Tecnologie Informatiche per il Web - Politecnico Di Milano / Giugno 2025

Simone Ranfoni - 10934656 Giacomo Merlo - 10848938

Indice

1 Introduzione

Il presente documento descrive l'analisi, la progettazione e i componenti di un'applicazione web per la gestione di un portale universitario che consente: ai docenti di inserire, modificare, visualizzare valutazioni di studenti iscritti a corsi da essi insegnati; agli studenti di visualizzare i voti ottenuti agli esami assieme alla possibilità di rifiutarli. Gli strumenti utilizzati per la realizzazione del progetto, come consigliato dal Professor Piero Fraternali, sono Eclipse e il server Tomcat per l'esecuzione dell'applicazione. Sono state sviluppate due versioni del sito con le seguenti caratteristiche:

1. **Versione HTML pura**: Frontend basato unicamente su HTML, reso dinamico tramite i template Thymeleaf. Backend in Java, con Servlet che gestiscono richieste GET e POST. Interazione con database MySQL Workbench tramite DAO (Data Access Objects), i cui risultati sono salvati in Beans e passati alle Servlet. I risultati dell'interazione utente vengono impostati nel WebContext e visualizzati tramite Thymeleaf.

2. **Versione RIA (Rich Internet Application)**: Struttura backend parzialmente riutilizzata, con adattamenti alle Servlet per l'assenza del meccanismo a template di Thymeleaf. Frontend interamente riscritto in JavaScript. Ogni interazione utente che scatena una richiesta al server è gestita in modo asincrono (AJAX); le risposte sono costituite da oggetti serializzati in JSON. JavaScript interpreta questi dati e aggiorna dinamicamente i componenti HTML. Le pagine HTML sono state consolidate in due principali (una per lo studente e una per il docente), oltre alla pagina di login. JavaScript gestisce la visibilità delle diverse sezioni HTML.

2 Analisi dei dati

L'analisi dei dati è stata condotta attraverso una duplice scansione del testo di specifica fornito. La prima scansione è mirata all'individuazione di entità, attributi e relazioni, fondamentale per la successiva definizione del database. La seconda analizza pagine (viste), componenti grafici, eventi e azioni, per una corretta organizzazione e progettazione delle pagine del sito.

2.1 Entità, Attributi, Relazioni

Un'applicazione permette di verbalizzare gli esiti degli esami di un appello. Il docente accede tramite login e seleziona nella HOME page un corso da una lista dei propri corsi, ordinata per nome del corso in modo alfabetico decrescente, e poi sceglie una data d'appello del corso scelto da un elenco ordinato per data decrescente. Ogni corso ha un solo docente. La selezione dell'appello porta a una pagina ISCRITTI, che mostra una tabella con tutti gli iscritti all'appello. La tabella riporta i seguenti dati: matricola, cognome e nome, email, corso di laurea, voto e stato di valutazione. Il voto può non essere ancora definito. Lo stato di valutazione dello studente rispetto all'appello può assumere i valori: non inserito, inserito, pubblicato, rifiutato e verbalizzato. Selezionando un'etichetta nell'intestazione della tabella, l'utente ordina le righe in base al valore di tale etichetta (ad esempio, selezionando "cognome" la tabella è riordinata in base al cognome). Successive selezioni della stessa etichetta invertono l'ordinamento: si parte con l'ordinamento crescente. Il valore del voto viene considerato ordinato nel modo seguente: , assente, rimandato, riprovato, 18, 19, ..., 30, 30 e lode. Nella tabella della pagina ISCRITTI ad ogni riga corrisponde un bottone "MODIFICA". Premendo il bottone compare una pagina con una form che mostra tutti i dati dello studente selezionato e un campo di input in cui è possibile scegliere il voto. L'invio della form provoca la modifica o l'inserimento del voto. Inizialmente le righe sono nello stato di valutazione "non inserito". L'inserimento e le successive eventuali modifiche portano la riga nello stato di valutazione "inserito". Alla tabella della pagina ISCRITTI è associato un bottone PUBBLICA che comporta la pubblicazione delle righe con lo stato di valutazione INSERITO. La pubblicazione rende il voto non più modificabile dal docente e visibile allo studente e cambia lo stato di valutazione della riga dello studente a "pubblicato". Lo studente accede tramite login e seleziona nella HOME page un corso tra quelli a cui è iscritto mediante una lista ordinata in modo alfabetico decrescente e poi una data d'appello del corso scelto selezionata da un elenco ordinato per data decrescente. Uno studente può essere iscritto a più appelli dello stesso corso. La selezione della data d'appello porta a una pagina ESITO che mostra il messaggio "Voto non ancora definito" se il docente non ha ancora pubblicato il risultato per quello studente in quell'appello. Altrimenti, la pagina mostra i dati dello studente, del corso, dell'appello e il voto assegnato. Se il voto è tra 18 e 30 e lode compare un bottone RIFIUTA. Premendo tale bottone la pagina mostra gli stessi dati con la dizione aggiunta "Il voto è stato rifiutato" e senza il bottone RIFIUTA. Il rifiuto del voto cambia lo stato di valutazione a "rifiutato" della riga dello studente per quell'appello nella pagina ISCRITTI del docente. Nella pagina ISCRITTI del docente la tabella degli iscritti è associata anche a un bottone VERBALIZZA. La pressione del bottone provoca il cambio di stato a "verbalizzato" per le righe nello stato "pubblicato" o "rifiutato" e comporta anche la creazione di un verbale e la disabilitazione della possibilità di rifiutare il voto. Il rifiuto implica la verbalizzazione di "rimandato" come voto. Un verbale ha un codice generato dal sistema, una data e ora di creazione ed è associato all'appello del corso a cui si riferisce e agli studenti (con nome, cognome, matricola e voto) che passano allo stato "verbalizzato". A seguito della pressione del bottone VERBALIZZA compare una pagina VERBALE che mostra i dati completi del verbale creato. Il docente dispone anche di una pagina VERBALI, in cui sono elencati tutti i verbali da lui creati, ordinati on modo alfabetico crescente per nome del corso e poi per data crescente di appello di ogni corso.

Entità, Attributi, Relazioni

2.2 Pagine (Viste), Componenti Grafici, Eventi, Azioni

Un'applicazione permette di verbalizzare gli esiti degli esami di un appello. Il docente accede tramite login e seleziona nella HOME page un corso da una lista dei propri corsi, ordinata per nome del corso in modo alfabetico decrescente, e poi sceglie una data d'appello del corso scelto da un elenco ordinato per data decrescente. Ogni corso ha un solo docente. La selezione dell'appello porta a una pagina ISCRITTI, che mostra una tabella con tutti gli iscritti all'appello. La tabella riporta i seguenti dati: matricola, cognome e nome, email, corso di laurea, voto e stato di valutazione. Il voto può non essere ancora definito. Lo stato di valutazione dello studente rispetto all'appello può assumere i valori: non inserito, inserito, pubblicato, rifiutato e verbalizzato. Selezionando un'etichetta nell'intestazione della tabella, l'utente ordina le righe in base al valore di tale etichetta (ad esempio, selezionando "cognome" la tabella è riordinata in base al cognome). Successive selezioni della stessa etichetta invertono l'ordinamento: si parte con l'ordinamento crescente. Il valore del voto viene considerato ordinato nel modo seguente: , assente, rimandato, riprovato, 18, 19, ..., 30, 30 e lode. Nella tabella della pagina ISCRITTI ad ogni riga corrisponde un bottone "MODIFICA". Premendo il bottone compare una pagina con una form che mostra tutti i dati dello studente selezionato e un campo di input in cui è possibile scegliere il voto. L'invio della form provoca la modifica o l'inserimento del voto. Inizialmente le righe sono nello stato di valutazione "non inserito". L'inserimento e le successive eventuali modifiche portano la riga nello stato di valutazione "inserito". Alla tabella della pagina ISCRITTI è associato un bottone PUBBLICA che comporta la pubblicazione delle righe con lo stato di valutazione INSERITO. La pubblicazione rende il voto non più modificabile dal docente e visibile allo studente e cambia lo stato di valutazione della riga dello studente a "pubblicato". Lo studente accede tramite login e seleziona nella HOME page un corso tra quelli a cui è iscritto mediante una lista ordinata in modo alfabetico decrescente e poi una data d'appello del corso scelto selezionata da un elenco ordinato per data decrescente. Uno studente può essere iscritto a più appelli dello stesso corso. La selezione della data d'appello porta a una pagina ESITO che mostra il messaggio "Voto non ancora definito" se il docente non ha ancora pubblicato il risultato per quello studente in quell'appello. Altrimenti, la pagina mostra i dati dello studente, del corso, dell'appello e il voto assegnato. Se il voto è tra 18 e 30 e lode compare un bottone RIFIUTA. Premendo tale bottone la pagina mostra gli stessi dati con la dizione aggiunta "Il voto è stato rifiutato" e senza il bottone RIFIUTA. Il rifiuto del voto cambia lo stato di valutazione a "rifiutato" della riga dello studente per quell'appello nella pagina ISCRITTI del docente. Nella pagina ISCRITTI del docente la tabella degli iscritti è associata anche a un bottone VERBALIZZA. La pressione del bottone provoca il cambio di stato a "verbalizzato" per le righe nello stato "pubblicato" o "rifiutato" e comporta anche la creazione di un verbale e la disabilitazione della possibilità di rifiutare il voto. Il rifiuto implica la verbalizzazione di "rimandato" come voto. Un verbale ha un codice generato dal sistema, una data e ora di creazione ed è associato all'appello del corso a cui si riferisce e agli studenti (con nome, cognome, matricola e voto) che passano allo stato "verbalizzato". A seguito della pressione del bottone VERBALIZZA compare una pagina VERBALE che mostra i dati completi del verbale creato. Il docente dispone anche di una pagina VERBALI, in cui sono elencati tutti i verbali da lui creati, ordinati on modo alfabetico crescente per nome del corso e poi per data crescente di appello di ogni corso.

Pagine, componenti grafici, eventi, azioni

3 Completamento della specifica

Di seguito vengono riportati chiarimenti e dettagli aggiuntivi rispetto alla specifica iniziale, definiti durante la realizzazione del progetto.

