TIW 2022/23 - Esercizio 4

Simone Ponginibbio

June 12, 2023

Esercizio 4: verbalizzazione degli esami

Versione Pure HTML:

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 in modo alfabetico decrescente e poi una data d'appello del corso scelto selezionata 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: ¡vuoto¿, 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.

Versione RIA:

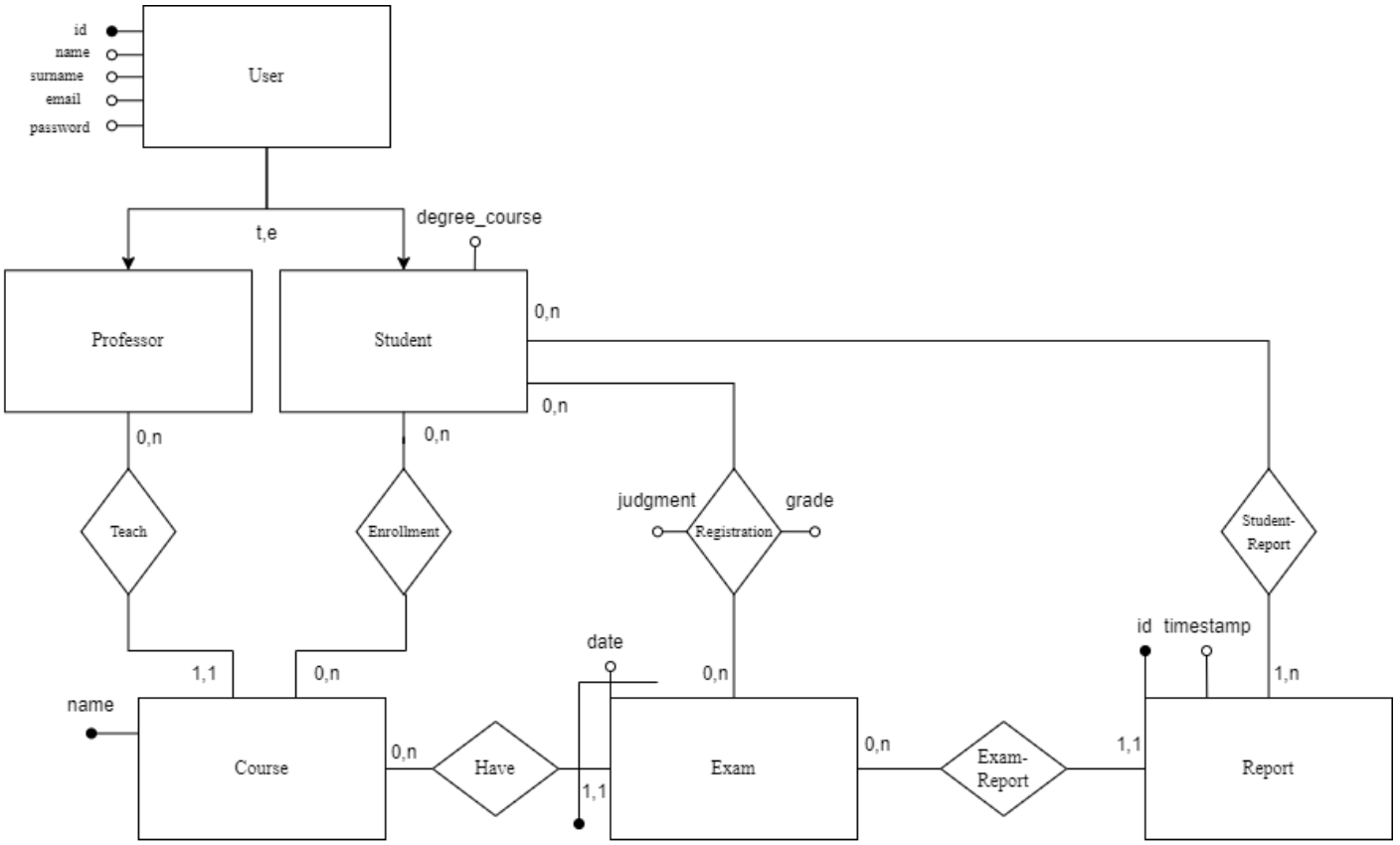
Si realizzi un’applicazione client server web che modifica le specifiche precedenti come segue:

- Dopo il login dell’utente, l’intera applicazione è realizzata con un’unica pagina per il docente e un’unica pagina per lo studente.
- Ogni interazione dell’utente è gestita senza ricaricare completamente la pagina, ma produce l’invocazione asincrona del server e l’eventuale modifica del contenuto da aggiornare a seguito dell’evento.
- La funzione di riordino della tabella degli iscritti è realizzata a lato client.
- Alla tabella degli iscritti è associato un bottone INSERIMENTO MULTIPLO che provoca la comparsa di una pagina modale con tutte e sole le righe nello stato “non inserito” associate a un campo di input. Il docente può inserire un voto per un insieme delle righe e premere un bottone INVIA che comporta l’invio al server dei voti, il cambio di stato delle righe coinvolte, la chiusura della finestra modale e l’aggiornamento della tabella degli iscritti.

Entintà, Attributi, Relazioni

Pagine, Componenti, Eventi, Azioni

Database Design:



Database Schema:

```
CREATE TABLE `dbexams`.`user` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  `surname` VARCHAR(45) NOT NULL,
  `email` VARCHAR(320) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `user_email_unique` (`email`)
);

CREATE TABLE `dbexams`.`professor` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  PRIMARY KEY (`id`),
  CONSTRAINT `user_id`
    FOREIGN KEY (`id`)
    REFERENCES `dbexams`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`student` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `degree course` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  CONSTRAINT `user_student_id`
    FOREIGN KEY (`id`)
    REFERENCES `dbexams`.`user` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);

CREATE TABLE `dbexams`.`course` (
  `name` VARCHAR(45) NOT NULL,
  `professor_id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`name`),
  CONSTRAINT `professor_id`
    FOREIGN KEY (`professor_id`)
    REFERENCES `dbexams`.`professor` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`exam` (
  `course_name` VARCHAR(45) NOT NULL,
  `date` DATE NOT NULL,
  PRIMARY KEY (`course_name`, `date`),
  CONSTRAINT `course_name`
    FOREIGN KEY (`course_name`)
    REFERENCES `dbexams`.`course` (`name`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`report` (
  `id` INT UNSIGNED NOT NULL AUTO_INCREMENT,
  `course_name` VARCHAR(45) NOT NULL,
  `exam_date` DATE NOT NULL,
  `timestamp` TIMESTAMP NOT NULL DEFAULT current_timestamp,
  PRIMARY KEY (`id`),
  CONSTRAINT `exam_report`
    FOREIGN KEY (`course_name` , `exam_date`)
    REFERENCES `dbexams`.`exam` (`course_name` , `date`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`student_report` (
  `student_id` INT UNSIGNED NOT NULL,
  `report_id` INT UNSIGNED NOT NULL,
  PRIMARY KEY (`student_id`, `report_id`),
  CONSTRAINT `student_report_id`
    FOREIGN KEY (`student_id`)
    REFERENCES `dbexams`.`student` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `report_id`
    FOREIGN KEY (`report_id`)
    REFERENCES `dbexams`.`report` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`registration` (
  `course_name` VARCHAR(45) NOT NULL,
  `exam_date` DATE NOT NULL,
  `student_id` INT UNSIGNED NOT NULL,
  `grade` VARCHAR(45) NULL,
  `judgment` VARCHAR(45) NOT NULL DEFAULT 'uninsered',
  CHECK (`grade` IN ('', 'absent', 'failed', 'tried again', '18', '19', '20', '21', '22', '23', '24', '25', '26', '27', '28', '29', '30', '30L')),
  CHECK (`judgment` IN ('uninsered', 'insered', 'published', 'declined', 'verbalised')),
  PRIMARY KEY (`course_name`, `exam_date`, `student_id`),
  CONSTRAINT `exam_registration`
    FOREIGN KEY (`course_name` , `exam_date`)
    REFERENCES `dbexams`.`exam` (`course_name` , `date`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE,
  CONSTRAINT `student_id`
    FOREIGN KEY (`student_id`)
    REFERENCES `dbexams`.`student` (`id`)
    ON DELETE NO ACTION
    ON UPDATE CASCADE
);
```

```
CREATE TABLE `dbexams`.`enrollment` (
  `student_id` INT UNSIGNED NOT NULL,
  `course_name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`student_id`, `course_name`),
  CONSTRAINT `student_id_enrollment`
    FOREIGN KEY (`student_id`)
    REFERENCES `dbexams`.`student` (`id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `course_name_enrollment`
    FOREIGN KEY (`course_name`)
    REFERENCES `dbexams`.`course` (`name`)
    ON DELETE CASCADE
    ON UPDATE CASCADE
);
```

