : String) : List<RecensioneI>
+getRecLByVoto(voto : int) : List<RecensioneI>
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : RecensioneI
+getRecLByIdLezione(idLezione : int) : List<RecensioneI>
<<Interface>>
CorsoDiStudioRepository
+getByTipo(string : String) : List<CorsoDiStudio>
<<Interface>>
RecensioneMRepository
+getRecMatByInsegnamento(string : String) : List<Recensionem>
+getRecMatByVoto(voto : int) : List<Recensionem>
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : Recensionem
+getRecByIdMateriale(idMateriale : int) : List<Recensionem>
<<Interface>>
ToolRepository
+getByName(string : String) : Tool
<<Interface>>
LezioneRepository
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+getLezioniByCorso(nome : String) : List<Lezione>
+getLezionibyIdAula(id : int) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
<<Interface>>
InsegnamentoRepository
+getByIdCorso(id : int) : List<Insegnamento>
+getByDocente(cognome : String, nome : String) : List<Insegnamento>
+getByInsegnamento(nome : String) : List<Insegnamento>
+getByCorso(nome : String) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
<<Interface>>
SegnalazioneRepository
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getByIdDocente(id : int) : List<Segnalazione>
+getByIdAula(id : int) : List<Segnalazione>
+getByIdSegr(id : int) : List<Segnalazione>
<<Interface>>
CalendarioRepository
<<Property>> +all : List<Calendario>
+getCalendaribyCds(idCds : int) : List<Calendario>