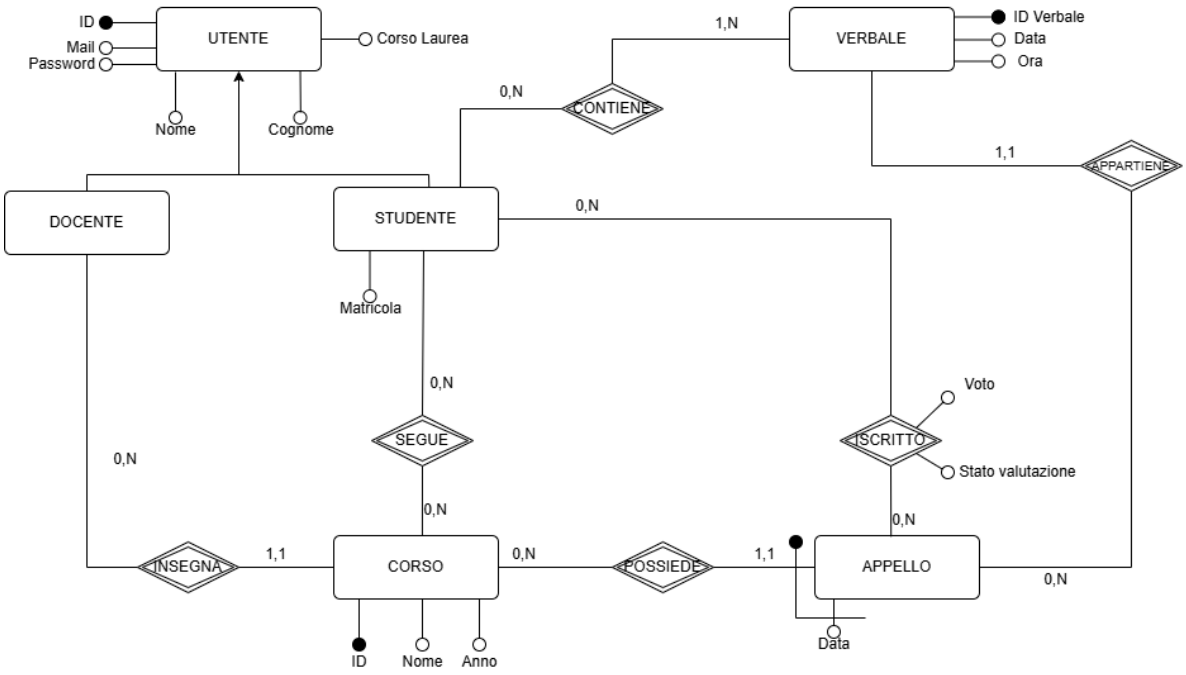
- Un corso potrebbe non avere appelli previsti, ma uno studente potrebbe seguire comunque il corso. Per questo motivo è stata aggiunta una relazione ulteriore di "Iscrizioni_Corsi" che associa ad ogni corso tutti i suoi iscritti.
- Ogni corso DEVE avere uno e un solo docente, ma possono esistere docenti a cui non sono stati assegnati corsi da insegnare.
- Possono esistere studenti non iscritti ad alcun corso (e, di conseguenza, a nessun esame).
- Un docente può decidere di verbalizzare, per uno stesso corso e data d'appello, gruppi di voti in momenti separati. Ne consegue la possibile creazione di diversi verbali per una stessa data d'appello. Una prerogativa è, però, che chiaramente ogni studente iscritto ad un appello di un dato esame comparirà in uno e un solo verbale. (Il suo voto, una volta verbalizzato, è unico, come stabilito dalla relazione "Iscrizioni_Appello"; l'unione degli studenti appartenenti ai verbali di un appello di un esame darà come risultato gli interi iscritti a quel preciso esame).

- Non esistono verbali vuoti: un docente è impossibilitato a verbalizzare (ma anche a pubblicare) 0 voti.
- Non esiste la possibilità per lo studente di "accettare" un voto; come in una situazione reale, il non-rifiuto e la successiva verbalizzazione da parte del professore comportano l'accettazione della valutazione.
- Per evitare che uno studente clicchi per errore il bottone, anche nella versione HTML pura è stata inserita la conferma del rifiuto del voto, con la conseguente aggiunta di una pagina HTML dedicata e 2 bottoni per rifiutare il voto o annullare l'operazione. Ci si attiene comunque alla specifica per la dizione aggiunta che conferma il rifiuto e la conseguente rimozione del bottone.

4 Database Design

In questa sezione vengono presentati i diagrammi concettuale e logico del database progettato per l'applicazione.

4.1 Schema Concettuale (E-R)



Lo schema E-R modella le entità principali come Utente, Corso, Appello, Verbale e le loro relazioni, includendo gli attributi identificati nell'analisi. La relazione 'Iscrizioni_Appello' è stata modellata come una tabella associativa per gestire la registrazione degli studenti agli appelli con i relativi voti e stati di valutazione. La relazione 'Iscrizioni_Corsi' gestisce gli studenti iscritti ai corsi indipendentemente dagli appelli.

4.2 Schema Logico Relazionale

ENTITA'

Utente(**id**, mail, psw, nome, cognome, matricola, corso di laurea)

Corso(**id**, nome, anno, id_prof)

Verbale(**id**, data_verbale, ora_verbale, id_corso, data)

Appello(**data, id_corso**)

RELAZIONI

Iscrizioni_Corsi(**id_corso**, **id_studente**)

Iscrizioni_Appello(**id_corso**, **data**, **id_studente**, voto, stato)

Studenti_Verbale(**id_verbale**, **id_studente**)

Lo schema logico traduce il modello E-R in tabelle relazionali, specificando chiavi primarie ed esterne per mantenere l'integrità referenziale. Ad esempio, la tabella 'Iscrizioni_Appello' avrà chiavi esterne verso 'Utente' (id_studente) e 'Appello' (data, id_corso); la tabella 'Iscrizioni_Corsi' verso 'Corso' (id_corso) e 'Utente' (id_studente); la tabella 'Studenti_Verbale' verso 'Utente' (id_studente). La tabella 'Corsi' avrà chiave esterna verso 'Utente' (id_prof), la tabella 'Verbale' verso 'Appello' (data, id_corso), la tabella 'Appello' verso 'Corso' (id_corso).

5 Application Design

Questa sezione illustra il design dell'applicazione per entrambe le versioni implementate: HTML pura e RIA. Vengono utilizzati diagrammi IFML (Interaction Flow Modeling Language) per rappresentare graficamente le interazioni e il flusso di navigazione.

5.1 Versione HTML pura

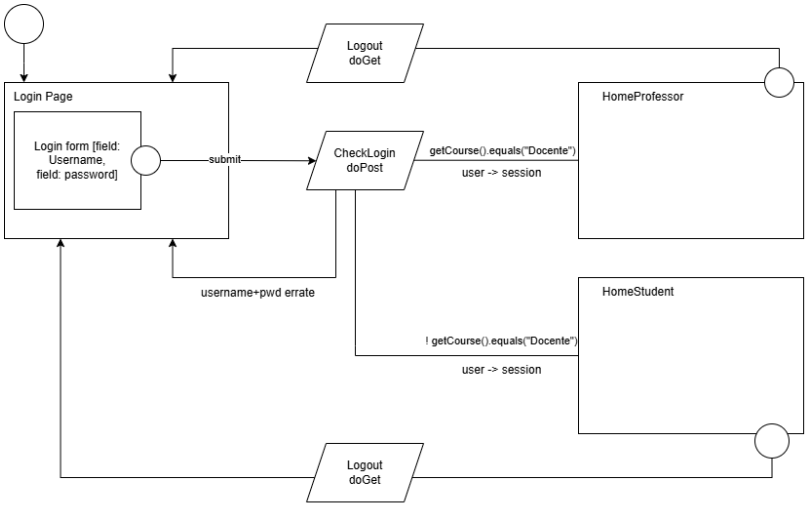


Figura 1: Schema delle interazioni di login/logout.

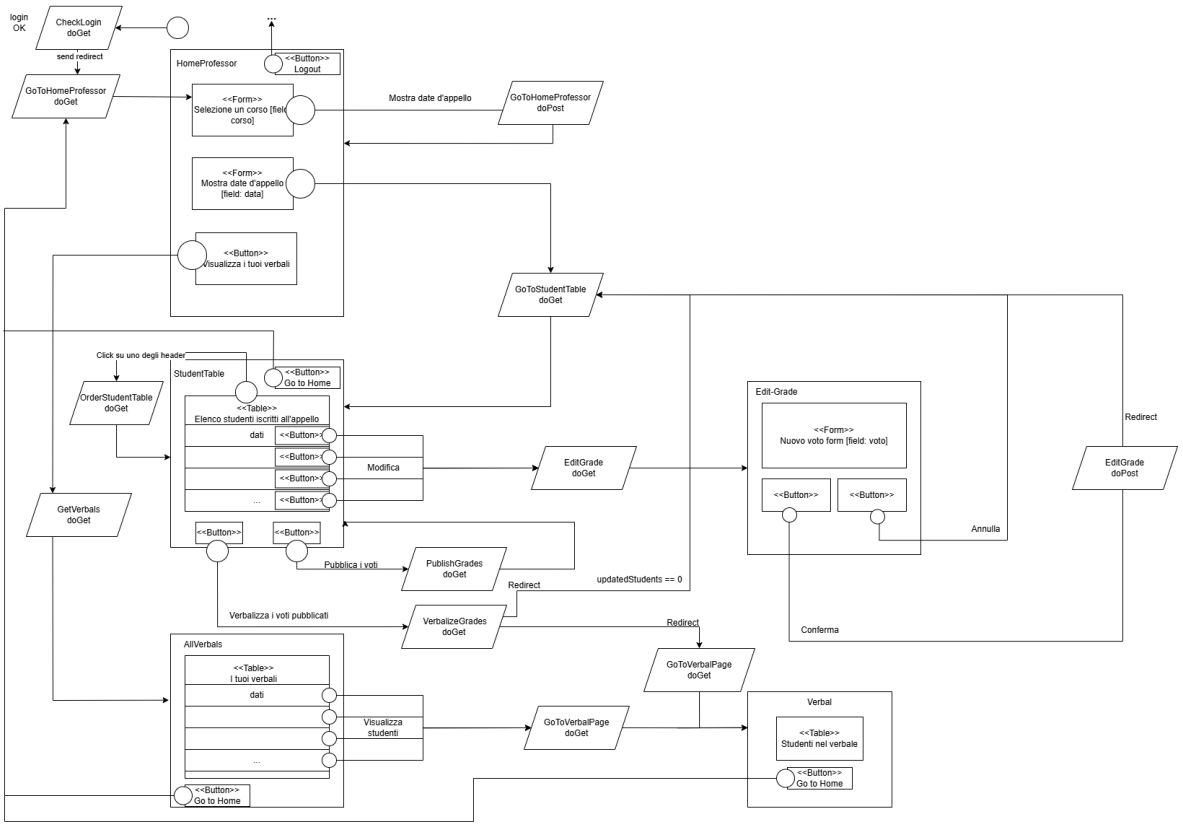


Figura 2: Schema delle interazioni di un utente di tipo Professore.

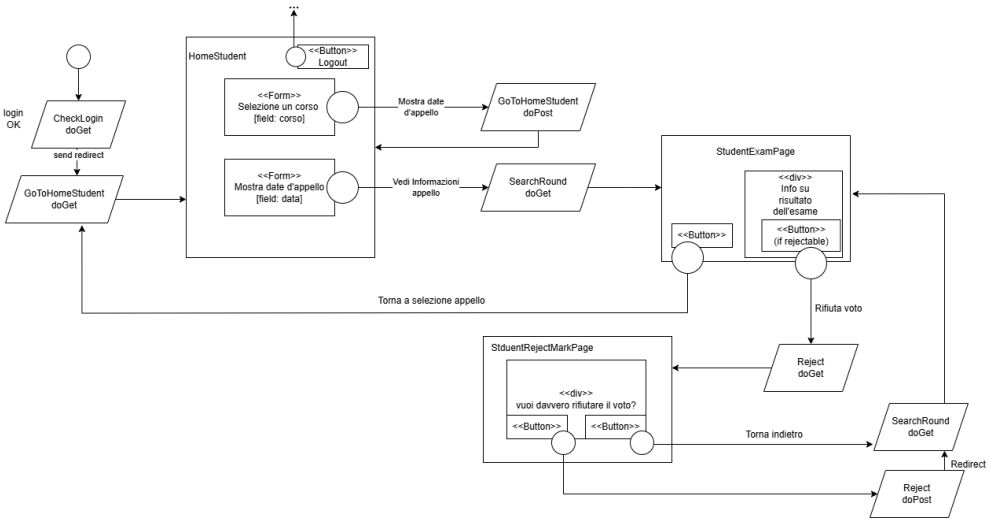


Figura 3: Schema delle interazioni di un utente di tipo Studente.

La versione HTML pura si basa su un modello multi-pagina, dove ogni interazione significativa dell'utente (es. selezione di un corso, richiesta di modifica di un voto) porta al caricamento di una nuova pagina HTML generata dinamicamente lato server tramite Thymeleaf.