Components (Pure HTML):

- Model Object (beans):
  - User
  - Student
  - Professor
  - Course
  - Exam
  - Enrollment
  - Registration
  - Report
  - StudentReport
- Data Access Objects (DAO):
  - UserDAO
  - StudentDAO
  - ProfessorDAO
  - CourseDAO
  - EnrollmentDAO
  - RegistrationDAO
  - ReportDAO
  - StudentReportDAO
- Controllers (servlets):
  - Login
  - Logout
  - RegisterStudent
  - RegisterProfessor
  - GoToLoginPage
  - GoToStudentRegisterPage
  - GoToProfessorRegisterPage
  - GoToHome
  - GoToExamDates
  - GoToExamOutcome
  - GoToDeclineGrade
  - GoToExamRegistrations
  - GoToModifyGrade
  - GoToInsertGrade
  - GoToPublishGrade
  - GoToVerbaliseGrade

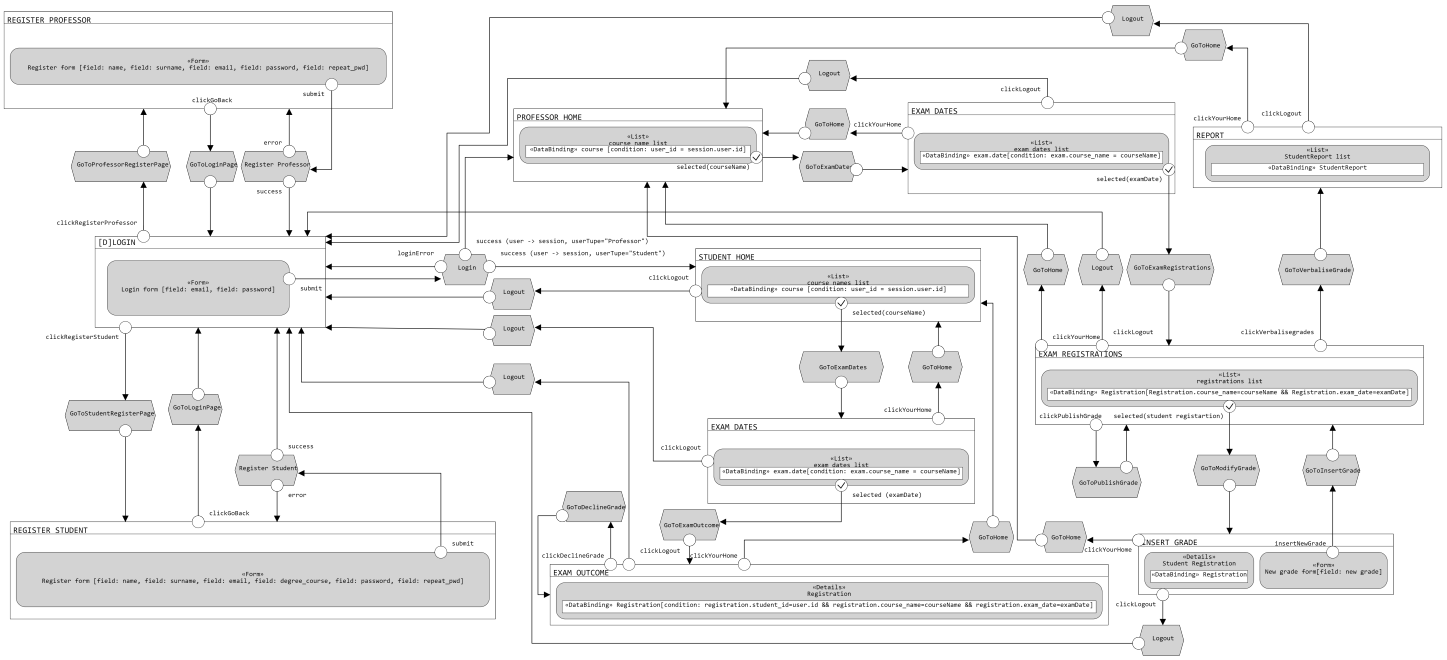
- Filters:
  - CheckLoggedUser
  - CheckNotLoggerUser
- Views (templates):
  - login.html
  - registerStudent.html
  - registerProfessor.html
  - studentHome.html
  - professorHome.html
  - infoPane.html
  - studentExamDates.html
  - professorExamDates.html
  - examOutcome.html
  - examRegistrations.html
  - momdifyGrade.html
  - report.html
  - error.html