it.unisalento.se.saw.restapi
<<Interface>>
IToolService
+findAll() : List<Tool>
+getById(idTool : int) : Tool
+getByName(string : String) : Tool
+save(tool : Tool) : Tool
+updateToolById(idTool : int) : Tool
toolService
ToolRestController
~toolService : IToolService
+ToolRestController()
+ToolRestController(toolService : IToolService)
+findAll() : ResponseEntity<List<ToolDTO>>
+save(toolDTO : ToolDTO) : ResponseEntity<Tool>
<<Interface>>
IRecensioneMService
+findAll() : List<RecensioneM>
+getRecMatByInsegnamento(string : String) : List<RecensioneM>
+getRecMatByVoto(voto : int) : List<RecensioneM>
+getById(idRecensioneM : int) : RecensioneM
+save(recensioneM : RecensioneM) : RecensioneM
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idMateriale : int) : RecensioneM
+getRecByIdMateriale(idMateriale : int) : List<RecensioneM>
recensioneMService
RecensioneMRestController
~recensioneMService : IRecensioneMService
+RecensioneMRestController(recensioneMService : IRecensioneMService)
+RecensioneMRestController()
+save(recensioneMDTO : RecensioneMDTO) : ResponseEntity<RecensioneM>
+getByMatricolaStudIdInsegnamento(idMatricola : int, idInsegnamento : int, idMateriale : int) : ResponseEntity<RecensioneMDTO>
+getRecByIdMateriale(idMateriale : int) : ResponseEntity<List<RecensioneMDTO>>
<<Interface>>
IInsegnamentoService
+findAll() : List<Insegnamento>
+getById(idInsegnamento : int) : Insegnamento
+save(insegnamento : Insegnamento) : Insegnamento
+getByIdCorso(id : int) : List<Insegnamento>
+getByIdDocente(idDocente : int) : List<Insegnamento>
insegnamentoService
CalendarioRestController
~calendarioService : ICalendarioService
+CalendarioRestController()
+CalendarioRestController(calendarioService : ICalendarioService)
+getAll() : ResponseEntity<List<CalendarioDTO>>
+getCalendarioById(idCalendario : int) : ResponseEntity<JSONObject>
+save(calendarioDTO : CalendarioDTO) : ResponseEntity<Calendario>
+deleteCal(id : int) : ResponseEntity<Void>
+deleteCal(allParams : Map<String, String>) : ResponseEntity<Void>
calendarioService
InsegnamentoRestController
~insegnamentoService : IInsegnamentoService
+InsegnamentoRestController()
+InsegnamentoRestController(insegnamentoService : IInsegnamentoService)
+findAll() : ResponseEntity<List<InsegnamentoDTO>>
+getById(idInsegnamento : int) : ResponseEntity<InsegnamentoDTO>
+save(insegnamentoDTO : InsegnamentoDTO) : ResponseEntity<Insegnamento>
+updateById(idInsegnamento : int, insegnamentoDTO : InsegnamentoDTO) : ResponseEntity<Insegnamento>
+getByIdCorso(id : int) : ResponseEntity<List<InsegnamentoDTO>>
+getByDocente(idDocente : int) : ResponseEntity<List<InsegnamentoDTO>>
<<Interface>>
ICalendarioService
<<Property>> +all : List<CalendarioComponent>
<<Property>> +calendari : List<Calendario>
+findAll() : List<CalendarioComponent>
+getCalendarioById(idCalendario : int) : CalendarioComponent
+save(calendario : Calendario) : Calendario
+updateCalendarioById(idCalendario : int) : Calendario
+getCalendaribyCds(idCds : int) : List<CalendarioComponent>
+deleteCalendario(id : int) : void
+deleteAll(list : List<Calendario>) : void
+getOne(id : int) : Calendario
CorsoDiStudioRestController
~corsoDiStudioService : ICorsoDiStudioService
+CorsoDiStudioRestController()
+CorsoDiStudioRestController(corsoDiStudioService : ICorsoDiStudioService)
+findAll() : ResponseEntity<List<CorsoDiStudioDTO>>
+getByTipo(string : String) : ResponseEntity<List<CorsoDiStudioDTO>>
+getById(idCorso : int) : ResponseEntity<CorsoDiStudioDTO>
+save(corsoDiStudioDTO : CorsoDiStudioDTO) : ResponseEntity<CorsoDiStudio>
+update(idCorso : int, corsoDiStudioDTO : CorsoDiStudioDTO) : ResponseEntity<CorsoDiStudio>