5.2 Versione RIA (Rich Internet Application)

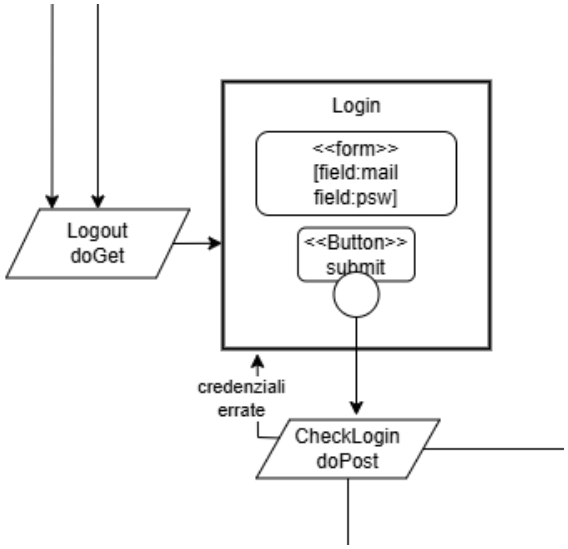


Figura 4: Schema delle interazioni di login/logout.

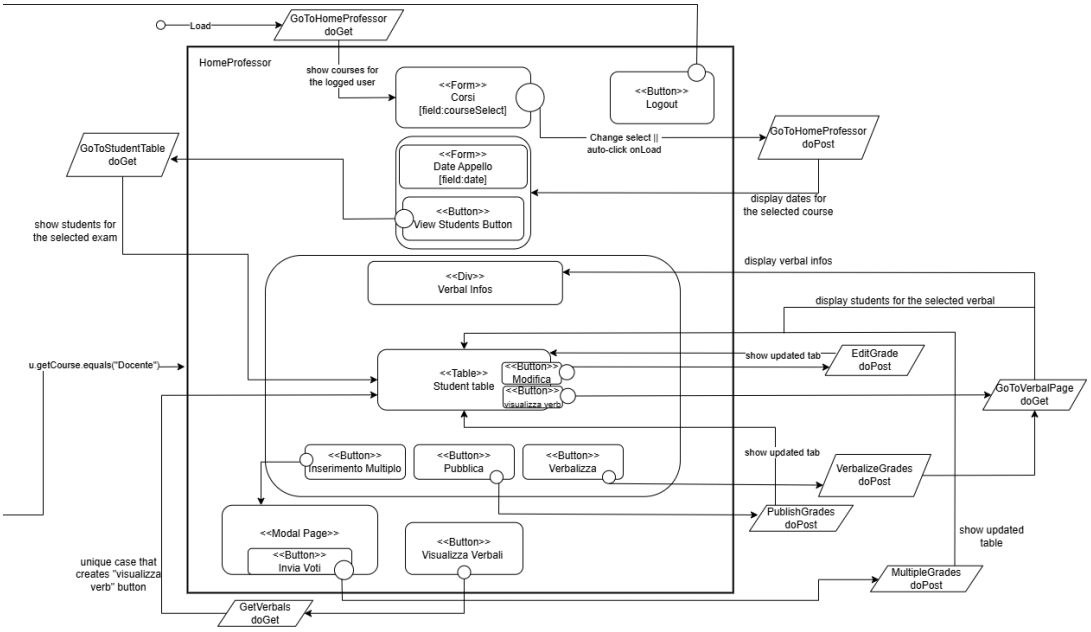


Figura 5: Schema delle interazioni di un utente di tipo Professore.

La versione RIA adotta un approccio a pagina singola (o quasi, con pagine principali per login, docente e studente). Le interazioni utente vengono gestite principalmente lato client tramite JavaScript, che effettua chiamate AJAX asincrone al server. Il server risponde con dati JSON, e JavaScript si occupa di aggiornare la vista dinamicamente senza ricaricare l'intera pagina.

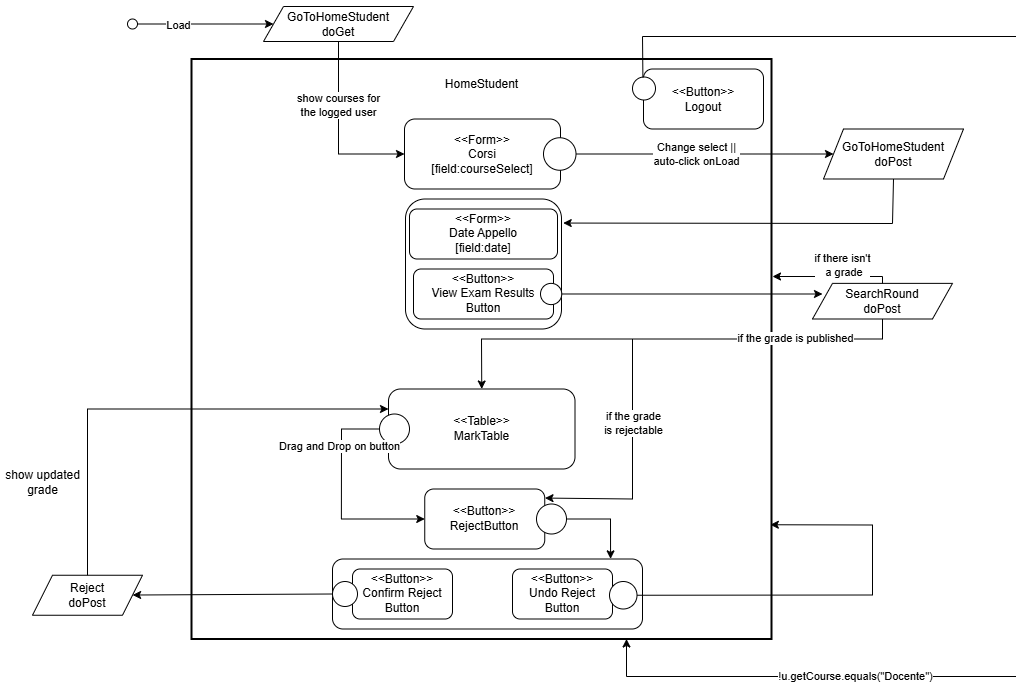


Figura 6: Schema delle interazioni di un utente di tipo Studente.

6 Componenti Server Side

I componenti lato server sono stati sviluppati in Java e sono in gran parte condivisi tra la versione HTML pura e la versione RIA. La struttura backend è molto simile tra le due versioni; per la versione RIA è stato necessario solo un riadattamento delle servlet per gestire risposte con il formato JSON invece di interagire con il meccanismo a template di Thymeleaf.

6.1 Controllers (Servlet)

Le servlet gestiscono le richieste HTTP provenienti dal client, coordinano la logica di business e interagiscono con i DAO per l'accesso ai dati.

- **CheckLogin**: Gestisce il processo di login degli utenti (docenti e studenti).
- **EditGrade**: Permette la modifica o l'inserimento di un voto per uno studente iscritto a un appello.
- **(GetUserData)**: Servlet ausiliaria per ottenere rapidamente i dati dell'utente loggato.
- **GetVerbals**: Recupera l'elenco dei verbali creati da un docente.
- **GoToHomeProfessor**: Duplice funzione per ottenere i corsi insegnati e le date d'appello di uno tra essi.
- **GoToHomeStudent**: Duplice funzione per ottenere i corsi seguiti e le date d'appello a cui lo studente è iscritto.

- **GoToStudentTable**: Recupera gli studenti iscritti a un determinato appello di un corso.
- **GoToVerbalPage**: Recupera le informazioni di un verbale, inclusi gli studenti che lo compongono.
- **Logout**: Gestisce il logout dell'utente.
- **(MultipleGrades)**: Gestisce l'inserimento multiplo della pagina modale.
- **<OrderStudentTable>**: Restituisce la tabella di studenti ordinata secondo un criterio scelto dall'utente.
- **PublishGrades**: Gestisce la pubblicazione dei voti inseriti.
- **Reject**: Gestisce la richiesta di rifiuto di un voto da parte dello studente.
- **SearchRound**: Gestisce la ricerca dell'esito di un appello di un determinato esame.
- **VerbalizeGrades**: Gestisce il processo di verbalizzazione dei voti pubblicati, di creazione del verbale e di aggiunta degli studenti ad esso associato. Si noti che è stato previsto un meccanismo di **rollback** per queste tre azioni a cascata, evitando dunque l'aggiunta di informazioni parziali in database in caso di errori.

(): solo RIA; <>: solo HTML

6.2 Model Objects (Beans)

I Beans sono classi Java semplici utilizzate per rappresentare e trasportare i dati tra i vari layer dell'applicazione (es. dai DAO alle Servlet, e dalle Servlet ai template Thymeleaf nella versione HTML).

- **Course**: Rappresenta l'entità corso.
- **Exam**: Rappresenta l'entità appello.
- **ExamResult**: Rappresenta l'esito di un esame per uno studente.
- **RegisteredStudent**: Rappresenta uno studente iscritto a un appello, con i suoi dati specifici.
- **UserBean**: Rappresenta un utente generico del sistema.
- **VerbalBean**: Rappresenta l'entità verbale.

I beans sono gli stessi tra versione RIA e HTML.

6.3 Data Access Objects (DAOs)

I DAO sono responsabili dell'interazione con il database (MySQL Workbench). Incapsulano la logica di accesso ai dati, eseguendo query SQL e mappando i risultati in Beans.

- **CourseDAO**: Gestisce le operazioni CRUD (Create, Read, Update, Delete) per fornire i gli appelli dei corsi (professore) o gli appelli a cui lo studente è iscritto.
- **ExamDAO**: Gestisce le operazioni CRUD per fornire l'esito dell'esame allo studente o rifiutarlo.
- **ProfessorDAO**: Gestisce le operazioni CRUD per trovare i corsi insegnati dal professore.
- **StudentDAO**: Gestisce le operazioni CRUD per trovare i corsi a cui lo studente è iscritto.
- **StudentTableDAO**: Gestisce le operazioni CRUD per tutto ciò che riguarda gli studenti iscritti ad un appello, incluse modifica, pubblicazione e verbalizzazione dei voti.
- **UserDAO**: Gestisce le operazioni CRUD per il login dell'utente.
- **VerbalDAO**: Gestisce le operazioni CRUD per l'entità Verbale.

7 Componenti Client Side (Versione RIA)

Nella versione RIA, il frontend è stato stravolto con l'uso intensivo di JavaScript per creare un'esperienza utente più dinamica e reattiva. Le interazioni utente generano richieste AJAX asincrone al server, e JavaScript si occupa di manipolare il DOM per visualizzare i risultati.

- **HomeProfessor.html**: Pagina principale per il docente.
  - `professorForms.js`
    - *CoursesList*
      - `Show`: Al ricaricamento della pagina, richiede al server la lista dei corsi insegnati dal docente loggato.
      - `Update`: Inserisce nelle opzioni del primo form (selezione corso) il risultato della query al server.
    - *DatesList*
      - `Show`: A seguito della scelta di un corso e della precedente richiesta delle date (fatta dal PageOrchestrator), popola il secondo form (selezione data appello).
- `studentTable.js`
  - *PageOrchestrator*
    - `createStudentTable`: Crea la tabella degli studenti dopo la selezione di un corso e una data d'appello.