Components (RIA):

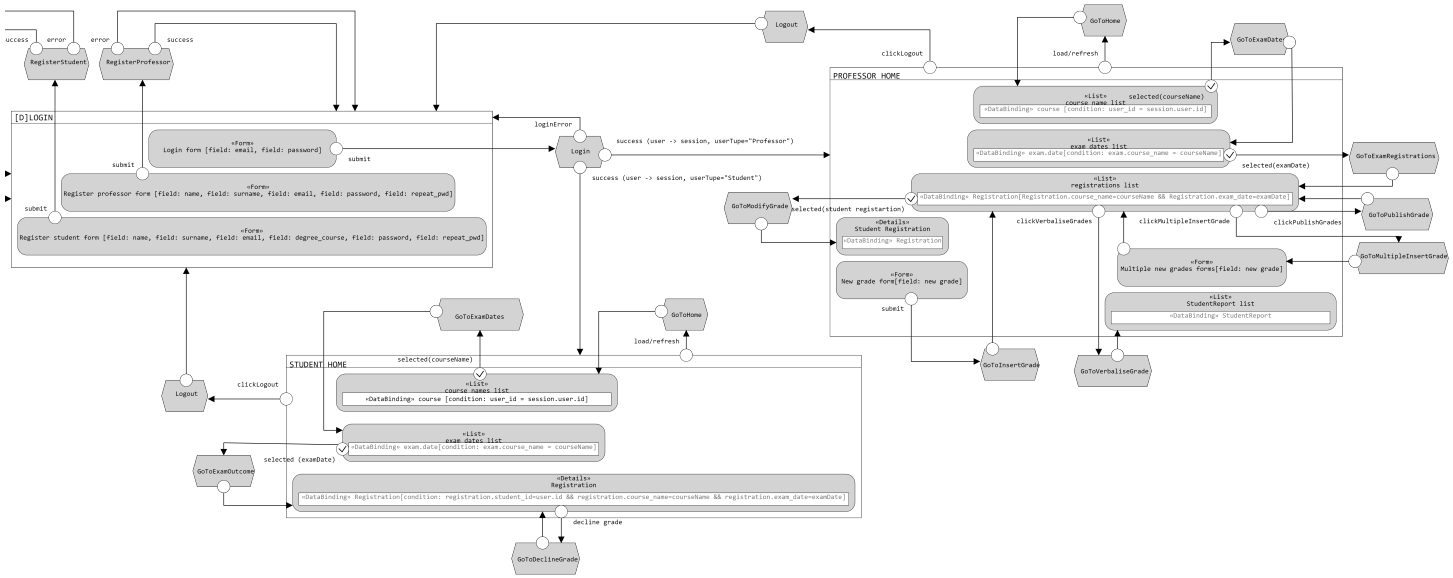
- Model Object (beans):
  - User
  - Student
  - Professor
  - Course
  - Exam
  - Enrollment
  - Registration
  - Report
  - StudentReport
- Data Access Objects (DAO):
  - UserDAO
  - StudentDAO
  - ProfessorDAO
  - CourseDAO
  - EnrollmentDAO
  - RegistrationDAO
  - ReportDAO
  - StudentReportDAO
- Controllers (servlets):
  - Login
  - Logout
  - RegisterStudent
  - RegisterProfessor
  - GoToHome
  - GoToExamDates
  - GoToExamOutcome
  - GoToDeclineGrade
  - GoToExamRegistrations
  - GoToModifyGrade
  - GoToInsertGrade
  - GoToMultipleInsertGrade
  - GoToPublishGrade
  - GoToVerbaliseGrade