corsoDiStudioService
<<Interface>>
ICorsoDiStudioService
+findAll() : List<CorsoDiStudio>
+getByTipo(string : String) : List<CorsoDiStudio>
+getById(idCorso : int) : CorsoDiStudio
+save(corsoDiStudio : CorsoDiStudio) : CorsoDiStudio
SegnalazioneRestController
~segnalazioneService : ISegnalazioneService
+SegnalazioneRestController()
+SegnalazioneRestController(segnalazioneService : ISegnalazioneService)
+getById(idSegnalazione : int) : ResponseEntity<SegnalazioneDTO>
+save(segnalazioneDTO : SegnalazioneDTO) : ResponseEntity<Segnalazione>
+updateStatoSegnal(idSegnalazione : int, segnalazioneDTO : SegnalazioneDTO) : ResponseEntity<Segnalazione>
+getbyIdDocente(id : int) : ResponseEntity<List<SegnalazioneDTO>>
+filterDoc(id : int, stato : String) : ResponseEntity<List<SegnalazioneDTO>>
+getbyIdAula(id : int) : ResponseEntity<List<SegnalazioneDTO>>
+getbyIdSegr(id : int) : ResponseEntity<List<SegnalazioneDTO>>
+filterSegr(id : int, stato : String) : ResponseEntity<List<SegnalazioneDTO>>
segnalazioneService
<<Interface>>
ISegnalazioneService
+findAll() : List<Segnalazione>
+getByStato(string : String) : List<Segnalazione>
+getByDocente(cognome : String, nome : String) : List<Segnalazione>
+getById(idSegnalazione : int) : Segnalazione
+save(segnalazione : Segnalazione) : Segnalazione
+getByIdDocente(id : int) : List<Segnalazione>
+getByIdAula(id : int) : List<Segnalazione>
+getByIdSegr(id : int) : List<Segnalazione>
RecensioneLRestController
~recensioneLService : IRecensioneLService
+RecensioneLRestController()
+RecensioneLRestController(recensioneLService : IRecensioneLService)
+save(recensioneLDTO : RecensioneLDTO) : ResponseEntity<RecensioneL>
+getByMatricolaStudIdInsegIdMaterial(idMatricola : int, idInsegnamento : int, idLezione : int) : ResponseEntity<RecensioneLDTO>
+getRecLByIdLezione(idLezione : int) : ResponseEntity<List<RecensioneLDTO>>
recensioneLService
<<Interface>>
IRecensioneLService
+findAll() : List<RecensioneL>
+getRecLByInsegnamento(string : String) : List<RecensioneL>
+getRecLByVoto(voto : int) : List<RecensioneL>
+getById(idRecensioneL : int) : RecensioneL
+save(recensioneL : RecensioneL) : RecensioneL
+getByMatricolaStudIdInsegIdLez(idMatricola : int, idInsegnamento : int, idLezione : int) : RecensioneL
+getRecLByIdLezione(idLezione : int) : List<RecensioneL>
<<Interface>>
ILezioneService
+findAll() : List<Lezione>
+getLezioniByIdCalendario(idCalendario : int) : List<Lezione>
+getCalLezioniByCorso(nome : String) : List<Lezione>
+save(lezione : Lezione) : Lezione
+getLezioneById(idLezione : int) : Lezione
+getLezioniByIdAula(id : int) : List<Lezione>
+deleteLezione(id : int) : void
+saveAll(list : List<Lezione>) : List<Lezione>
+getLezioniByIdDocente(idDocente : int) : List<Lezione>
+getLezioniByIdInsegnamento(idInsegnamento : int) : List<Lezione>
lezioneService
LezioneRestController
~lezioneService : ILezioneService
+LezioneRestController()
+LezioneRestController(lezioneService : ILezioneService)
+save(lezioneDTO : LezioneDTO) : ResponseEntity<Lezione>
+getLezioneById(idLezione : int) : ResponseEntity<LezioneDTOComp>
+getLezioniByCalendario(calendario_IdCalendario : int) : ResponseEntity<List<JSONObject>>
+deleteLez(id : int) : ResponseEntity<Void>
+addLezioni(lez : List<LezioneDTO>) : ResponseEntity<List<Lezione>>
+getLezioniByIdCalendario(calendario_IdCalendario : int) : ResponseEntity<List<JSONObject>>
+update(lezioneDTO : LezioneDTO) : ResponseEntity<Lezione>
+getLezioniByIdDocente(idDocente : int) : ResponseEntity<List<JSONObject>>
+getCalLezioniByCorso(nome : String) : ResponseEntity<List<JSONObject>>
+getLezioniByIdInsegnamento(idInsegnamento : int) : ResponseEntity<List<LezioneDTOComp>>


it.unisalento.se.saw.restapi
<<Interface>>
IMaterialeService
+findAll() : List<Materiale>
+save(materiale : Materiale) : Materiale
+getById(idMateriale : int) : Materiale
+getMatByIdInsegnamento(idInsegnamento : int) : List<Materiale>
materialeService
MaterialeRestController
~materialeService : IMaterialeService
+MaterialeRestController()
+MaterialeRestController(materialeService : IMaterialeService)
+getById(idMateriale : int) : ResponseEntity<MaterialeDTO>
+save(materialeDTO : MaterialeDTO) : ResponseEntity<Materiale>
+getbyIdInsegnamento(idInsegnamento : int) : ResponseEntity<List<MaterialeDTO>>
<<Interface>>
IEsameService
+findAll() : List<Esame>
+getEsameById(idEsame : int) : List<Esame>
+save(esame : Esame) : Esame
+getEsameById(idEsame : int) : Esame
+deleteEsame(id : int) : void
+saveAll(list : List<Esame>) : List<Esame>
+getEsamiibyIdAula(id : int) : List<Esame>
esameService
EsameRestController
~esameService : IEsameService
+EsameRestController()
+EsameRestController(esameService : IEsameService)
+save(esameDTO : EsameDTO) : ResponseEntity<Esame>
+getEsameById(idEsame : int) : ResponseEntity<EsameDTOComp>
+getEsameByCalendario(calendario_IdCalendario : int) : ResponseEntity<List<JSONObject>>
+deleteEsame(id : int) : ResponseEntity<Void>
+addEsami(exam : List<EsameDTO>) : ResponseEntity<List<Esame>>
+getEsameByIdCalendario(calendario_IdCalendario : int) : ResponseEntity<List<JSONObject>>
+update(esameDTO : EsameDTO) : ResponseEntity<Esame>
<<Interface>>
IAttrezzaturaService
+findAll() : List<Attrezzatura>
+save(attrezzatura : Attrezzatura) : Attrezzatura
+getIdAttByAT(idAula : int, idTool : int) : Attrezzatura
+updateById(idAttrezzatura : int) : Attrezzatura
+deleteAtt(id : int) : void
+getAttrezzaturabyIdAula(id : int) : List<Attrezzatura>
+deleteAll(list : List<Attrezzatura>) : void
+getOne(id : int) : Attrezzatura
attrezzaturaService
<<Interface>>
IAulaService
+findAll() : List<Aula>
+save(aula : Aula) : Aula
+getById(idAula : int) : Aula
+getByNomeAula(string : String) : Aula
+deleteAula(id : int) : void
+deleteAll(list : List<Aula>) : void
aulaService
AttrezzaturaRestController
~attrezzaturaService : IAttrezzaturaService
+AttrezzaturaRestController()
+AttrezzaturaRestController(attrezzaturaService : IAttrezzaturaService)
+save(attrezzaturaDTO : AttrezzaturaDTO) : ResponseEntity<Attrezzatura>
+deleteAtt(id : int) : ResponseEntity<Void>
+deleteAttrezzature(allParams : Map<String, String>) : ResponseEntity<Void>
+getAttbyIdAula(id : int) : ResponseEntity<List<AttrezzaturaDTO>>
AulaRestController
~aulaService : IAulaService
+AulaRestController()
+AulaRestController(aulaService : IAulaService)
+findAll() : ResponseEntity<List<AulaDTO>>
+getById(idAula : int) : ResponseEntity<AulaDTO>
+getByNomeAula(string : String) : ResponseEntity<AulaDTO>
+save(aulaDTO : AulaDTO) : ResponseEntity<Aula>
+updateAulaById(idAula : int, aulaDTO : AulaDTO) : ResponseEntity<Aula>
+deleteAula(id : int) : ResponseEntity<Void>
+deleteAule(allParams : Map<String, String>) : ResponseEntity<Void>
+search(term : String) : ResponseEntity<List<AulaDTO>>
studenteService
<<Interface>>
IStudenteService
+logStudent(idMatricola : int) : Studente
+findAll() : List<Studente>
+getByMatricola(idMatricola : int) : Studente
+save(studente : Studente) : Studente
+updateStudByMatricola(matricola : int) : Studente
+getAllStudByIdCdS(idCdS : int) : List<Studente>
UserRestController
~userService : IUserService
+UserRestController()
+UserRestController(userService : IUserService)
+findAll() : ResponseEntity<List<UserDTO>>
+getById(idMatricola : int) : ResponseEntity<UserDTO>
+login(idMatricola : int, password : String) : ResponseEntity<UserDTO>
userService
<<Interface>>
IUserService
+findAll() : List<User>
+isValidate(idMatricola : int, password : String) : User
+save(user : User) : User
+updateUserById(idMatricola : int) : User
+removeUserById(idMatricola : int) : User
+getById(idMatricola : int) : User
+getMatricola(email : String) : Integer
+isStudente(id : int) : Studente
+isDocente(id : int) : Docente
+isSegreteria(id : int) : Segreteria
<<Interface>>
IDocenteService
+logDocente(idMatricola : int) : Docente
+findAll() : List<Docente>
+save(docente : Docente) : Docente
docenteService
StudenteRestController
~studenteService : IStudenteService
~userService : IUserService
+StudenteRestController()
+StudenteRestController(studenteService : IStudenteService)
+findAll() : ResponseEntity<List<StudenteDTO>>
+getByMatricola(matricola : int) : ResponseEntity<StudenteDTO>
+getAllStudByIdCdS(idCdS : int) : ResponseEntity<List<StudenteDTO>>
userService
userService
SegreteriaRestController
~segreteriaService : ISegreteriaService
~userService : IUserService
+SegreteriaRestController()
+SegreteriaRestController(segreteriaService : ISegreteriaService)
+findAll() : ResponseEntity<List<SegreteriaDTO>>
userService
DocenteRestController
~docenteService : IDocenteService
~userService : IUserService
+DocenteRestController()
+DocenteRestController(docenteService : IDocenteService)
+findAll() : ResponseEntity<List<DocenteDTO>>
+getByMatricola(matricola : int) : ResponseEntity<DocenteDTO>
+save(studenteDTO : DocenteDTO) : ResponseEntity<Docente>
+updateDocByMatricola(idMatricola : int, docenteDTO : DocenteDTO) : ResponseEntity<Docente>
+searchDoc(term : String) : ResponseEntity<List<DocenteDTO>>
segreteriaService
<<Interface>>
ISegreteriaService
+logSegreteria(idMatricola : int) : Segreteria
+findAll() : List<Segreteria>
+getByMatricola(idMatricola : int) : Segreteria
+save(segreteria : Segreteria) : Segreteria
+updateSegByMatricola(matricola : int) : Segreteria

Tecnologie Implementate

La realizzazione del sistema software è stata possibile tramite l'utilizzo di differenti tecnologie di tipo open-source. Per la realizzazione del Back-End "Web Server", è stato usato il linguaggio di programmazione **JAVA** tramite l'uso dell'ambiente di sviluppo Eclipse. Si è fatto uso delle diverse librerie standard messe a disposizione dallo stesso linguaggio, ma tramite il sito **Maven Repository** sono state introdotte nuove librerie necessarie per il corretto funzionamento del server, andando ad aggiungerle e a compilare nella "build.gradle" del progetto Spring-Gradle.
Fra le tante librerie aggiunte, quella che assume un'importanza maggiore è la libreria di **Firebase**. Firebase è una piattaforma software che ha bisogno di essere collegata al nostro progetto, in modo da sviluppare applicazioni real-time e di alta qualità. Per la creazione del progetto sono state utilizzate alcune delle funzionalità:

- **Firebase Cloud Database;**
- **Firebase Cloud Storage;**
- **Firebase Authentication;**
- **Firebase Cloud Messaging;**

Firebase Cloud Messaging è usato per l'invio di notifiche o di dati nel momento in cui viene seguita una determinata azione. In questo modo, sia che l'applicazione sia in background o in foreground, l'utente vedrà a schermo la notifica ricevuta con il relativo messaggio, se presente.
Il Web Server utilizza una componente di filtraggio, tramite il quale analizza le richieste ricevute dalle applicazioni di Front-End, andando ad elaborare solo quelle che presentano il campo Authorization settato nell'header, escluse quelle di registrazione e di login.
Firebase Cloud Database è stato utilizzato per creare delle room di chat tra i vari utenti delle Mobile App e dove viene salvata attraverso uno *Storage Cloud* la "history" delle Chat.
Firebase Cloud Storage, viene utilizzato per l'update dei file dei diversi insegnamenti da parte del Docente e attraverso la generazione di un link si può accedere allo storage eseguire il download del file.

La creazione delle applicazioni di Front-End, cioè la WebApp per la *Segreteria* e il *Docente*, e le Mobile Apps per lo *Studente* e il *Docente*, sono state realizzate tramite l'utilizzo del linguaggio di programmazione **TypeScript**, il quale è un'estensione del linguaggio **JavaScript**.

La ***WebApp*** è stata realizzata tramite l'uso di **Angular6** e per ampliarne le funzionalità sono stati installati e implementati diversi moduli. Sono stati utilizzati moduli come **NG-FullCalendar**, per la visualizzazione dei calendari e l'implementazione delle varie lezioni e esami del corso;
SweetAlert2, per gestire e mostrare le notifiche in foreground con una grafica e caratteristiche differenti da quelle tradizionali e **Angular-Material** per personalizzare la parte grafica delle Web Apps.

Le due ***Mobile Apps*** sono state sviluppate tramite il framework **Ionic4,** che permette la creazione di app ibride attraverso l'utilizzo di **Ionic4** stesso per quanto riguarda i componenti grafici, **Angular4** per quanto riguarda la logica di business e plugin **Cordova** per permettere l'interazione dell'app con i vari componenti hardware dei dispositivi mobile.
In particolare i moduli più importanti utilizzati sono stati **GoogleMaps** per gestire la visualizzazione delle mappe e la ricerca degli indirizzi e le loro coordinate; **NG-Calendar** per creare un calendario interattivo e gestire in vari eventi per giorno, mese o settima; i vari moduli di **Firebase** per la gestione dei download, delle chat e delle notifiche e **SweetAlert2**, per gestire e mostrare le notifiche in foreground con una grafica e caratteristiche differenti da quelle tradizionali;

La struttura delle applicazioni sia Web che Mobile, avendo alla base **Angular6** e **Ionic4**, sono composta da 4 elementi in particolare:

- ***Components***, che svolgono la funzione di controller e quindi gestiscono la logica di business;
- ***Pagine html5***, per la parte grafica;
- ***Classi di model***, tramite le quali vengono aggregati logicamente e gestiti i dati;
- ***Services***, tramite i quali vengono gestite le chiamate asincrone ai servizi REST esposti.

Entrando nello specifico, anche nelle app di Front-End è stato utilizzato il pattern Factory in quanto si è deciso di disaccoppiare i *components* dai *services*. Tutto questo è stato possibile poichè per ogni Service si è creata un'interfaccia e una sua specifica implementazione; il component ha conoscenza solo delle interfacce dei Services e comunica con loro tramite esse. Infine la definizione della relazione fra l'interfaccia e la sua relativa implementazione è stata realizzata in *app.module* nella sezione *services.*

Inoltre lavorando in team è stato fondamentale l'utilizzo di Git come strumento di controllo del versioning del codice.

Test e Metriche

La correttezza, la completezza e l'affidabilità sono caratteristiche imprescindibili di un progetto software. È possibile verificare la presenza di tali caratteristiche solo tramite procedure di testing, le quali, tramite l'uso di opportuni test e metriche, ci forniscono delle informazioni quantitative, in modo da determinare e correggere malfunzionamenti e difetti prima della fase di pubblicazione del software. Il Web Server è l'elemento più delicato del progetto. Un suo eventuale malfunzionamento o difetto può compromettere la qualità delle informazioni ricevute o inviate dalle restanti applicazioni. Per questo motivo è stato usato il framework di *unit testing* **JUnit**, in modo da testare la correttezza delle singole unità che compongono il software. Seguendo questo approccio si è deciso di testare la parte dei servizi **Rest** e di seguito mostreremo alcuni esempi di test effettuati sul nostro codice:

Test JUnit

```
@RunWith(MockitoJUnitRunner.class)
public class UserRestControllerTest {

    public static final MediaType APPLICATION_JSON_UTF8 = new MediaType(MediaType.APPLICATION_JSON.getType(),
            MediaType.APPLICATION_JSON.getSubtype(),
            Charset.forName("utf8"));

    private MockMvc mockMvc;

    // @Spy
    @InjectMocks
    private UserRestController userController;


    @Mock
    private IUserService userService;

    @InjectMocks
    UserDTO userDTO;


    @Before
    public void setUp() {

        // mockMvc = MockMvcBuilders.standaloneSetup(new UserRestController(userServiceMock)).build();
        /*set up fisso test*/
        MockitoAnnotations.initMocks(this);
        mockMvc = MockMvcBuilders.standaloneSetup(userController).build();

    }
```

```java
@Test /*scelgo la funzione da testare*/
public void findAll() throws Exception {

    List<User> userList = new ArrayList<User>();
    userList.add(new User());
    userList.add(new User());

    when(userService.findAll()).thenReturn(userList); /*impostare la condizione di test*/

    mockMvc.perform(get("/user/findAll"))
        .andExpect(status().isOk())  /*mi aspetto una risposta http "OK"*/
        .andExpect(content().contentType(APPLICATION_JSON_UTF8));

    when(userService.findAll()).thenThrow(Exception.class); /*impostare la condizione di test*/

    mockMvc.perform(get("/user/findAll"))
    .andExpect(status().isBadRequest());

    verify(userService, times(2)).findAll(); /*verifica che con una chiamata al metodo le asserzioni siano corrette*/
    verifyNoMoreInteractions(userService);
}
```

JUnit Test Coverage

it.unisalento.se.saw.restapi (12-giu-2019 15.37.39)

Element	Coverage	Covered Instructions	Missed Instruct...	Total Instructions
UniApp	85,3 %	20.028	3.454	23.482
src/main/java	75,9 %	8.934	2.841	11.775
it.unisalento.se.saw.services	0,0 %	0	1.823	1.823
it.unisalento.se.saw.restapi	92,7 %	6.937	548	7.485
it.unisalento.se.saw.dto.composit	79,8 %	824	209	1.033
it.unisalento.se.saw.configuratior	0,0 %	0	167	167
it.unisalento.se.saw.util	44,4 %	63	79	142
it.unisalento.se.saw.dto	98,8 %	988	12	1.000
it.unisalento.se.saw.strategy	97,6 %	122	3	125
build/generated/src/java	61,8 %	990	613	1.603
src/test/java	100,0 %	10.104	0	10.104

Metriche

Per poter testare alcune proprietà del software o per verificarne le relative specifiche si usano le metriche software, che permettono di estrapolare delle informazioni quantitative. Per tale motivo è stato usato il plugin Metrics, mediante il quale si determinano i valori di media e di deviazione standard di metriche come: WMC, DIT, NOC, McCabe Cyclomatic Complexity ed altre.

Console | Problems | Debug | Servers | Debug Shell | Coverage | Metrics - UniApp - McCabe Cyclomatic Complexity (avg/max per method)

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
McCabe Cyclomatic Complexity (avg/max per method)		1,219	0,935	10	/UniApp/src/main/java/it/unisalento/se/saw/resta...	filterDoc
java		1,321	1,119	10	/UniApp/src/main/java/it/unisalento/se/saw/resta...	filterDoc
it.unisalento.se.saw.restapi		2,806	2,084	10	/UniApp/src/main/java/it/unisalento/se/saw/resta...	filterDoc
SegnalazioneRestController.java		4,6	3,323	10	/UniApp/src/main/java/it/unisalento/se/saw/resta...	filterDoc
CorsoDiStudioRestController.java		3,143	2,531	8	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getByTipo
InsegnamentoRestController.java		3,5	2,5	7	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getByIdCorso
LezioneRestController.java		3	1,633	7	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getLezioniByIdInsegnar
RecensioneMRestController.java		2,6	2,245	7	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getRecByIdMateriale
UserRestController.java		3,2	2,04	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	login
MaterialeRestController.java		2,4	1,855	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getbyIdInsegnamento
CalendarioRestController.java		2,429	1,591	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getAll
ToolRestController.java		2,5	2,062	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	findAll
DocenteRestController.java		2,857	2,03	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	findAll
AulaRestController.java		2,5	1,432	6	/UniApp/src/main/java/it/unisalento/se/saw/resta...	findAll
RecensioneLRestController.java		2	1,095	4	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getRecLByIdLezione
EsameRestController.java		2,333	1,054	4	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getEsameByCalendario
SegreteriaRestController.java		1,667	0,943	3	/UniApp/src/main/java/it/unisalento/se/saw/resta...	findAll
StudenteRestController.java		2	0,894	3	/UniApp/src/main/java/it/unisalento/se/saw/resta...	findAll
AttrezzaturaRestController.java		2	0,816	3	/UniApp/src/main/java/it/unisalento/se/saw/resta...	deleteAttrezzature
it.unisalento.se.saw.services		1,33	1,211	8	/UniApp/src/main/java/it/unisalento/se/saw/servic...	deleteAll
it.unisalento.se.saw.strategy		1,857	1,355	4	/UniApp/src/main/java/it/unisalento/se/saw/strate...	sortList
it.unisalento.se.saw.util		1,75	0,829	3	/UniApp/src/main/java/it/unisalento/se/saw/util/P...	hash
it.unisalento.se.saw.dto.composite		1,011	0,105	2	/UniApp/src/main/java/it/unisalento/se/saw/dto/c...	toJson
it.unisalento.se.saw.IService		1	0	1	/UniApp/src/main/java/it/unisalento/se/saw/IServi...	findAll
it.unisalento.se.saw.configurations		1	0	1	/UniApp/src/main/java/it/unisalento/se/saw/confi...	viewResolver
it.unisalento.se.saw.dto		1	0	1	/UniApp/src/main/java/it/unisalento/se/saw/dto/T...	getIdTool
it.unisalento.se.saw.repositories		1	0	1	/UniApp/src/main/java/it/unisalento/se/saw/repos...	getByStato
it.unisalento.se.saw.controllers		0	0			
Number of Parameters (avg/max per method)		0,722	1,091	14	/UniApp/src/main/java/it/unisalento/se/saw/dto/c...	LezioneDTOComp
Nested Block Depth (avg/max per method)		1,024	0,752	5	/UniApp/src/main/java/it/unisalento/se/saw/resta...	getbyIdInsegnamento
Afferent Coupling (avg/max per packageFragment)		13,583	19,259	69	/UniApp/build/generated/src/java/it/unisalento/s...	
Efferent Coupling (avg/max per packageFragment)		9,25	7,373	18	/UniApp/src/main/java/it/unisalento/se/saw/servic...	
Instability (avg/max per packageFragment)		0,546	0,372	1	/UniApp/src/main/java/it/unisalento/se/saw/confi...	
Abstractness (avg/max per packageFragment)		?	?	1	/UniApp/src/main/java/it/unisalento/se/saw/IServi...	
Normalized Distance (avg/max per packageFragment)		?	?	0,812	/UniApp/build/generated/src/java/it/unisalento/s...	
Depth of Inheritance Tree (avg/max per type)		0,82	0,592	4	/UniApp/src/main/java/it/unisalento/se/saw/confi...	
Weighted methods per Class (avg/max per type)	1343	10,492	9,242	46	/UniApp/src/main/java/it/unisalento/se/saw/resta...	
Number of Children (avg/max per type)	24	0,188	0,512	3	/UniApp/src/main/java/it/unisalento/se/saw/dto/...	
Number of Overridden Methods (avg/max per type)	16	0,125	0,559	4	/UniApp/src/main/java/it/unisalento/se/saw/dto/S...	
Lack of Cohesion of Methods (avg/max per type)		0,347	0,403	1,2	/UniApp/src/test/java/it/unisalento/se/saw/restapi...	
Number of Attributes (avg/max per type)	395	3,086	4,098	19	/UniApp/src/main/java/it/unisalento/se/saw/dto/...	
Number of Static Attributes (avg/max per type)	30	0,234	0,897	9	/UniApp/src/main/java/it/unisalento/se/saw/confi...	
Number of Methods (avg/max per type)	1098	8,578	7,702	38	/UniApp/src/main/java/it/unisalento/se/saw/dto/...	
Number of Static Methods (avg/max per type)	4	0,031	0,278	3	/UniApp/src/main/java/it/unisalento/se/saw/util/P...	
Specialization Index (avg/max per type)		0,033	0,164	1	/UniApp/src/main/java/it/unisalento/se/saw/confi...	
Number of Classes (avg/max per packageFragment)	128	10,667	6,908	18	/UniApp/src/main/java/it/unisalento/se/saw/servic...	
Number of Interfaces (avg/max per packageFragment)	33	2,75	5,932	16	/UniApp/src/main/java/it/unisalento/se/saw/IServi...	
Number of Packages	12					
Total Lines of Code	10610					
Method Lines of Code (avg/max per method)	3903	3,542	8,517	74	/UniApp/src/test/java/it/unisalento/se/saw/restapi...	login

Metodologia di Sviluppo

Il sistema è stato realizzato applicando le regole legate alle metodologie di sviluppo agile. Per tale motivo durante le prime fasi è stato deciso di affidarsi a Scrum. Questo ha portato alla suddivisione del lavoro in Sprint di durata bisettimanale. Alla scadenza dello sprint è stato rilasciato lo Sprint Backlog in cui sono state raccolte i task completati e il tempo necessario per portarli a termine.

BACKLOG ITEMS	ORE
Comprensione traccia e stesura casi d'uso	**6**
Progettazione Database	**6**
Configurazione Development Software	**8**
Progettazione Back-End	**5**
Creazione e Popolazione Database	**15**
Implementazione Servizi Rest	**20**
Implementazione Interfaccia Web App	**20**
Implementazione Logica Web App	**12**
Verifica Web App	**4**
Implementazione Interfaccia Mobile Apps	**20**
Implementazione Logica Mobile Apps	**12**
Verifica Mobile Apps	**4**
Implementazione Firebase	**18**
Test e Metriche	**20**
Stesura Documentazione	**6**
TOTALE	**164**

Sprint Backlog

SPRINT 1 ITEM	L	M	M	G	V	S	D	L	M	M	G	V	S	D
Comprensione traccia e stesura casi d'uso	4	2												
Progettazione Database		2	4											
Configurazione Development Software				4	4									
Progettazione Back-End						2		3						
Creazione e Popolazione Database						3		2	2	2	2	2	2	
Implementazione Servizi Rest									3	3	3	3	3	
Test							1							1
TOTALE	4	4	4	4	4	5	1	5	5	5	5	5	5	1

SPRINT 2 ITEM	L	M	M	G	V	S	D	L	M	M	G	V	S	D
Implementazione Servizi Rest	3	2												
Implementazione Interfaccia Web App	3	3	3	3	3	3	2							
Implementazione Logica Web App			3	3	3	3								
Verifica Web App								2						
Implementazione Interfaccia Mobile Apps								4	5	3	3	3	1	1
Implementazione Logica Mobile Apps										3	3	3	3	
Verifica Mobile Apps													2	
TOTALE	6	5	6	6	6	6	2	6	5	6	6	6	6	1

SPRINT 3 ITEM	L	M	M	G	V	S	D	L	M	M	G	V	S	D
Implementazione Firebase	4	4	4	4	2									
Verifica Web App								2						
Verifica Mobile Apps								2						
Test e Metriche								3	5	4	4	3	2	
Stesura documentazion									1	1	2	2		
TOTALE	4	4	4	4	2	0	0	6	6	5	6	5	2	0

Burndown chart