    * `createTableHeader`: Crea l'intestazione della tabella degli studenti.
    * `createStudentRow`: Crea una riga della tabella inserendo i dati dello studente e il bottone per la modifica del voto.
    * `createGradeDropDown`: Crea il menu a tendina per selezionare un voto per uno studente.
    * `addActionButtons`: Crea i bottoni di pubblicazione, verbalizzazione e inserimento multiplo (se ci sono studenti disponibili per utilizzarlo).
    * `updateGrade`: Effettua una chiamata al server per modificare il voto.
    * `verbalizeGrades`: Effettua una chiamata al server per verbalizzare i voti pubblicati o rifiutati.
    * `publishGrades`: Effettua una chiamata al server per pubblicare i voti inseriti.
    * `showMultipleGradesModal`: Analizza quali studenti sono idonei per l'inserimento multiplo, crea la tabella apposita e mostra la finestra modale.
    * `hideModal`: Nasconde la finestra modale.
    * `submitMultipleGrades`: Effettua una chiamata al server per inviare l'inserimento multiplo dei voti.
    * `getVerbalData`: Effettua una chiamata al server per ottenere i dati di un verbale.
    * `displayVerbalData`: Nasconde altre tabelle (se presenti) e mostra le informazioni sul verbale e gli studenti inclusi.
    * `createSimpleStudentTableHeader`: Crea un'intestazione di tabella senza bottoni (per la visualizzazione nel verbale).
    * `createVerbalInfo`: Compone le etichette contenenti le informazioni sul verbale.
    * `getVerbals`: Effettua una chiamata al server per ottenere i verbali del docente loggato.
    * `createVerbalTable`: Crea la tabella dei verbali del docente loggato.

- `sortTable.js`: Contiene funzioni ausiliarie per l'ordinamento delle tabelle.
  - `getCellValue`: Ottiene il contenuto di una cella della tabella.
  - `createComparer`: Crea una funzione di comparazione per due elementi di una colonna, usata per l'ordinamento.
  - `makeSortable`: Rende una tabella ordinabile, aggiungendo elementi grafici (es. frecce) nell'header.
  - `sortTable`: Applica l'ordinamento alla tabella sulla colonna selezionata e secondo l'ordine scelto dall'utente.
  - `updateArrowColors`: Aggiorna il colore delle frecce di ordinamento per indicare la colonna e la direzione correnti (es. rende nera l'ultima freccia cliccata).

- `login.js`
  - `handleLoginResponse`: Gestisce la risposta del server alla richiesta di login. Reindirizza alla home page corretta (docente o studente) in base al tipo di utente loggato.

- `studentHome.js`: Gestisce la logica della home page dello studente.
  - *CoursesList*
    - `show`: Richiede al server la lista dei corsi a cui è iscritto lo studente loggato.
    - `update`: Inserisce la lista dei corsi nelle opzioni del primo form (selezione corso).
  - *DatesList*
    - `show`: Inserisce la lista delle date degli appelli a cui lo studente è iscritto (se presenti) nel secondo form (selezione data appello).
  - *MarkTable*
    - `show`: Gestisce la visualizzazione dell'esito dell'appello relativo al corso e alla data selezionati.
    - `createRejectButton`: Crea il bottone per permettere allo studente di rifiutare la valutazione, se applicabile.
  - *RejectDialog*
    - `show`: Gestisce la visualizzazione del dialogo di conferma per il rifiuto del voto.
    - `handleCancel`: Gestisce l'annullamento dell'operazione di rifiuto.
    - `handleReject`: Gestisce la conferma definitiva del rifiuto del voto.
- `dragAndDrop.js`: Contiene funzioni ausiliarie per la gestione di funzionalità di drag and drop.

8 Filtri

Per evitare di effettuare controlli di autorizzazione in ogni metodo di ogni servlet, sono stati utilizzati nel progetto due filtri, uno per ciascun ruolo che un utente può avere.

8.1 ProfessorAuthFilter

Questo filtro si occupa di assicurarsi che la sessione sia ancora valida, e che l'utente che sta cercando di chiamare il metodo della servlet sia autorizzato a farlo. Per questo motivo, è stato mappato nel web.xml su tutte le servlet relative ad azioni svolte da un utente di tipo professore. Il controllo avviene con il parametro "Corso di laurea" dell'userBean salvato in sessione. "Docente" è un parametro univoco associato ad un utente professore; qualsiasi altra stringa rappresenta un corso di laurea di uno studente universitario.

8.2 StudentAuthFilter

L'unica differenza rispetto al filtro precedente è il controllo sull'utente: ora viene controllato che il parametro "Corso di laurea" sia diverso da "Docente". Il filtro è mappato su tutte le servlet che corrispondono ad azioni svolte da un utente di tipo studente.

9 Diagrammi Sequenziali

In questa sezione verranno inseriti i diagrammi sequenziali che descrivono le interazioni principali dell'utente con il sistema, mostrando il flusso di chiamate tra client, servlet, DAO e database per le operazioni chiave.
Si noti che nei diagrammi sono mantenute le operazioni svolte dai rispettivi filtri.

9.1 Versione HTML

9.1.1 Studente

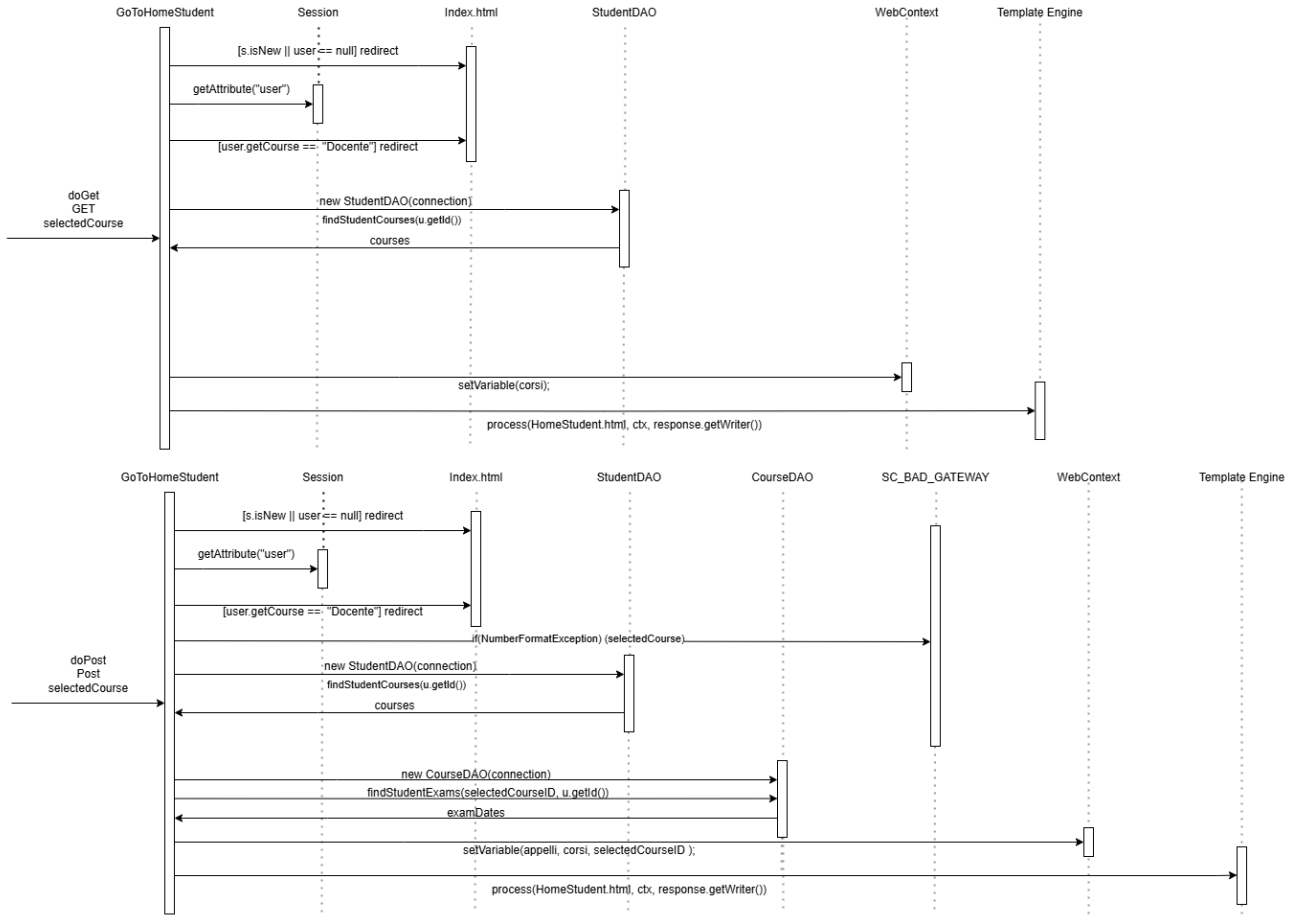


Figura 7: Diagramma Sequenziale GoToHomeStudent (GET & POST).

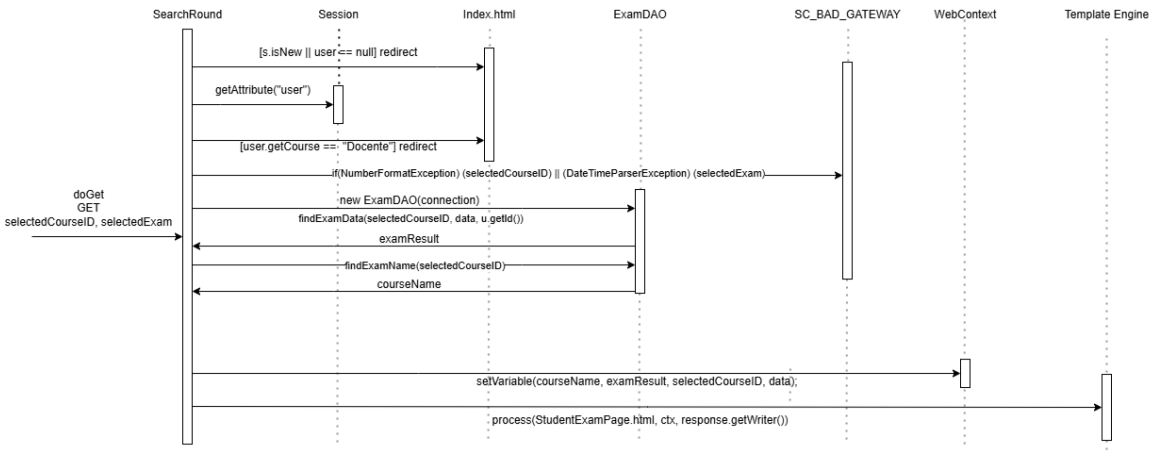


Figura 8: Diagramma Sequenziale SearchRound.

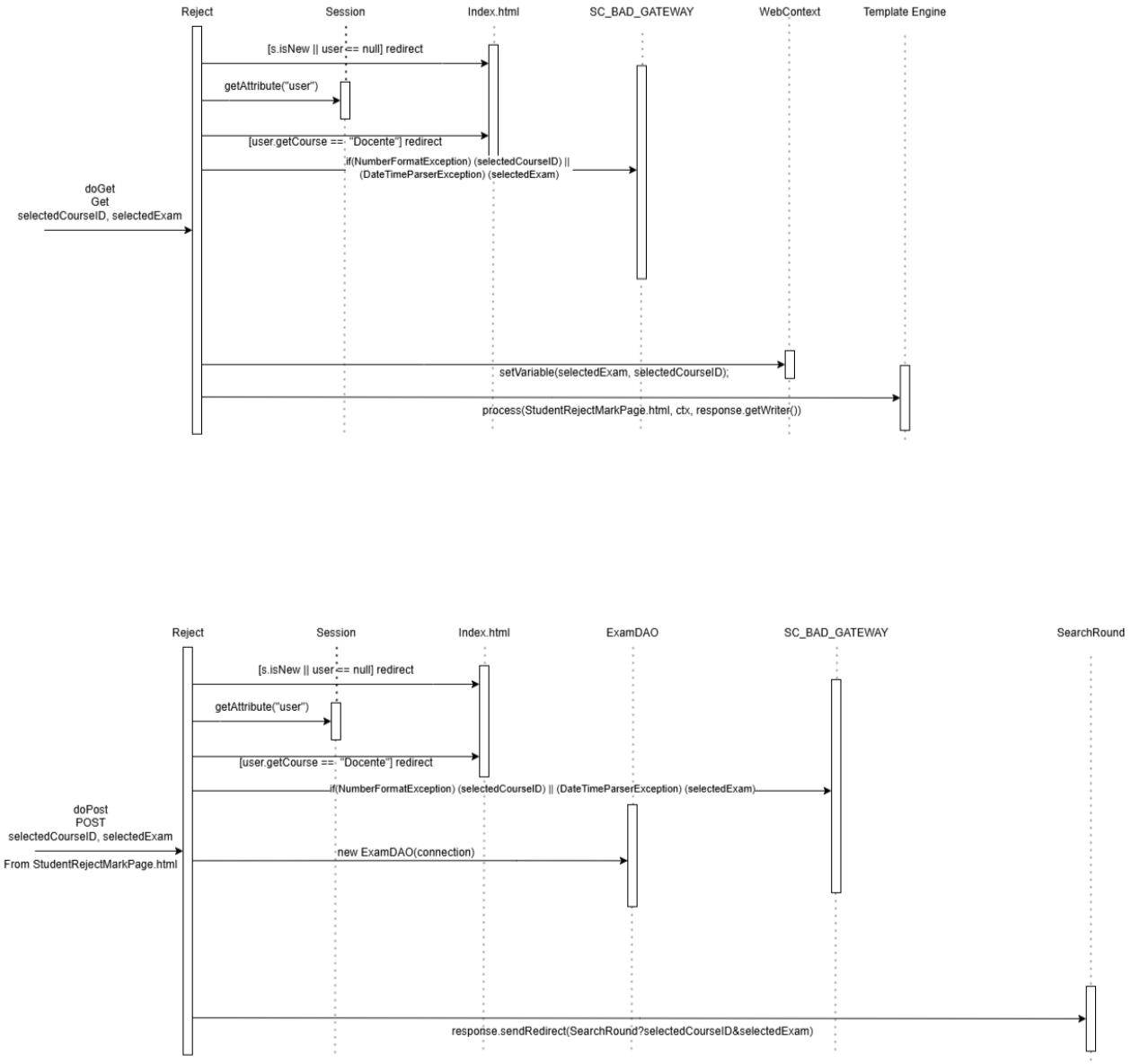


Figura 9: Diagramma Sequenziale Reject (GET & POST).

9.1.2 Professore

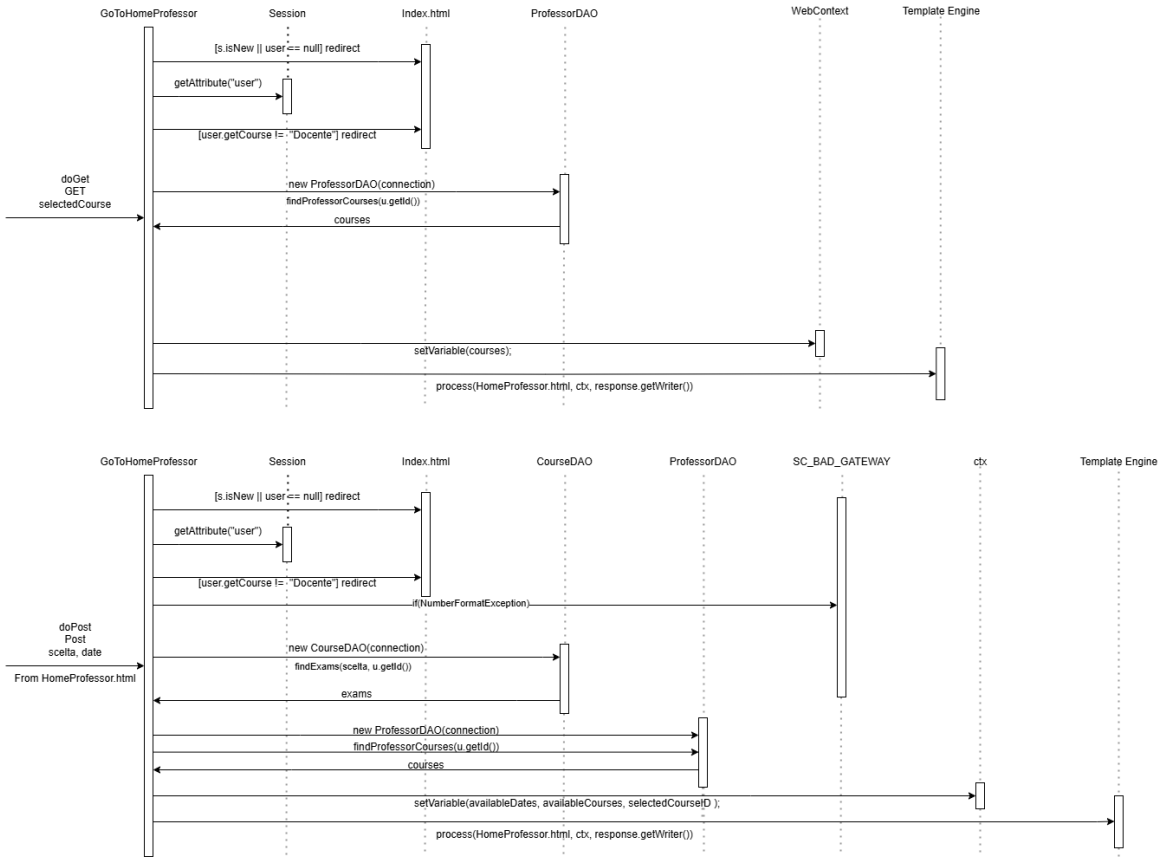


Figura 10: Diagramma Sequenziale GoToHomeProfessor (GET & POST).

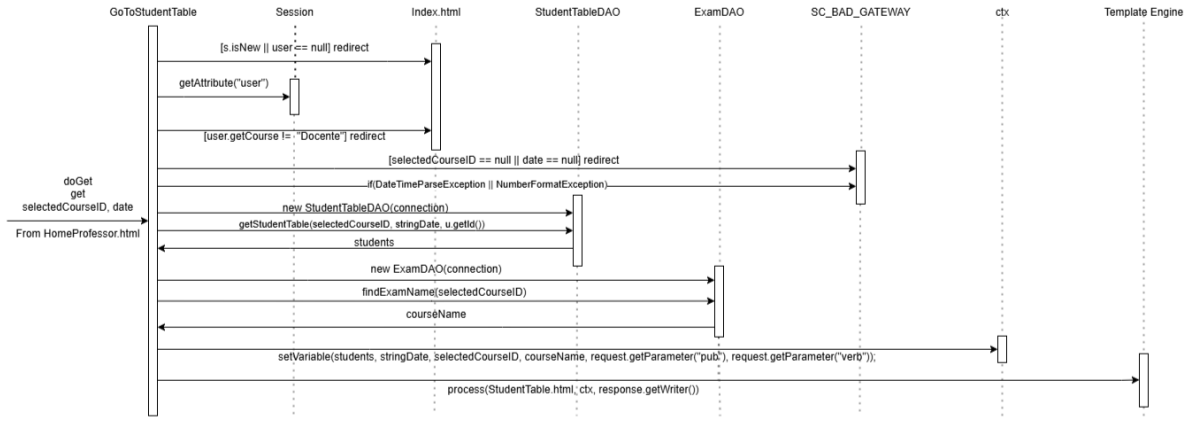


Figura 11: Diagramma Sequenziale GoToStudentTable.

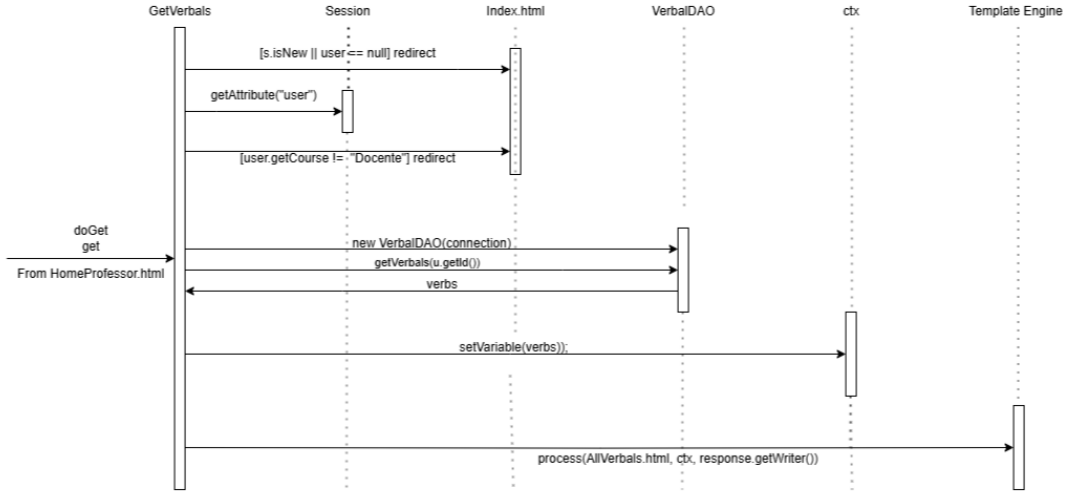


Figura 12: Diagramma Sequenziale GetVerbals.

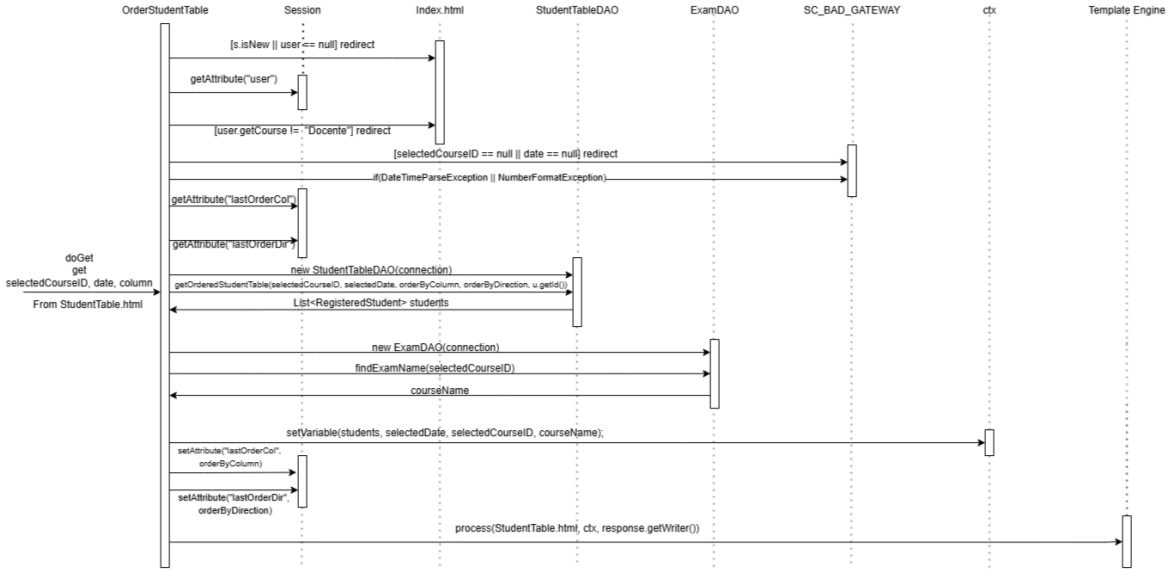


Figura 13: Diagramma Sequenziale OrderStudentTable.

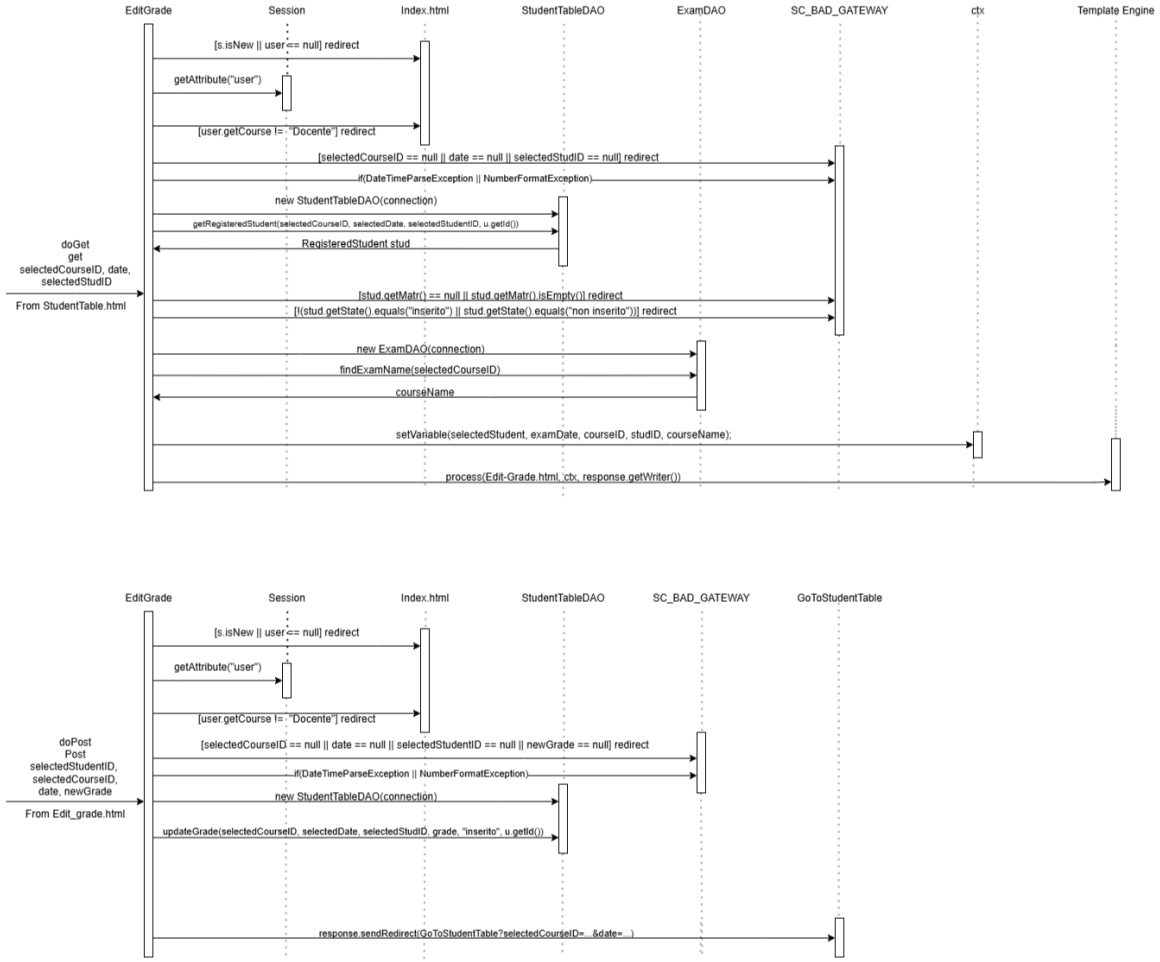


Figura 14: Diagramma Sequenziale EditGrade (GET & POST).

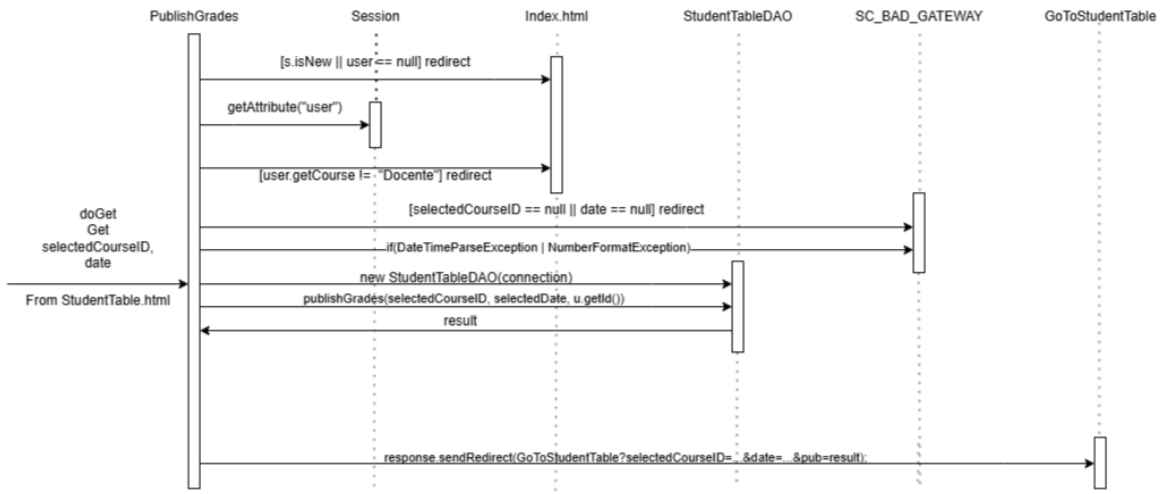


Figura 15: Diagramma Sequenziale PublishGrades.

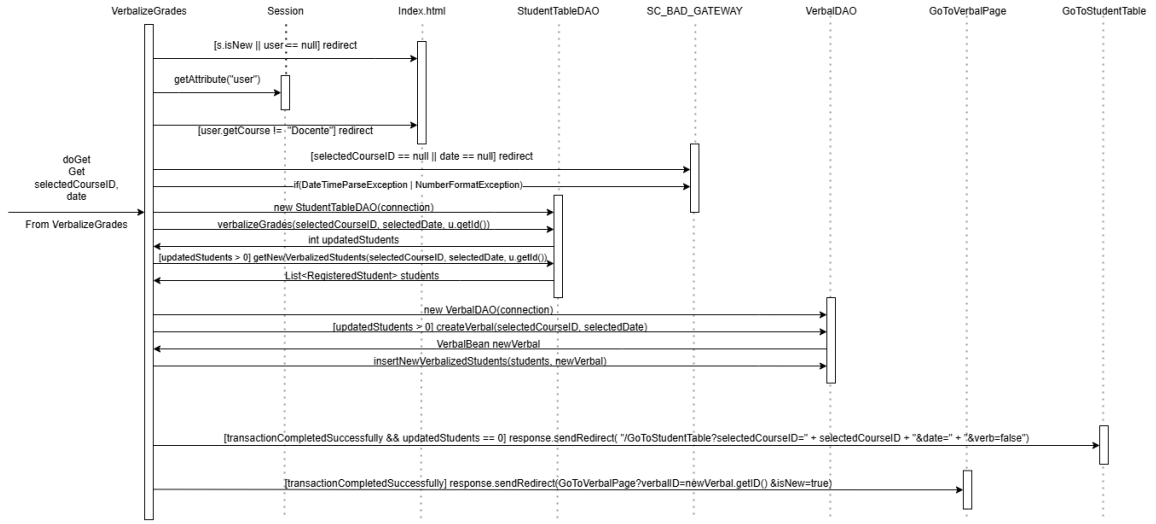


Figura 16: Diagramma Sequenziale VerbalizeGrades.

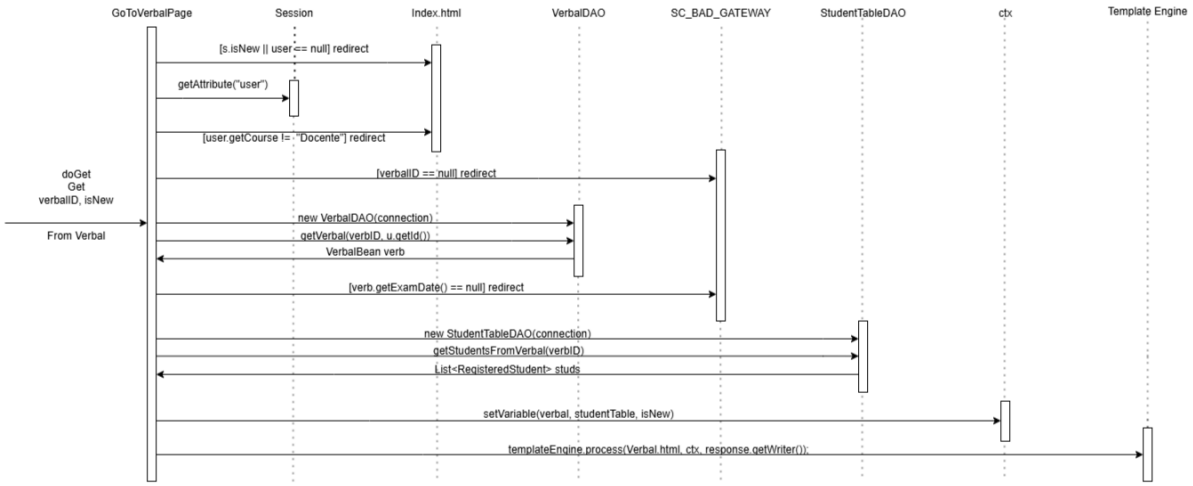


Figura 17: Diagramma Sequenziale GoToVerbalPage.

9.1.3 Login & Logout

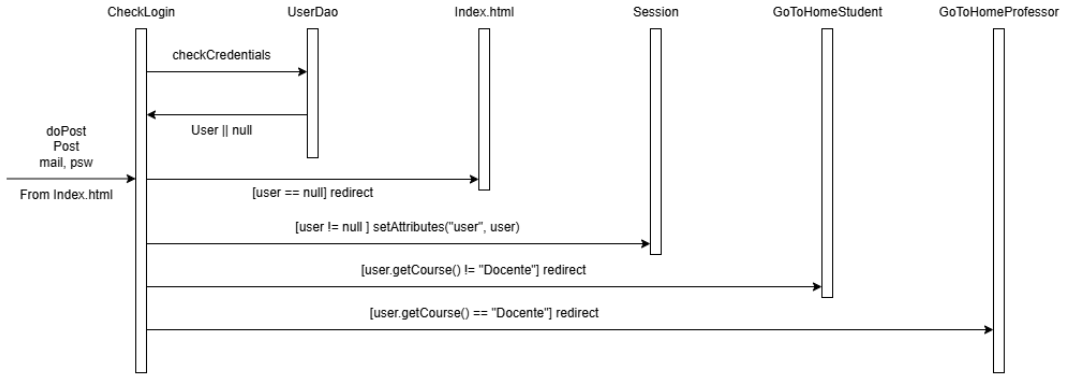


Figura 18: Diagramma Sequenziale Login.

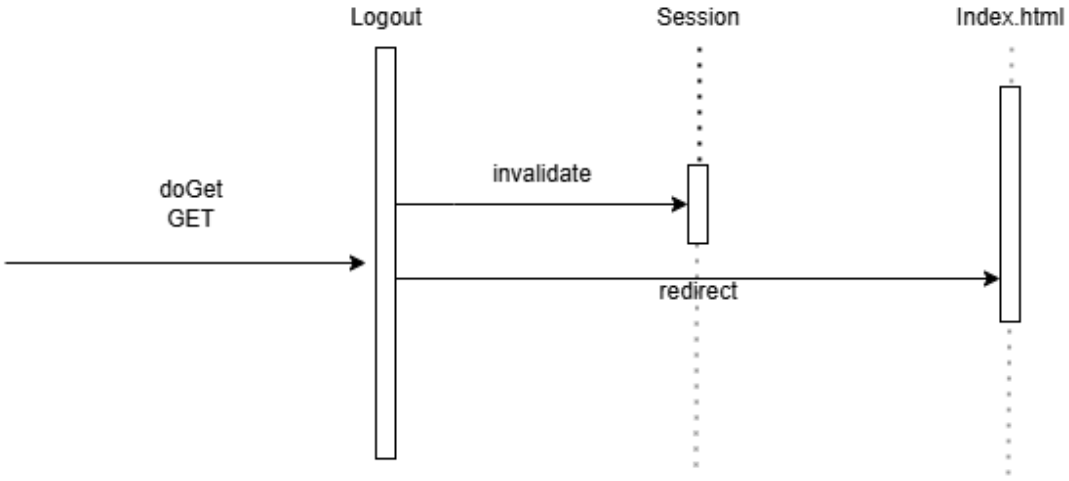


Figura 19: Diagramma Sequenziale Logout.

9.2 Versione RIA

9.2.1 Login

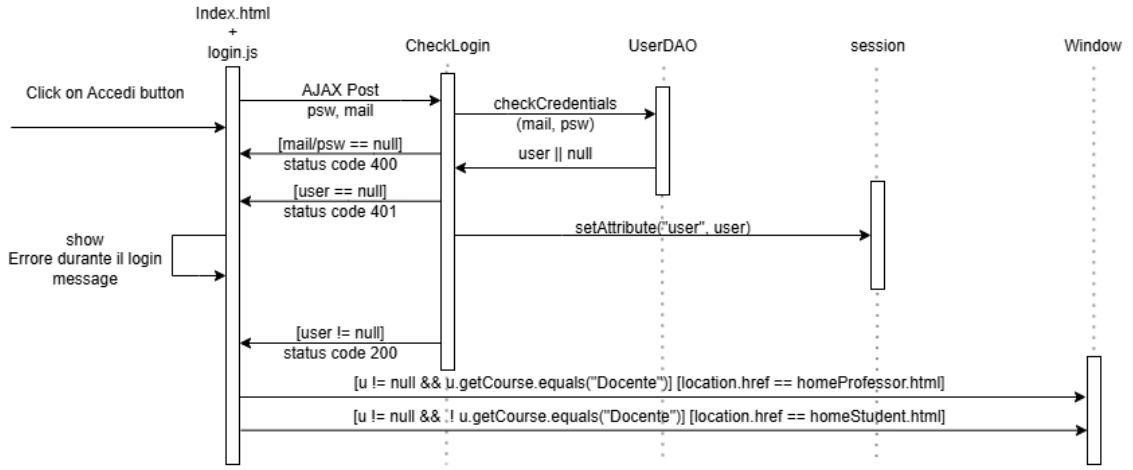


Figura 20: Diagramma Sequenziale Login.

9.2.2 Studente

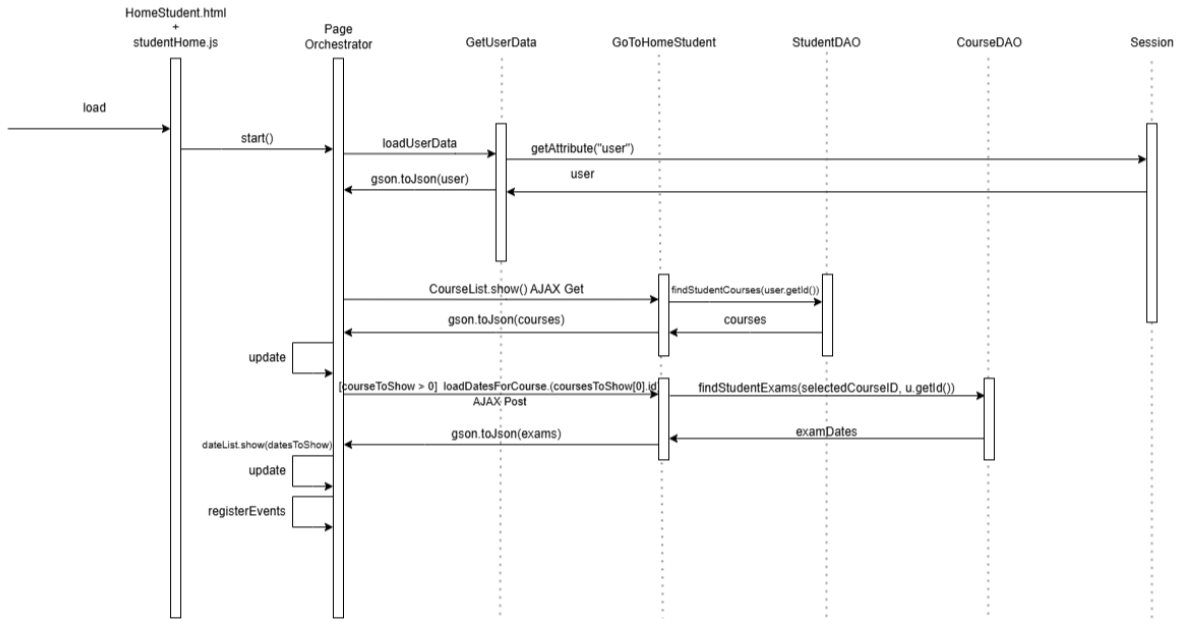


Figura 21: Diagramma Sequenziale caricamento pagina studente.

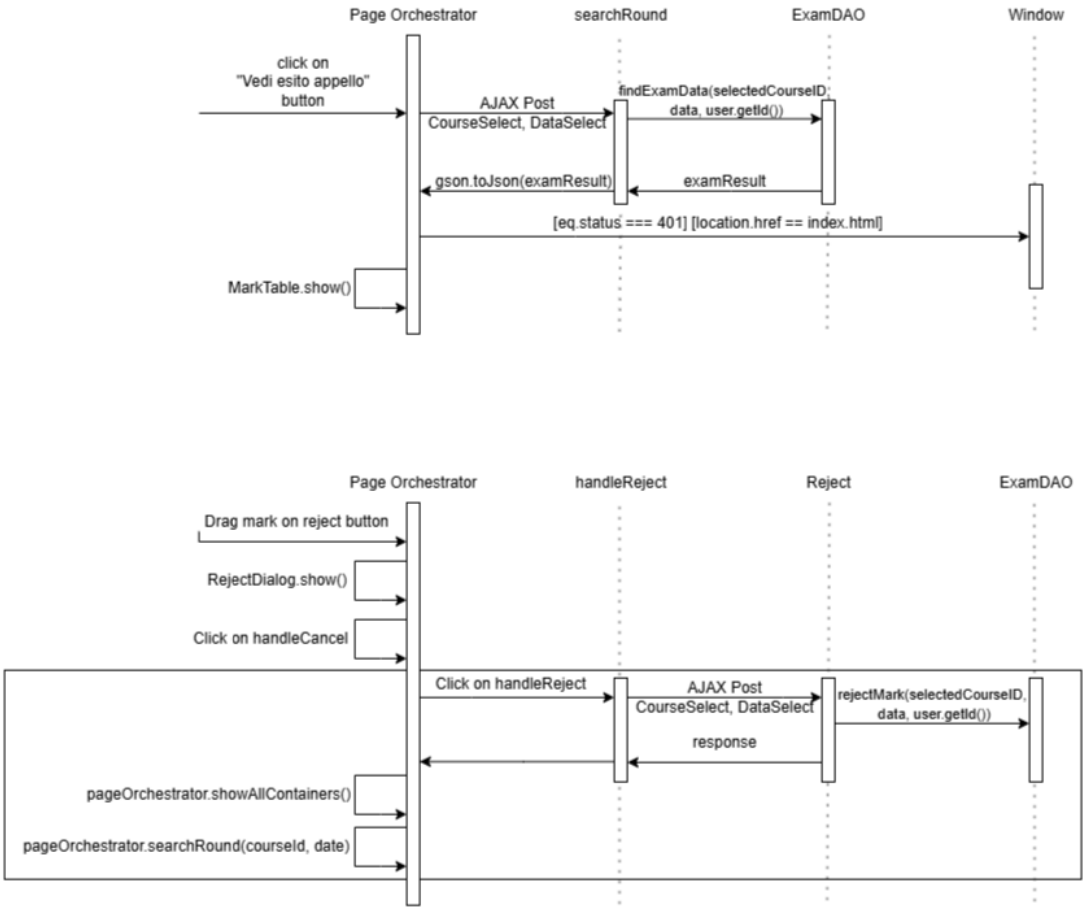


Figura 22: Diagramma Sequenziale click su "Vedi esito appello" e interazioni di rifiuto del voto.

9.2.3 Professore

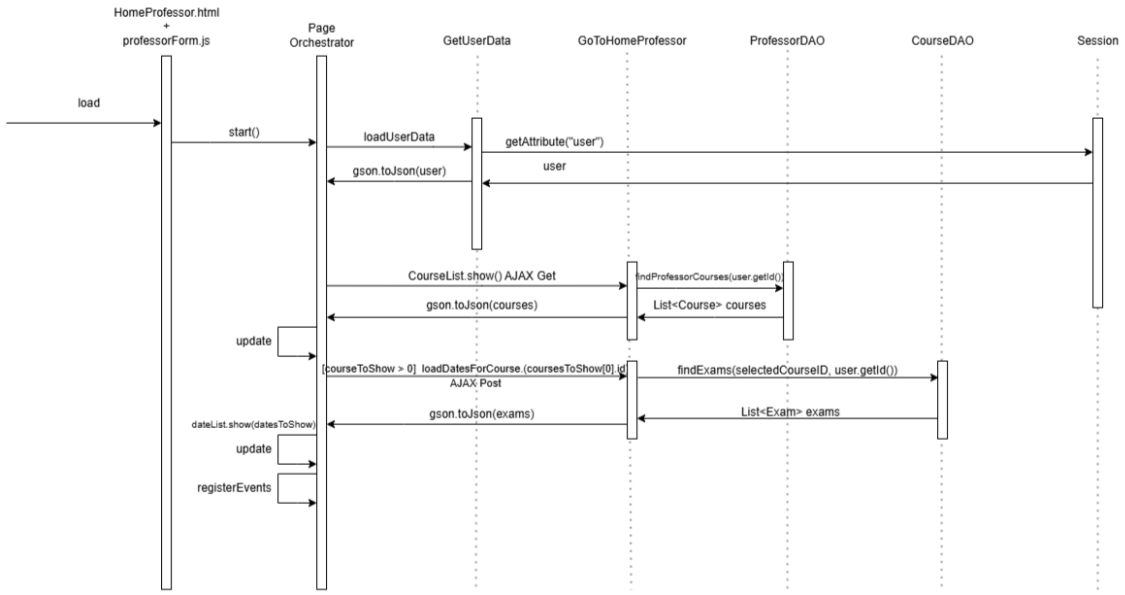


Figura 23: Diagramma Sequenziale caricamento pagina professore.

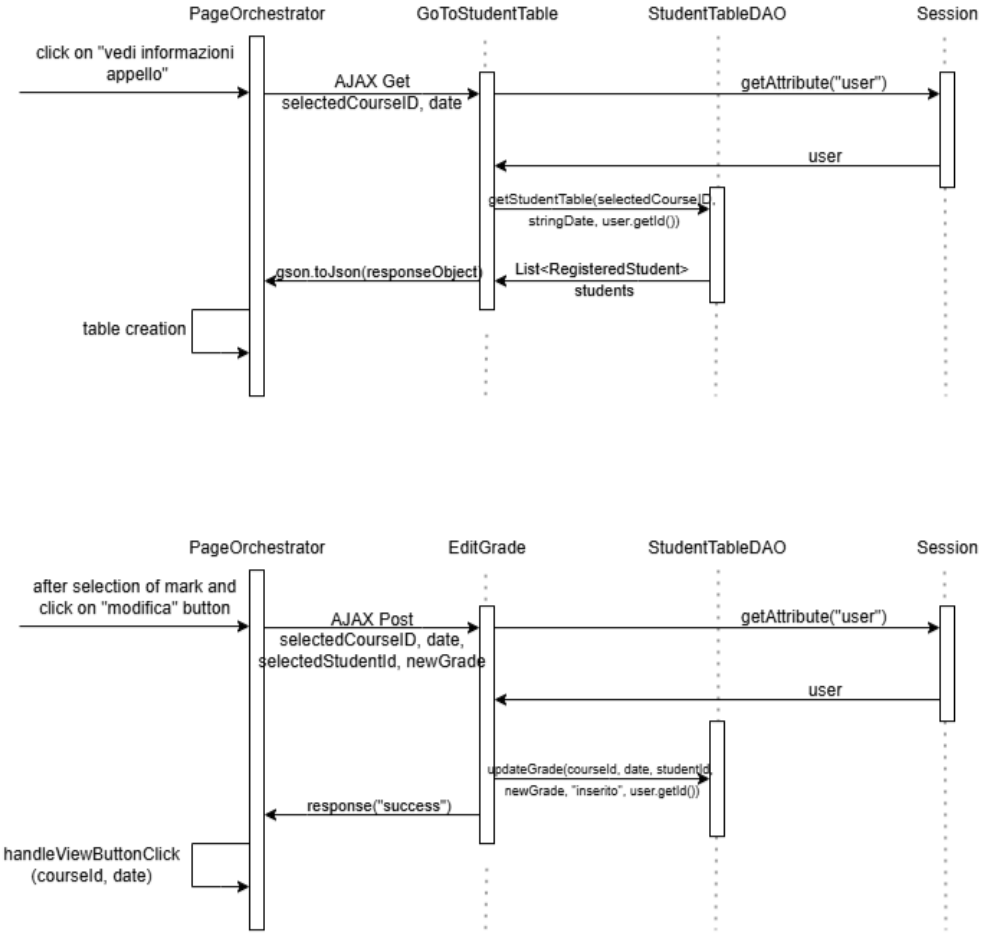


Figura 24: Diagramma Sequenziale click su "Vedi informazioni appello" e click su "Modifica" per l'inserimento di una valutazione.

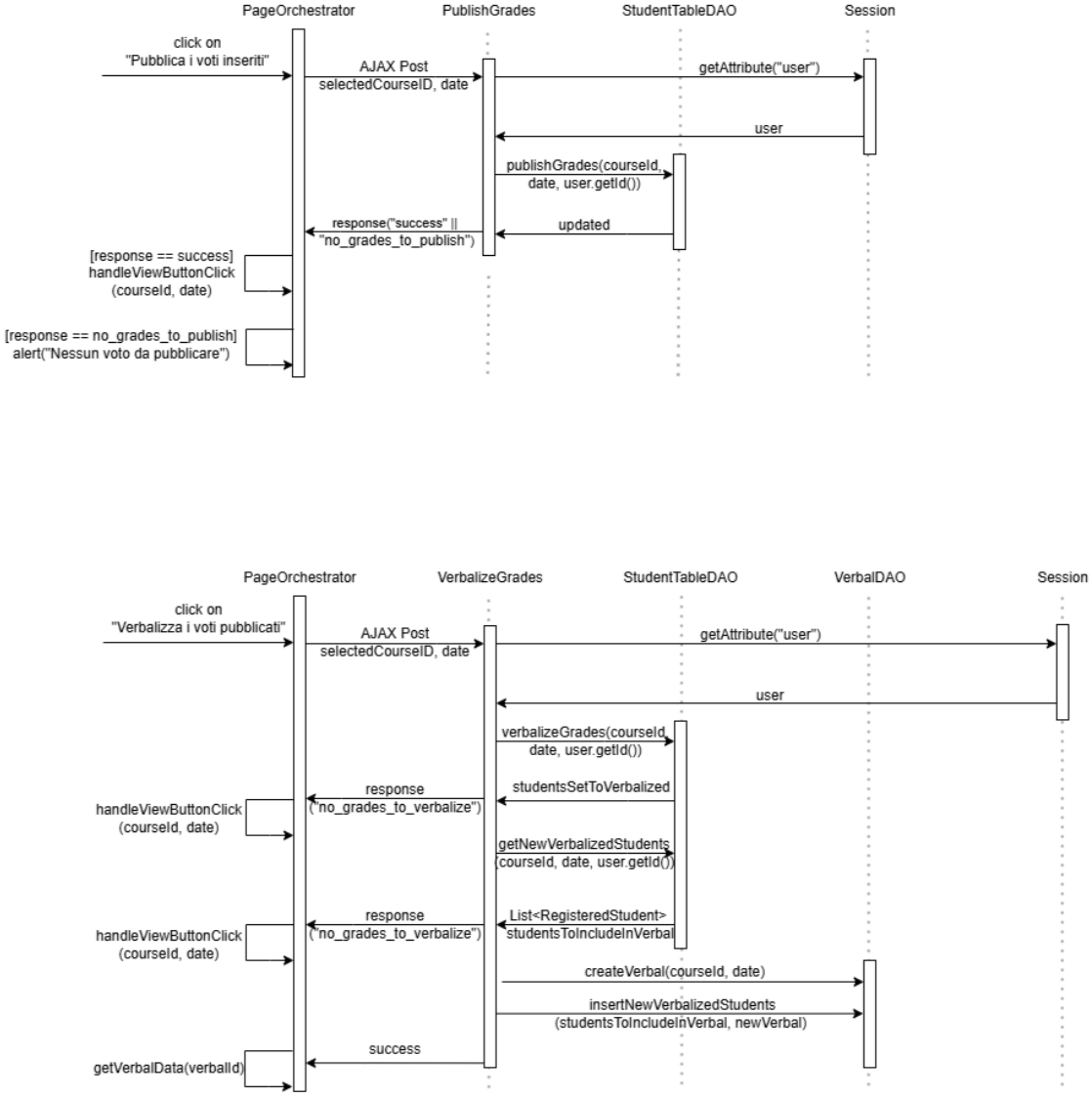


Figura 25: Diagramma Sequenziale pubblicazione e verbalizzazione voti, con conseguente creazione di verbale.

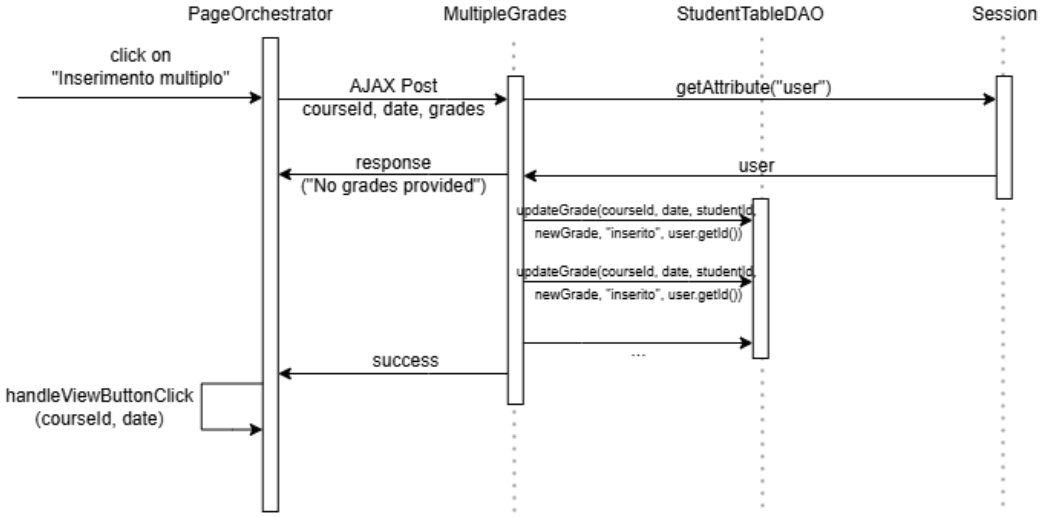


Figura 26: Diagramma Sequenziale inserimento di valutazioni multiple.

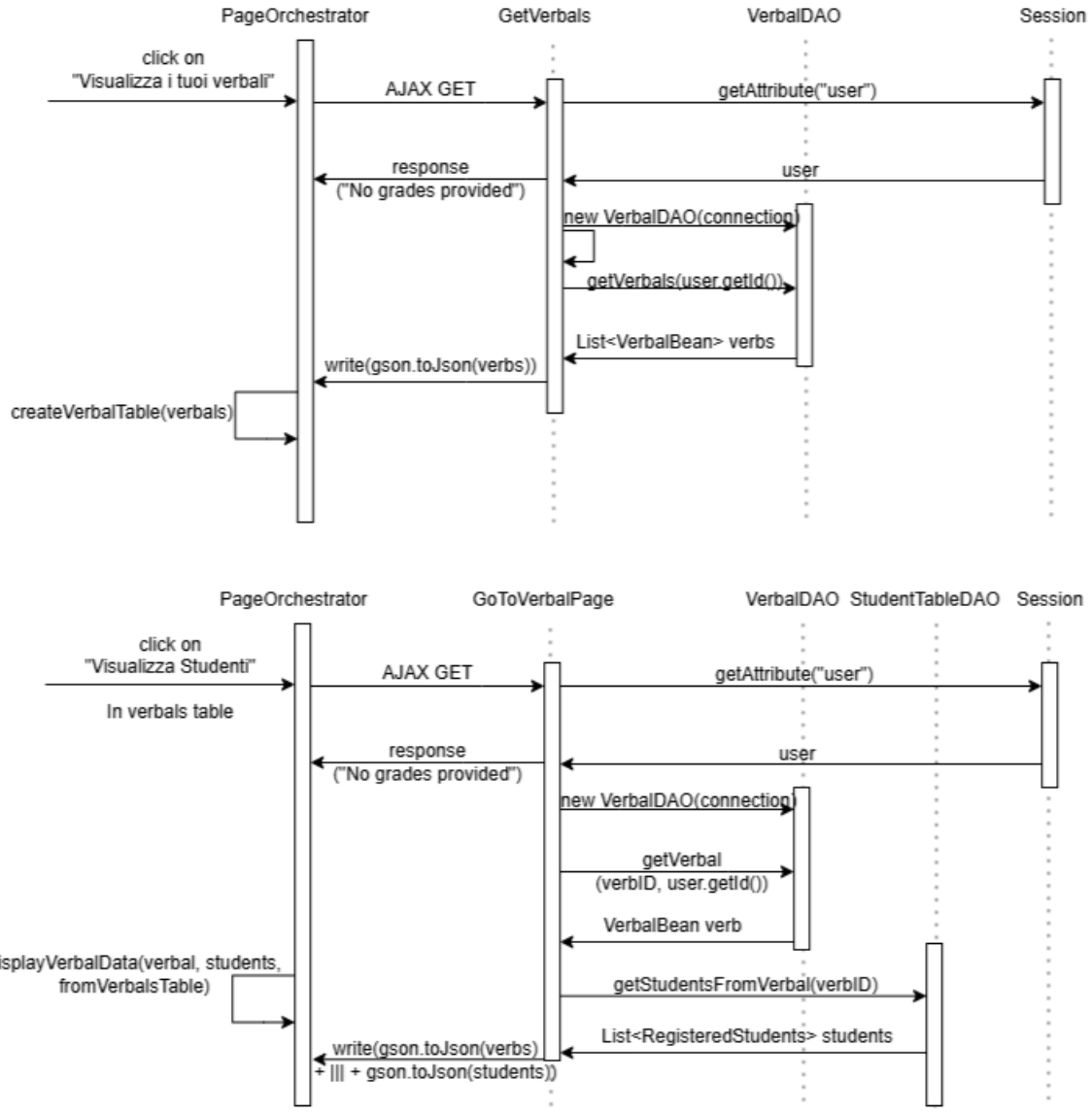


Figura 27: Diagramma Sequenziale visualizzazione dei verbali e delle informazioni di un verbale.