- Filters:
  - CheckLoggedUser
  - CheckNotLoggerUser
- Views (templates):
  - login.html
  - studentHome.html
  - professorHome.html
- Packets:
  - PacketExam
  - PacketReport

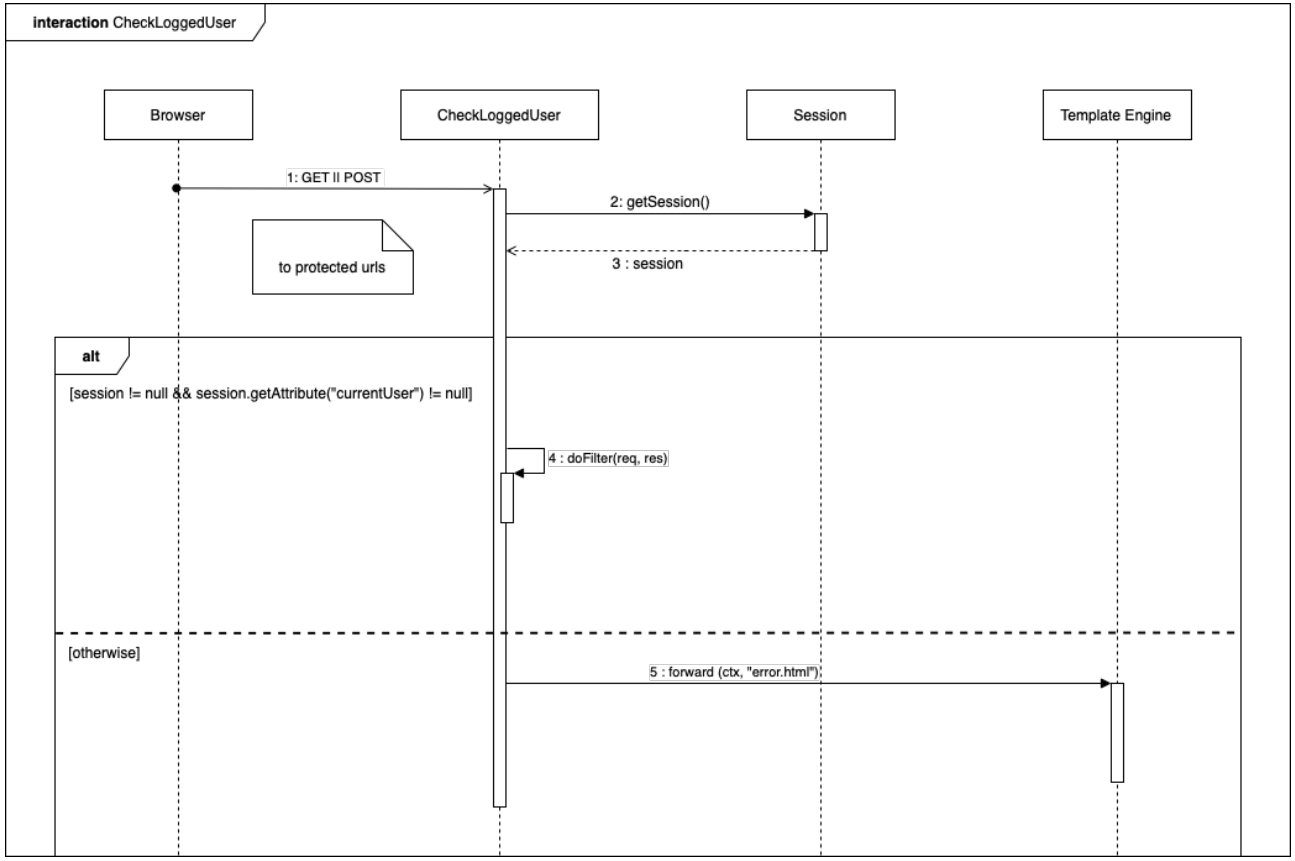
Design Applicativo (IFML) Pure HTML:



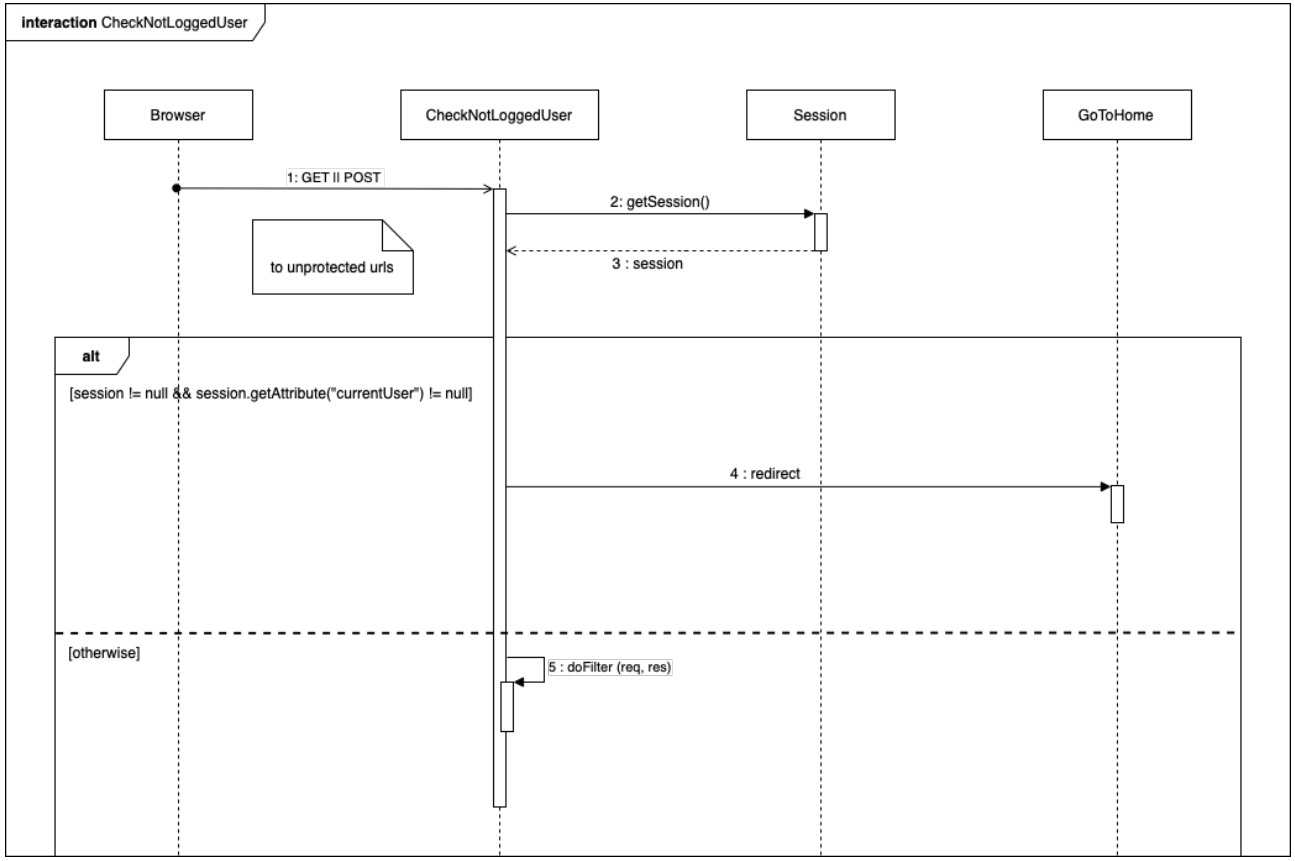
Design Applicativo (IFML) RIA:



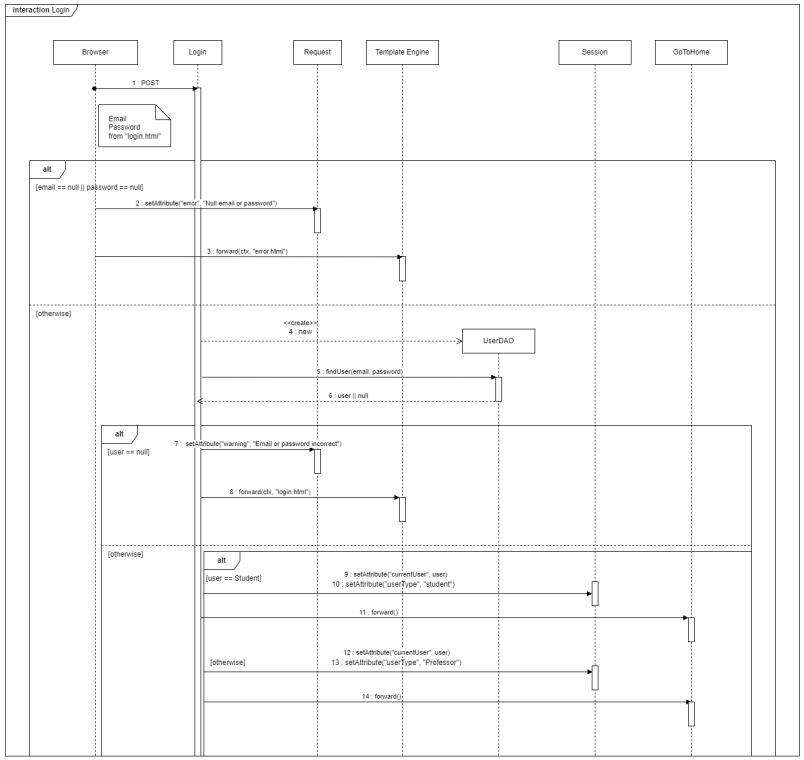
CheckLoggedUser:



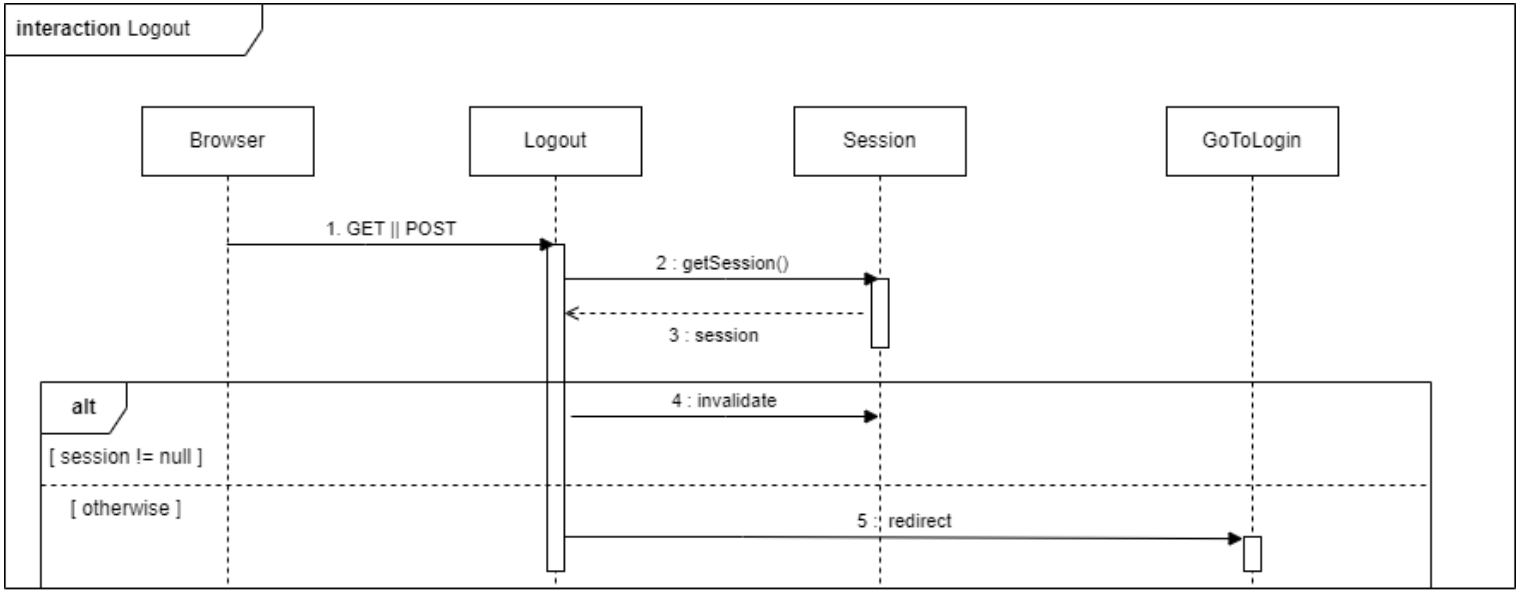
CheckNotLoggedUser:



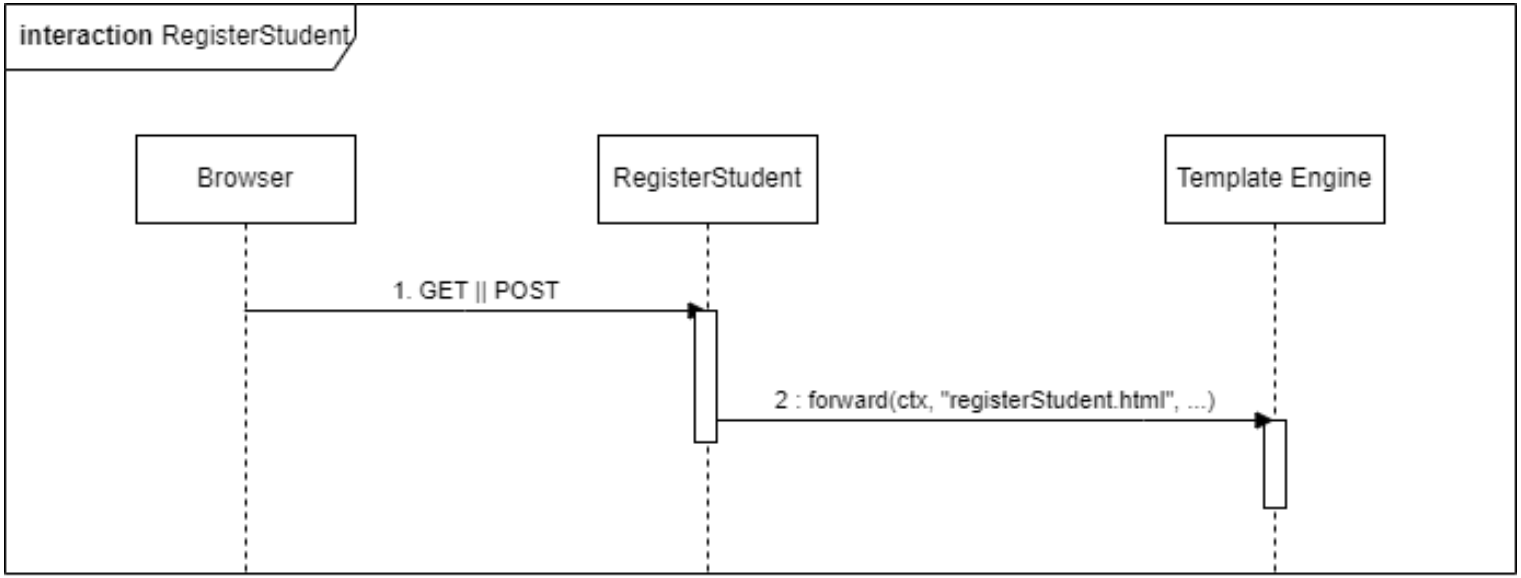
Login:


interaction Login
Browser
Login
Request
Template Engine
Session
GoToHome
1 : POST
Email
Password
from "login.html"
alt
[email == null || password == null]
2 : setAttribute("error", "Null email or password")
3 : forward(ctx, "error.html")
[otherwise]
<<create>>
4 : new
UserDAO
5 : findUser(email, password)
6 : user || null
alt
[user == null]
7 : setAttribute("warning", "Email or password incorrect")
8 : forward(ctx, "login.html")
[otherwise]
alt
[user == Student]
9 : setAttribute("currentUser", user)
10 : setAttribute("userType", "student")
11 : forward()
[otherwise]
12 : setAttribute("currentUser", user)
13 : setAttribute("userType", "Professor")
14 : forward()

Logout:


interaction Logout
Browser
Logout
Session
GoToLogin
1. GET || POST
2 : getSession()
3 : session
alt
[session != null]
[otherwise]
4 : invalidate
5 : redirect

RegisterStudent:



RegisterProfessor: