Gibellini Federico - 10669617
Montemurro Elena - 10627677

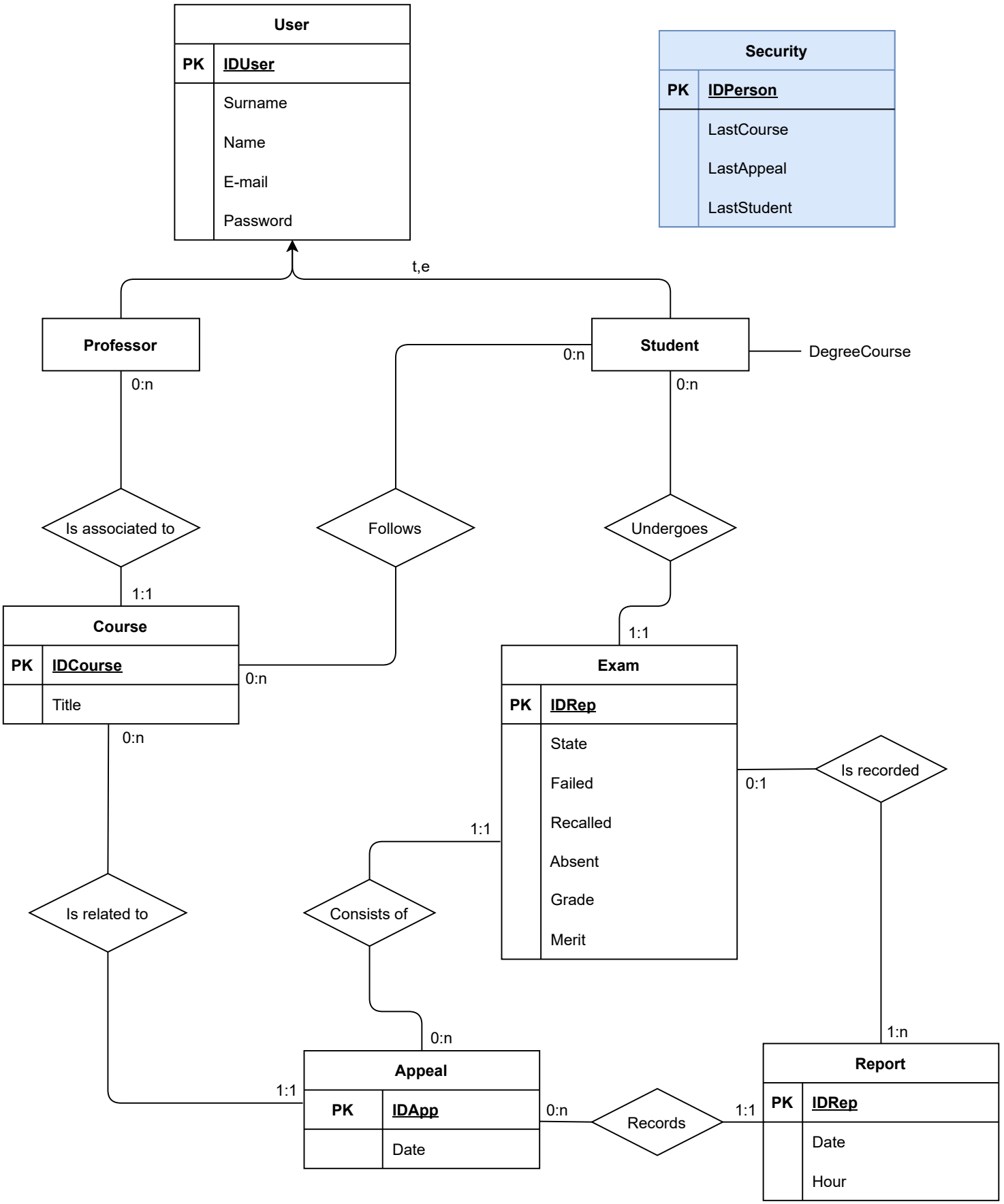
Documentazione relativa al progetto di Tecnologie Informatiche per il Web, anno accademico 2020-2021

Analisi della specifica – traccia 4: verbalizzazione di esami

Analisi dei dati

Nella presente sezione sono evidenziati nel testo della specifica per la versione pure HTML i contenuti per la creazione del database; sono indicati in rosso le entità, in verde gli attributi, in blu le 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 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. Il voto può non essere ancora definito. Lo stato 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. 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 selezionare il voto. L'invio della form provoca la modifica o l'inserimento del voto. Inizialmente le righe sono nello stato "non inserito". L'inserimento e le successive eventuali modifiche portano la riga nello stato "inserito". Alla tabella è associato un bottone PUBBLICA che comporta la pubblicazione delle righe con lo stato INSERITO. La pubblicazione rende il voto non più modificabile dal docente e visibile allo studente e cambia lo stato 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. 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 della riga dello studente per quell'appello anche 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.



PROFESSOR (ID_PROF, Name, Surname, E-mail, Password)
STUDENT (ID_STUD, Name, Surname, E-mail, Password, Degree_Course)
COURSE (ID_COURSE, Id_Professor, Title)
FOLLOWINGS (ID_COURSE, ID_STUD)
APPEAL (ID_APP, Id_Course, Date)
EXAM (ID_STUDENT, ID_APP, Failed, Recalled, Absent, Grade*, Merit, State, Id_Report)
REPORT (ID_REP, Id_App, Date, Hour)

SECURITY (ID_PERSON, LastCourse, LastAppeal, LastStudent)

No real need for "Security" Table; introduced only for security reasons

Schema del database

Tabella "professor"

```
CREATE TABLE `professor` (
   `id_professor` int NOT NULL AUTO_INCREMENT,
   `name` varchar(45) NOT NULL,
   `surname` varchar(45) NOT NULL,
   `email` varchar(45) NOT NULL,
   `password` varchar(45) NOT NULL,
   PRIMARY KEY (`id_professor`)
)
```

Tabella "student"

```
CREATE TABLE `student` (
   `id_student` int NOT NULL AUTO_INCREMENT,
   `name` varchar(45) NOT NULL,
   `surname` varchar(45) NOT NULL,
   `email` varchar(45) NOT NULL,
   `password` varchar(45) NOT NULL,
   `degree_course` varchar(45) NOT NULL,
   PRIMARY KEY (`id_student`)
)
```

Tabella "course"

```
CREATE TABLE `course` (
   `id_course` int NOT NULL AUTO_INCREMENT,
   `id_professor` int NOT NULL,
   `title` varchar(45) NOT NULL,
   PRIMARY KEY (`id_course`),
   KEY `id_professor_idx` (`id_professor`),
   CONSTRAINT `id_professor` FOREIGN KEY (`id_professor`) REFERENCES `professor` (`id_professor`) ON
         UPDATE CASCADE
)
```

Tabella "followings"

```
CREATE TABLE `followings` (
   `id_student` int NOT NULL,
   `id_course` int NOT NULL,
   PRIMARY KEY (`id_student`,`id_course`),
   KEY `course_idx` (`id_course`),
   CONSTRAINT `course_followed` FOREIGN KEY (`id_course`) REFERENCES `course` (`id_course`) ON UPDATE
         CASCADE,
   CONSTRAINT `student_courses` FOREIGN KEY (`id_student`) REFERENCES `student` (`id_student`) ON
         UPDATE CASCADE
)
```

Tabella “appeal”

```
CREATE TABLE `appeal` (
  `id_appeal` int NOT NULL AUTO_INCREMENT,
  `id_course` int NOT NULL,
  `date` date NOT NULL,
  PRIMARY KEY (`id_appeal`),
  KEY `id_course_idx` (`id_course`),
  CONSTRAINT `id_course` FOREIGN KEY (`id_course`) REFERENCES `course` (`id_course`) ON UPDATE
      CASCADE
)
```

Tabella “exam”

```
CREATE TABLE `exam` (
  `id_appeal` int NOT NULL,
  `id_student` int NOT NULL,
  `state` varchar(20) DEFAULT 'not entered',
  `failed` tinyint(1) DEFAULT NULL,
  `recalled` tinyint(1) DEFAULT NULL,
  `absent` tinyint(1) DEFAULT NULL,
  `grade` int DEFAULT NULL,
  `merit` tinyint(1) DEFAULT NULL,
  `id_report` int DEFAULT NULL,
  PRIMARY KEY (`id_appeal`,`id_student`),
  KEY `student_idx` (`id_student`),
  KEY `report_idx` (`id_report`),
  CONSTRAINT `appeal` FOREIGN KEY (`id_appeal`) REFERENCES `appeal` (`id_appeal`) ON UPDATE
      CASCADE,
  CONSTRAINT `student` FOREIGN KEY (`id_student`) REFERENCES `student` (`id_student`) ON UPDATE
      CASCADE,
  CONSTRAINT `enteredChecked` CHECK (((`state` <> 'entered') or ((`state` = 'entered') and (`failed` is not
      null) and (`recalled` is not null) and (`absent` is not null)))),
  CONSTRAINT `exam_chk_1` CHECK (((`merit` <> 1) or ((`grade` >= 30) and (`grade` <> NULL)))),
  CONSTRAINT `FailBooleans` CHECK ((((`failed` <> 1) or (`grade` < 18)) and ((`failed` <> 1) or (`absent` <> 1))
      and ((`failed` <> 1) or (`recalled` <> 1)) and ((`absent` <> 1) or (`recalled` <> 1)) and ((`absent` <> 1) or
      (`grade` < 18)) and ((`recalled` <> 1) or (`grade` < 18)))),
  CONSTRAINT `GradeValues` CHECK (((`grade` >= 18) and (`grade` <= 30))),
  CONSTRAINT `MeritAvailability` CHECK ((((`merit` <> '1') or (`grade` is not null)) and ((`merit` <> '1')
      or (`grade` >= 30)))),
  CONSTRAINT `NullGrade` CHECK ((((`merit` is null) and (`grade` is null)) or ((`merit` is not null) and (`grade`
      is not null)))),
  CONSTRAINT `Reported` CHECK ((((`state` = 'recorded') and (`id_report` is not null)) or ((`id_report` is null)
      and (`state` <> 'recorded')))),
  CONSTRAINT `stateValues` CHECK ((`state` in ('not entered', 'entered', 'published', 'refused', 'recorded')))
)
```

Tabella “report”

```
CREATE TABLE `report` (
   `id_report` int NOT NULL AUTO_INCREMENT,
   `id_appeal` int NOT NULL,
   `date` date DEFAULT NULL,
   `hour` time DEFAULT NULL,
   PRIMARY KEY (`id_report`),
   KEY `appeal_idx` (`id_appeal`),
   KEY `appeal_rep_idx` (`id_appeal`),
   CONSTRAINT `appeal_rep` FOREIGN KEY (`id_appeal`) REFERENCES `appeal` (`id_appeal`) ON UPDATE
        CASCADE
)
```

Tabella “security”

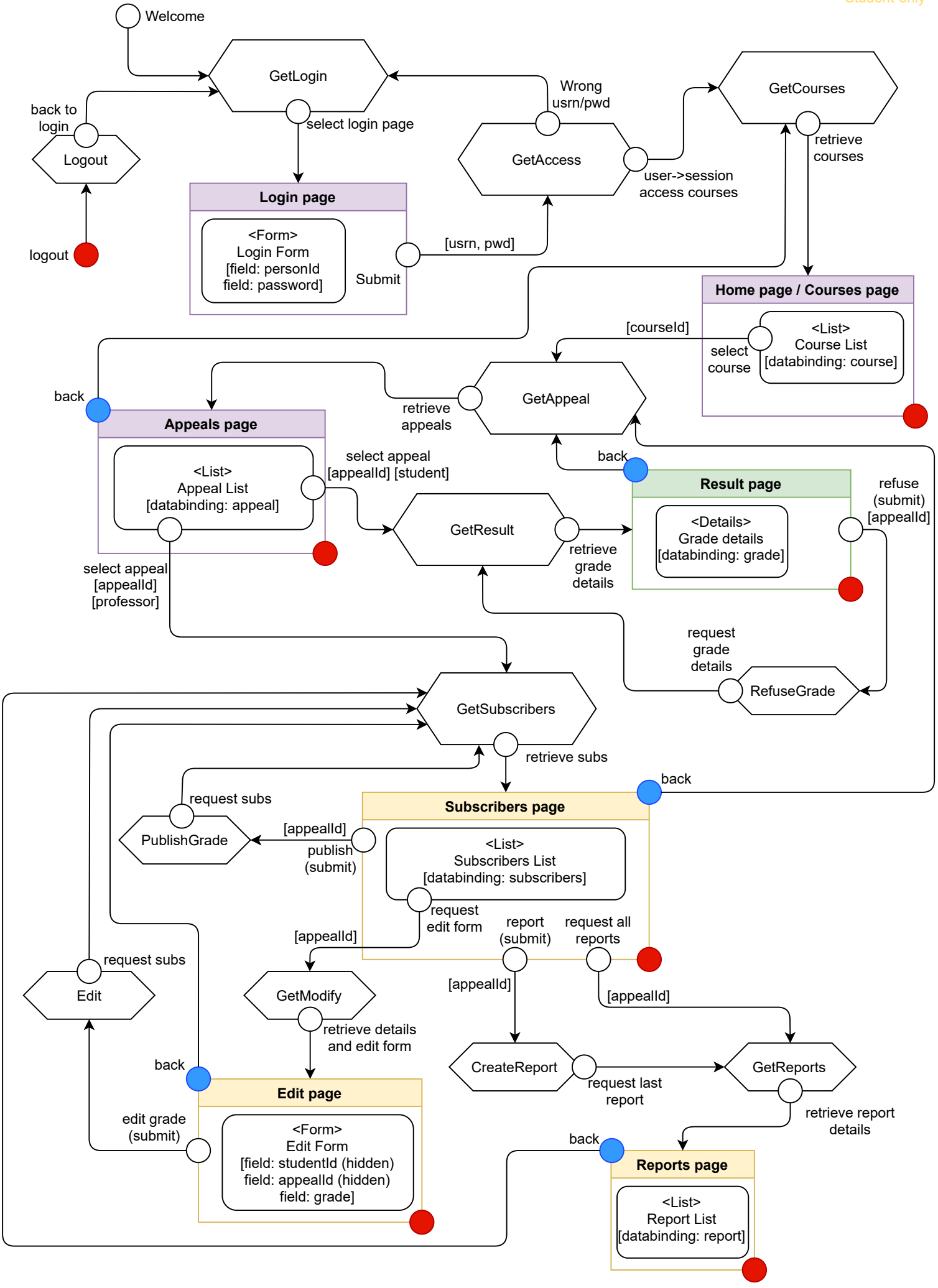
```
CREATE TABLE `security` (
   `id_person` int NOT NULL,
   `last_course` int DEFAULT NULL,
   `last_appeal` int DEFAULT NULL,
   `last_student` int DEFAULT NULL,
   PRIMARY KEY (`id_person`),
   KEY `last_course_idx` (`last_course`),
   KEY `last_appeal_idx` (`last_appeal`),
   KEY `last_student_idx` (`last_student`),
   CONSTRAINT `last_appeal` FOREIGN KEY (`last_appeal`) REFERENCES `appeal` (`id_appeal`) ON UPDATE
        CASCADE,
   CONSTRAINT `last_course` FOREIGN KEY (`last_course`) REFERENCES `course` (`id_course`) ON UPDATE
        CASCADE,
   CONSTRAINT `last_student` FOREIGN KEY (`last_student`) REFERENCES `student` (`id_student`) ON ù
        UPDATE CASCADE
)
```

Analisi dei requisiti
Nella presente sezione sono evidenziati nel testo della specifica per la versione pure HTML i contenuti web per la realizzazione dell'applicazione; sono indicati in rosso le pagine, in verde i componenti, in blu gli eventi, in fucsia le 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 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. Il voto può non essere ancora definito. Lo stato 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. 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 selezionare il voto. L'invio della form provoca la modifica o l'inserimento del voto. Inizialmente le righe sono nello stato "non inserito". L'inserimento e le successive eventuali modifiche portano la riga nello stato "inserito". Alla tabella è associato un bottone PUBBLICA che comporta la pubblicazione delle righe con lo stato INSERITO. La pubblicazione rende il voto non più modificabile dal docente e visibile allo studente e cambia lo stato 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. 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 della riga dello studente per quell'appello anche 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.

Flow diagram for Pure HTML version

Common pages
Professor-only
Student-only



Componenti

Model objects (beans):

- User
- Course
- Appeal
- Grade
- Report
- ErrorMsg

Data Access Objects (DAO):

- UserDAO
  - getUser (id, password)
- CourseDAO
  - getCoursebyId (courseId)
  - getCoursesByProfessorId (professorId)
  - getCoursesByStudentId (studentId)
  - hasCourse (courseId, personId, accessRights)
- AppealDAO
  - getAppealsByCourse (courseId, persondId, accessRights)
  - getAppealById (appealId)
  - hasAppeal (appealId, persondId, accessRights)
- GradeDAO
  - getGradesByAppealId (appealId)
  - getGradesByFieldASC (appealId, field)
  - getGradesByFieldDESC (appealId, field)
  - enterGrade (appealId, studentId, gradeValue)
  - editGrade (grade)
  - refuseGrade (appealId, studentId)
  - failRefused (appealId)
  - reportGrade (appealId, reportId)
  - countReportableGrades (appealId)
  - publishGrade (appealId)
  - getResultByAppealAndStudent (appealId, studentId)
  - getRecordedGrades (report)
- ReportDAO
  - getReportById (reportId)
  - getReportByAppeal (appeal)
  - createReport (appealId)
  - gelLastReport (appealId)
  - getAllReports (appealId)
- SecurityDAO
  - getLastCourse (personId)
  - getLastAppeal (personId)
  - getLastStudent (personId)
  - setLastCourse (personId, courseId)
  - setLastAppeal (personId, appealId)
  - setLastStudent (personId, studentId)
  - clearRow (personId)
  - insertRow (personId)
  - removeRow (personId)

Controllers (servlets):

- GetLogin
- GetAccess
- GetCourses
- GetAppeal
- GetSubscribers
- GetResult
- GetModify
- Edit
- PublishGrade
- CreateReport
- GetReports
- RefuseGrade
- Logout

Views (templates):

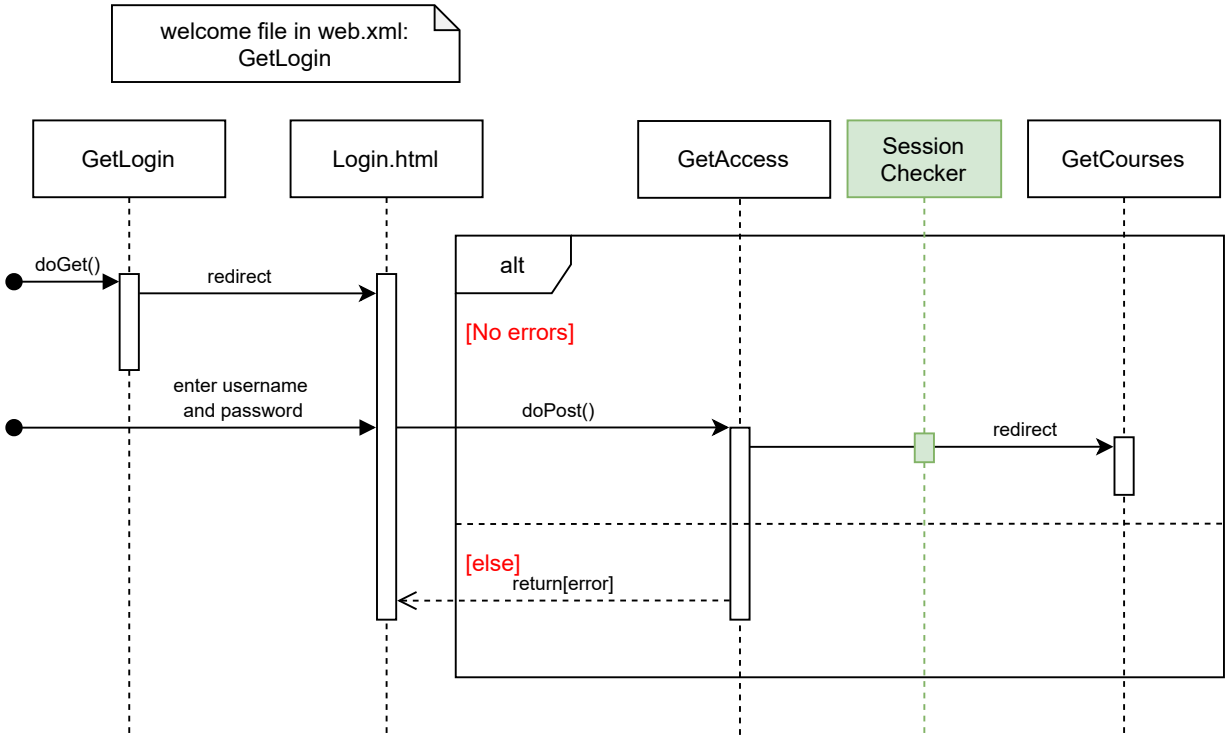
- Login
  - Login form
- Home (course)
  - List of courses
- Appeal
  - List of appeals
- Subscribers
  - List of Subscribers
  - Edit buttons
  - Publish button
  - Report button
- Modify
  - Edit form
- Reports
  - Report details
- Result
  - Grade details
  - Refuse button

Filters:

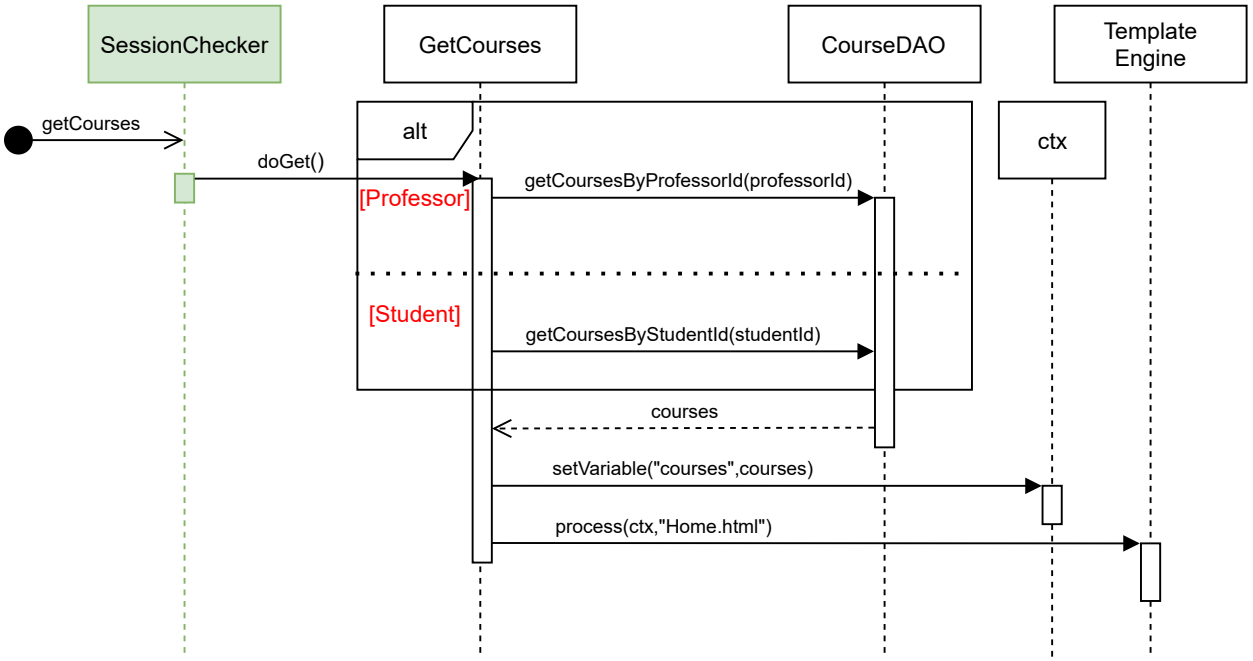
- SessionChecker
- ProfessorFilter
- StudentFilter
- GetMethodFilter
- PostMethodFilter
- AppealChecker

SEQUENCE DIAGRAM PURE HTML

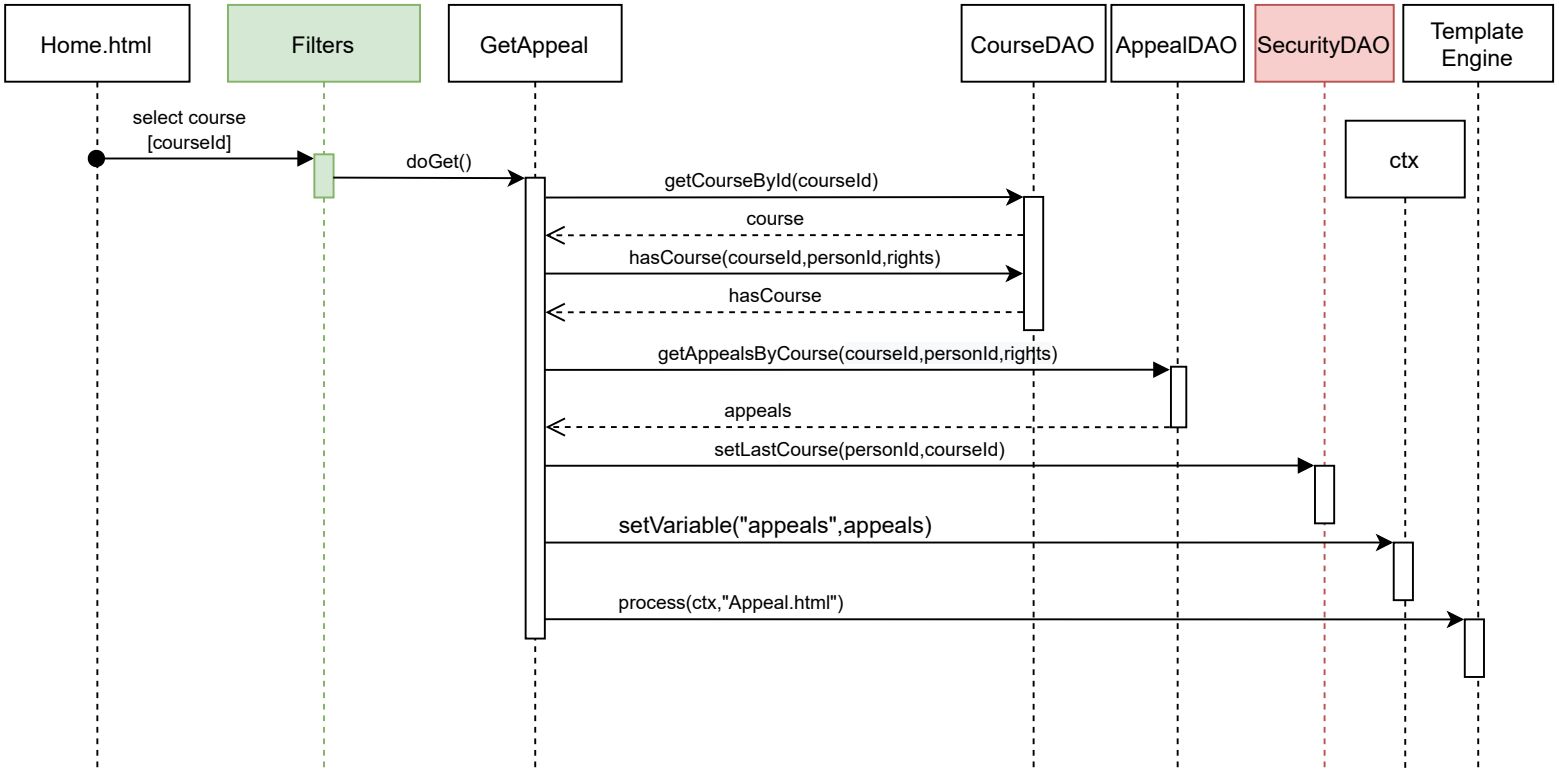
Login



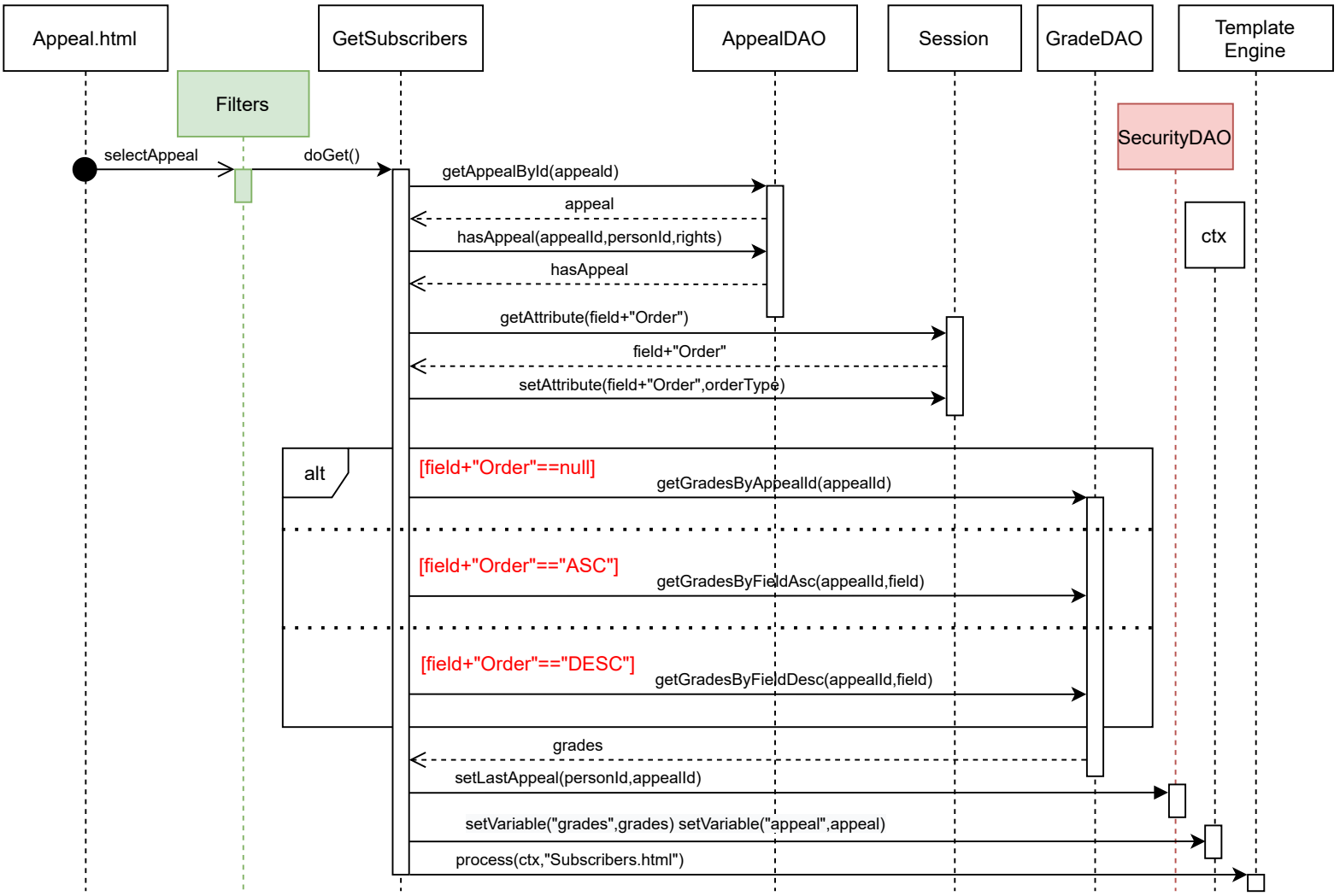
Go to home



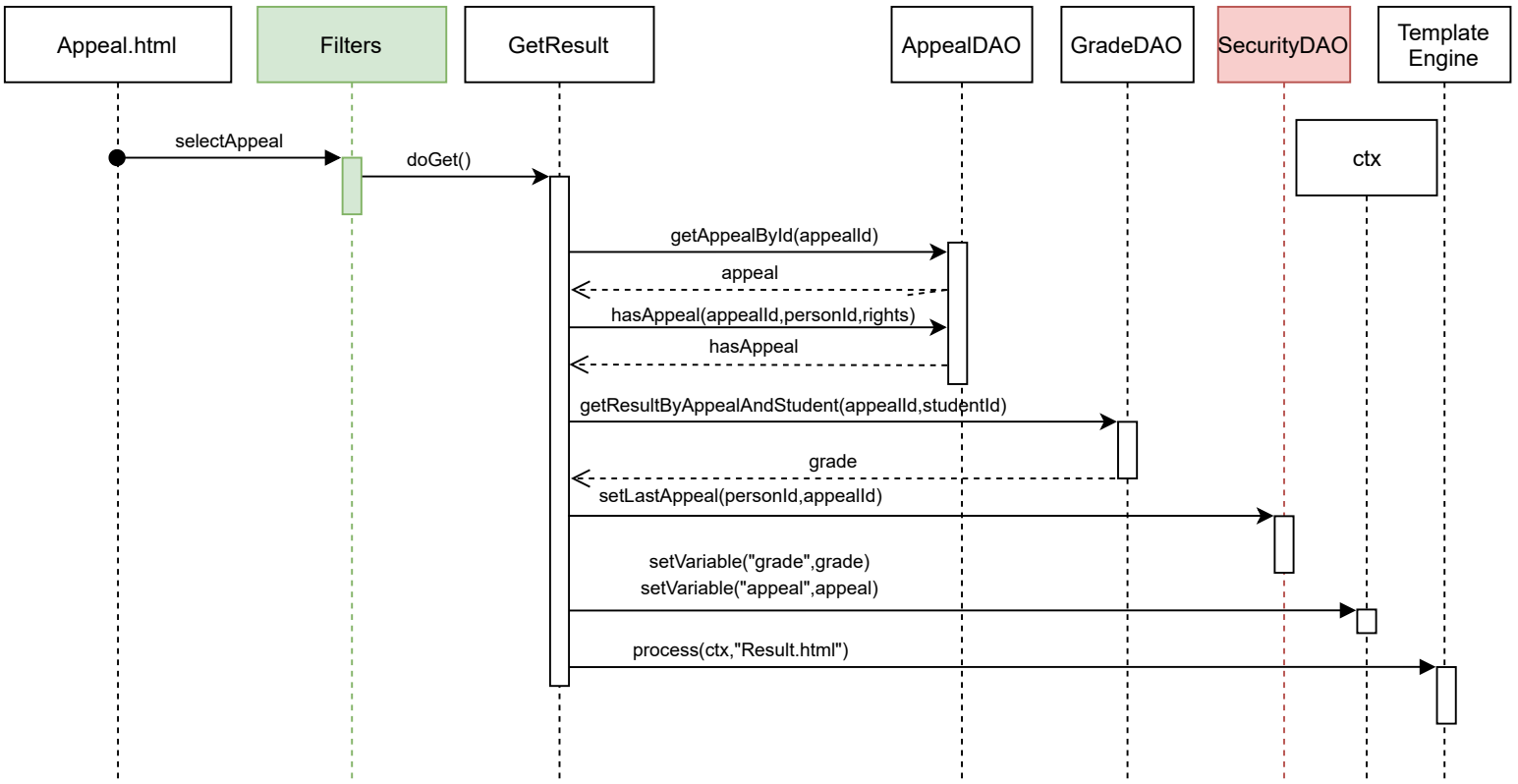
Go to appeals



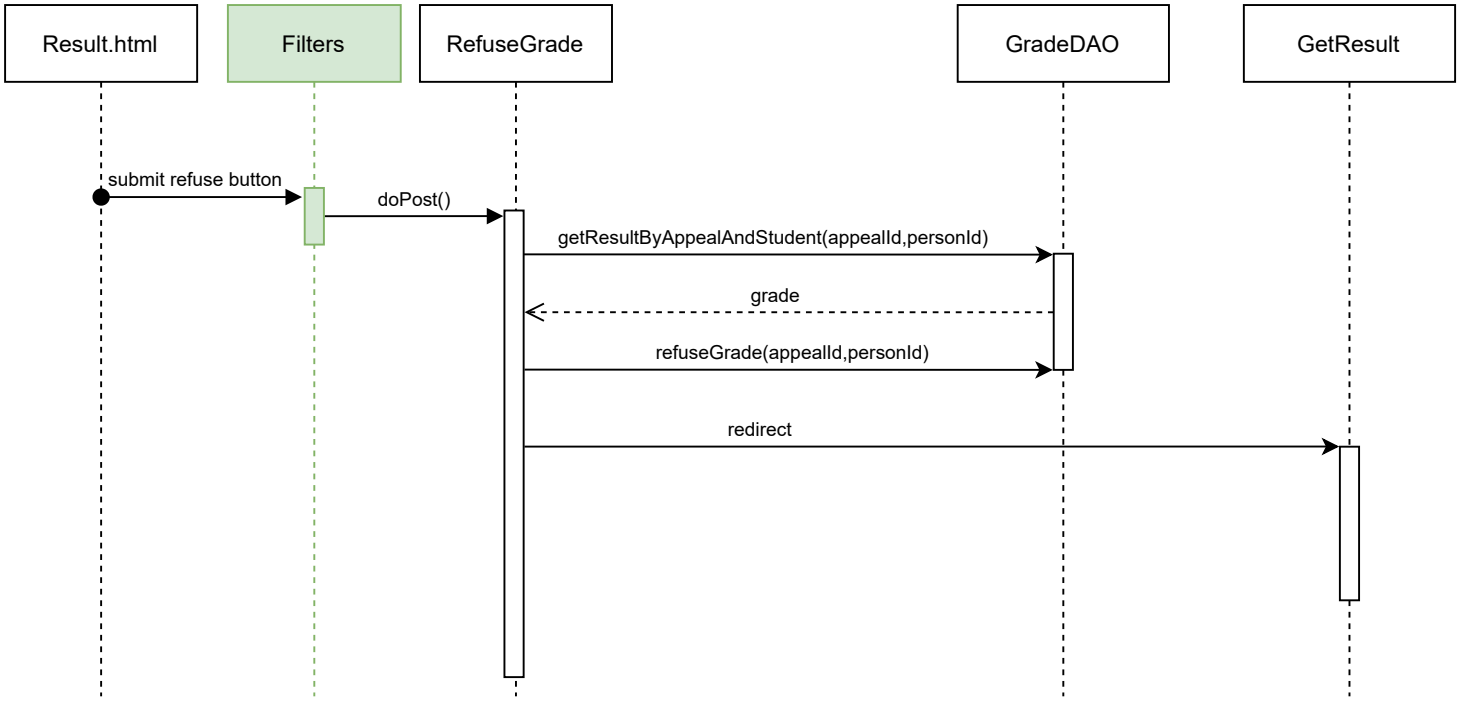
Go to subscribers



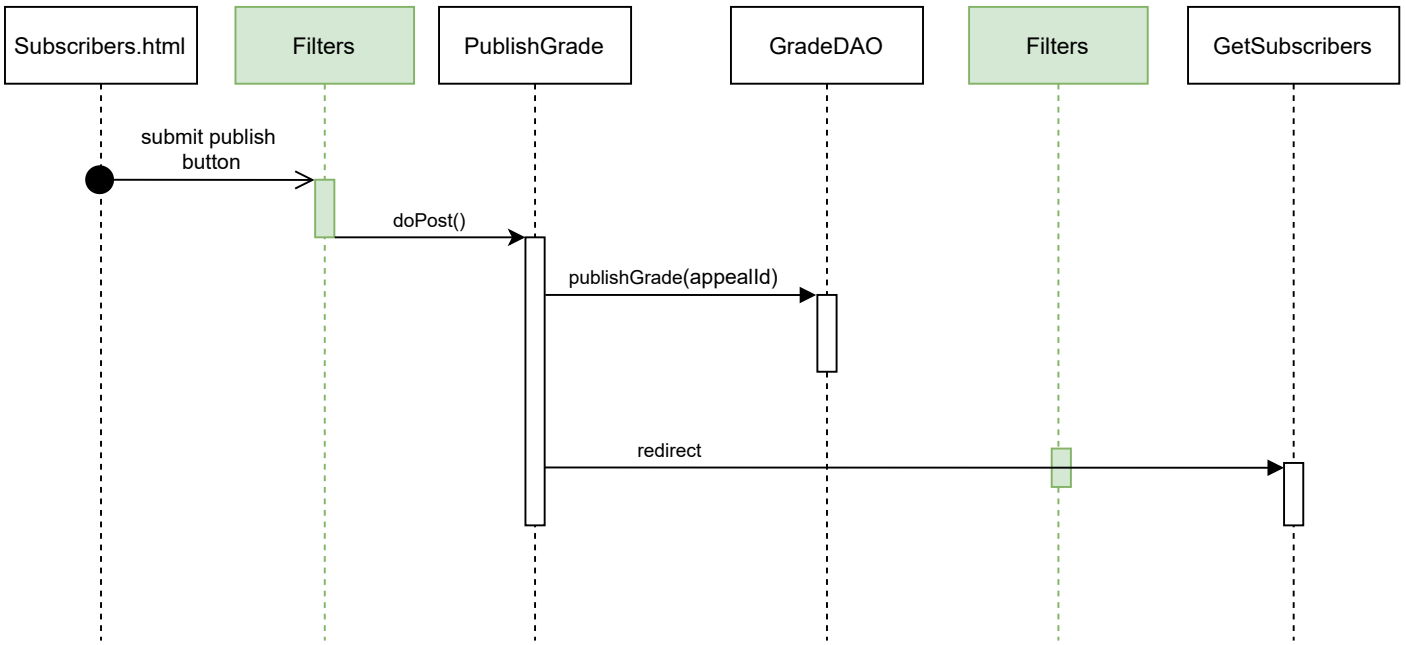
Go to result



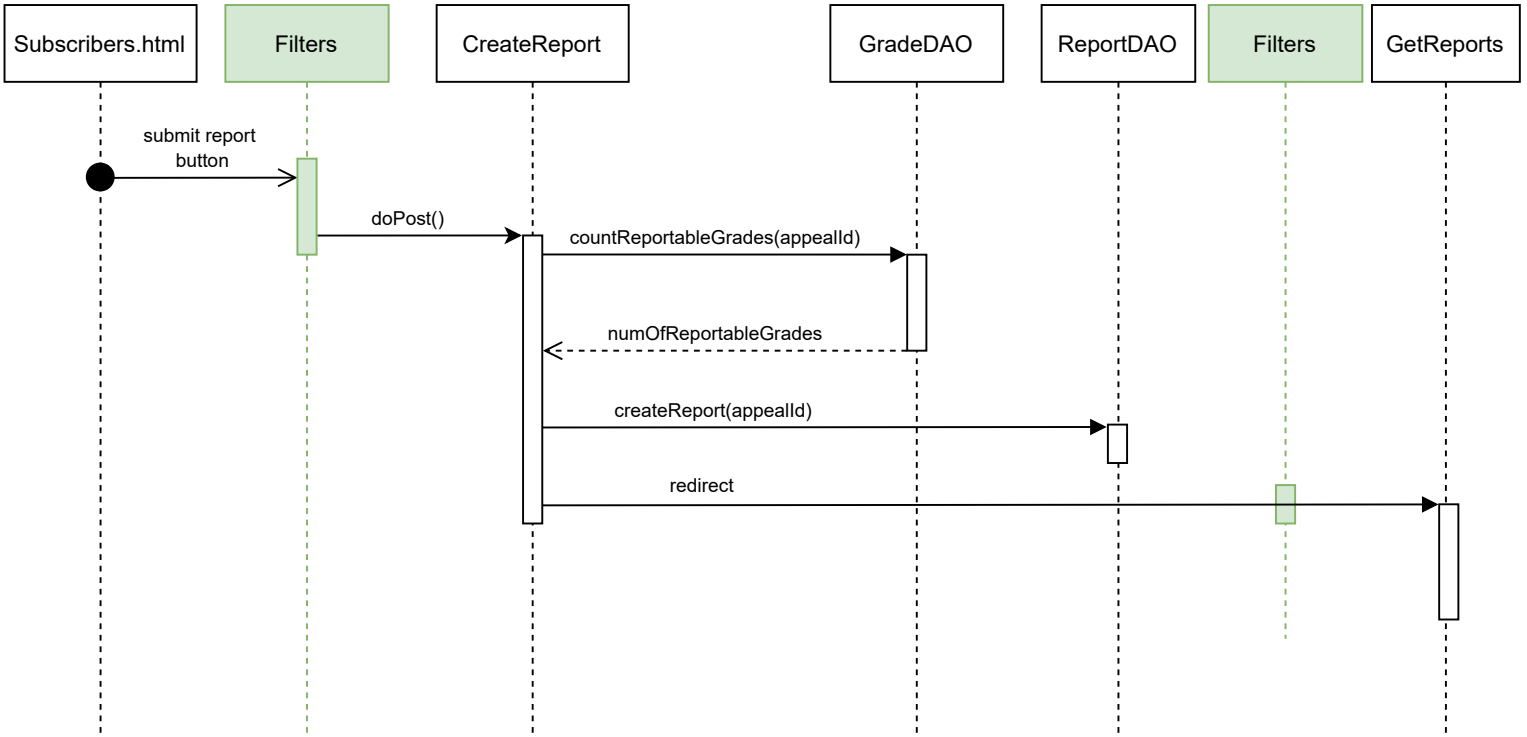
Refuse grade



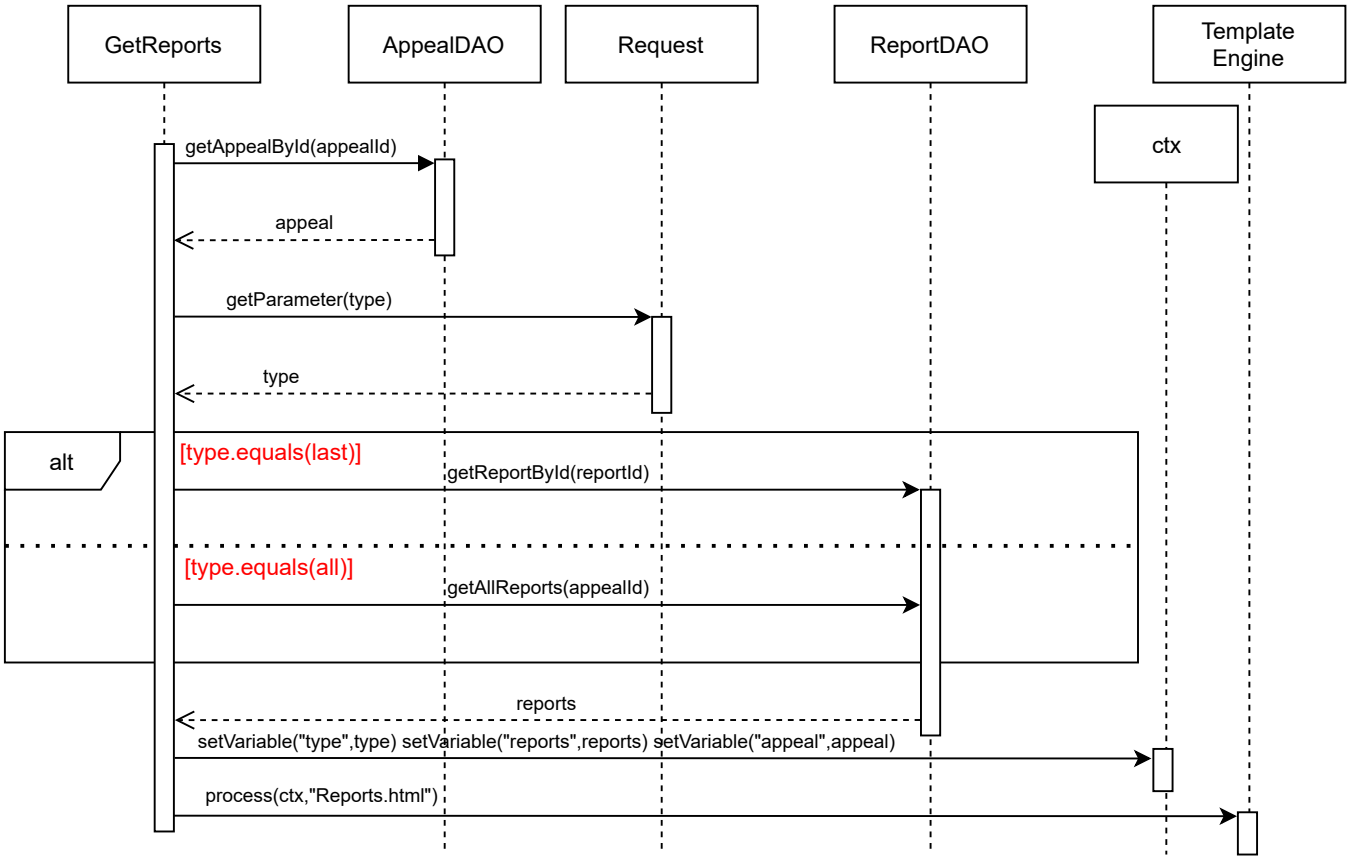
Publish grade



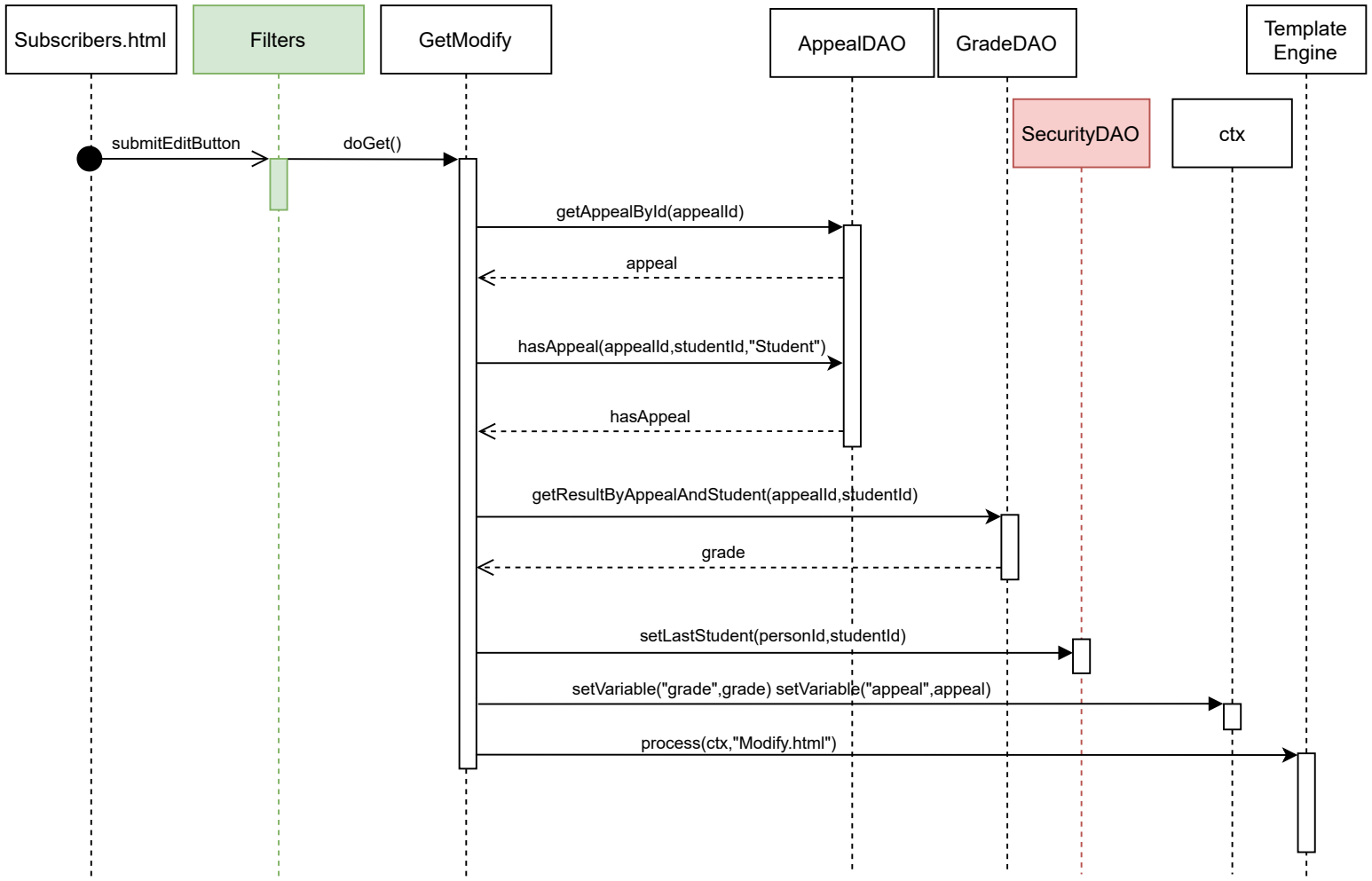
Create report



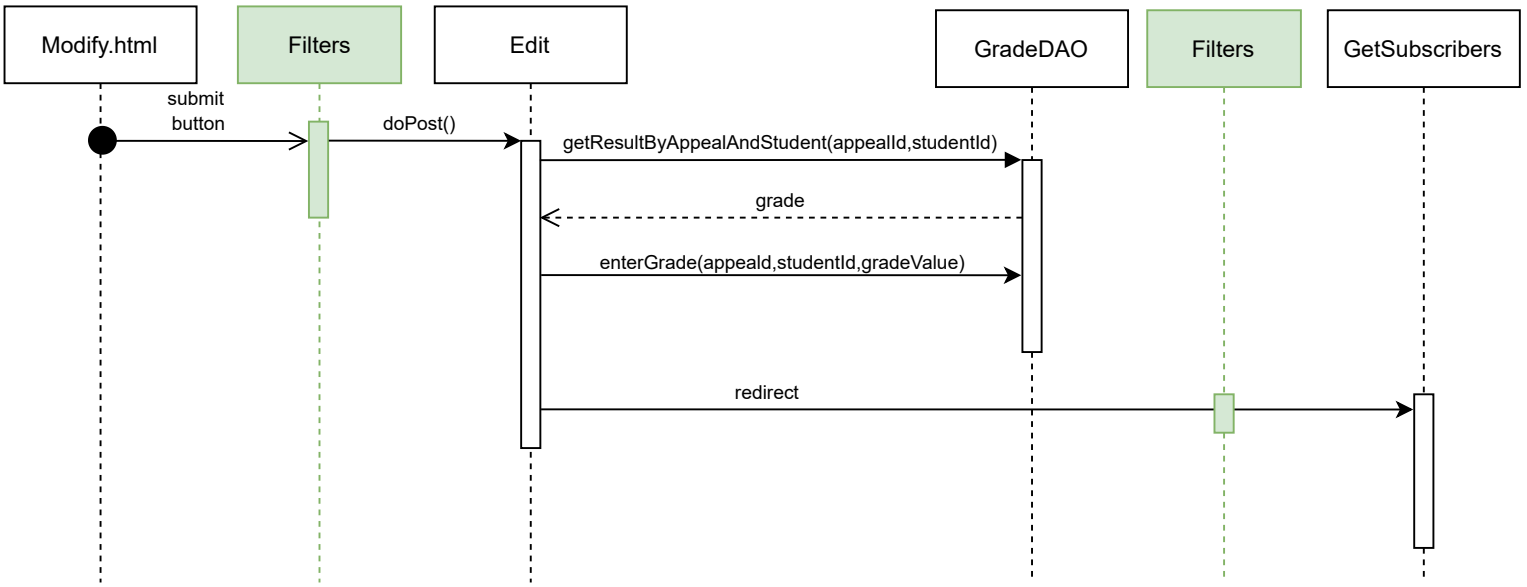
Go to reports



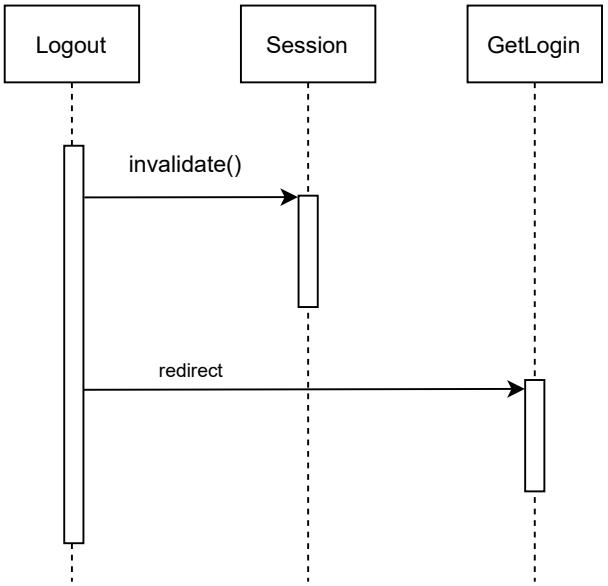
Go to modify



Edit grade



Logout



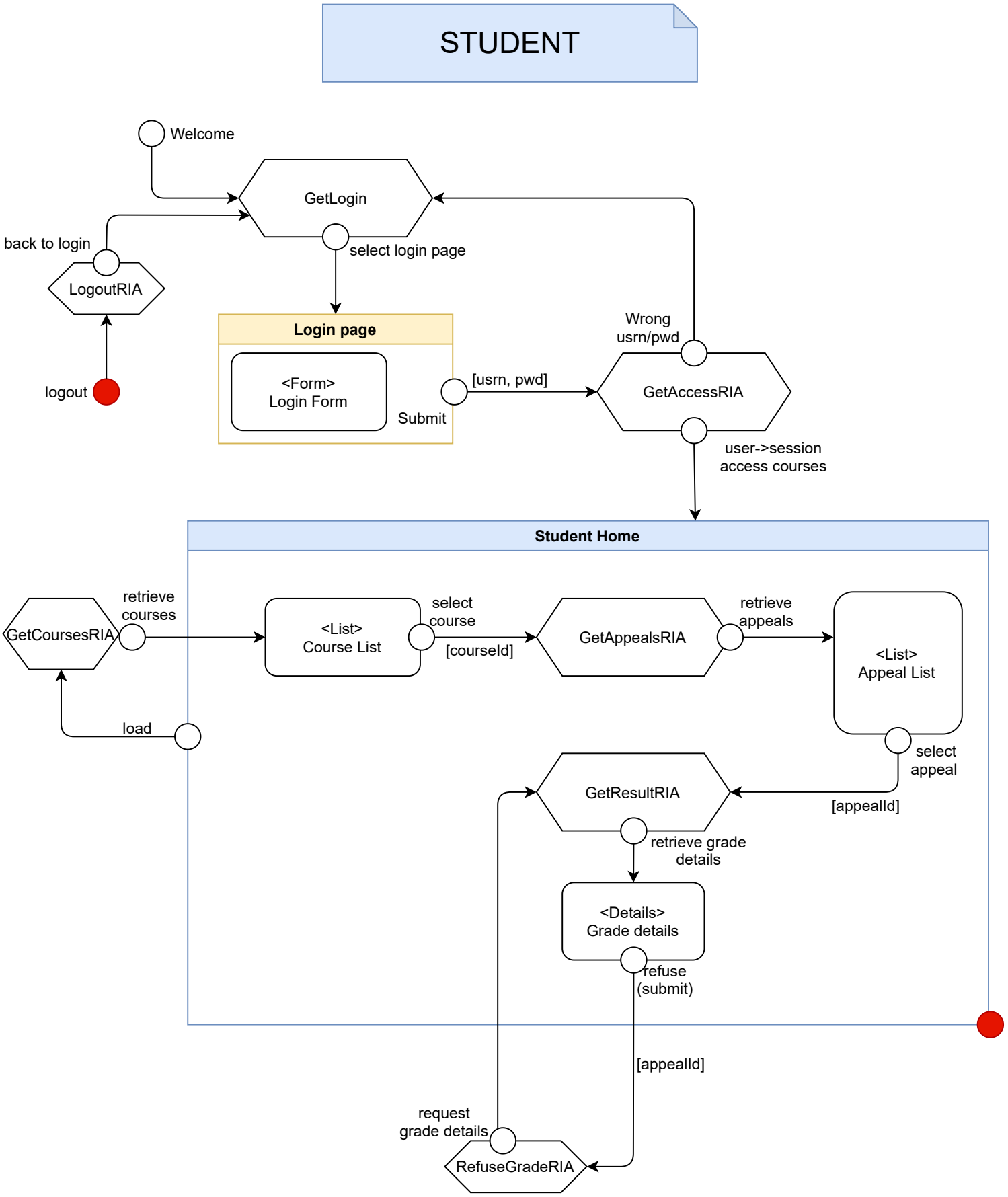
Analisi dei requisiti
Nella presente sezione sono evidenziati nel testo della specifica per la versione RIA i contenuti web per la realizzazione dell'applicazione; sono indicati in rosso le pagine, in verde i componenti, in blu gli eventi, in fucsia le azioni.

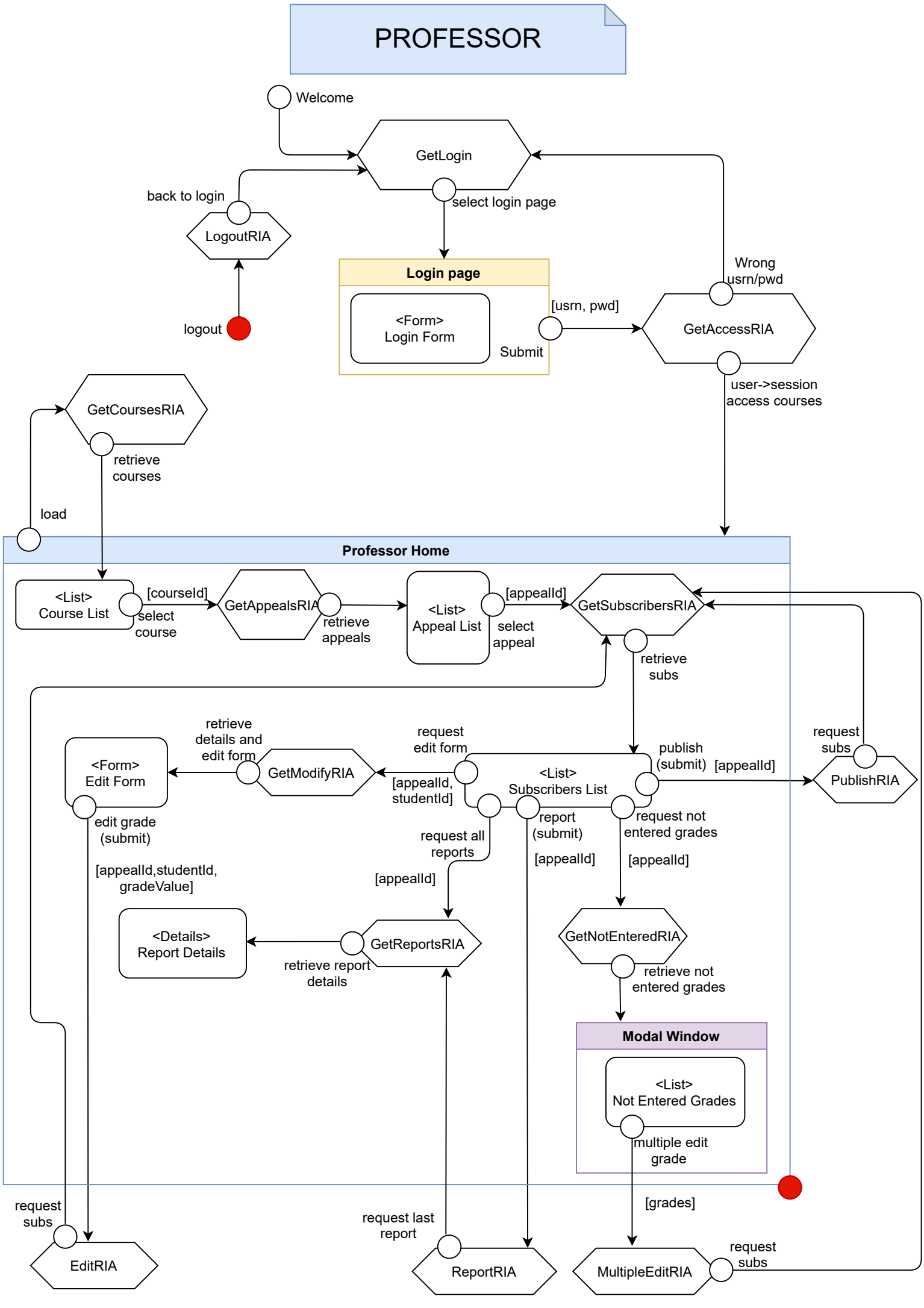
Si realizzi un'applicazione client server web che modifica le specifiche precedenti come segue:

- 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 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.

Rispetto alla versione precedente si considerino solo le pagine evidenziate in questa sezione mentre, in merito ai componenti, si considerino ancora presenti tutti quelli già introdotti; essi si troveranno, però, raccolti nelle sole due nuove pagine.

Flow diagram for RIA version




PROFESSOR
Welcome
GetLogin
back to login
LogoutRIA
select login page
Login page
<Form>
Login Form
Submit
[usrn, pwd]
GetAccessRIA
Wrong
usrn/pwd
logout
user->session
access courses
GetCoursesRIA
retrieve
courses
load
Professor Home
<List>
Course List
[courseId]
select
course
GetAppealsRIA
retrieve
appeals
<List>
Appeal List
[appealId]
select
appeal
GetSubscribersRIA
retrieve
subs
retrieve
details and
edit form
request
edit form
publish
(submit)
[appealId]
request
subs
<Form>
Edit Form
GetModifyRIA
[appealId,
studentId]
<List>
Subscribers List
PublishRIA
edit grade
(submit)
request all
reports
report
(submit)
request not
entered grades
[appealId,studentId,
gradeValue]
[appealId]
[appealId]
[appealId]
<Details>
Report Details
GetReportsRIA
retrieve report
details
GetNotEnteredRIA
retrieve not
entered grades
Modal Window
<List>
Not Entered Grades
multiple edit
grade
request
subs
EditRIA
request last
report
ReportRIA
[grades]
MultipleEditRIA
request
subs

Componenti
Nota: per i DAO non sono stati riportati i metodi già introdotti nella versione pure HTML; i metodi indicati di seguito sono, quindi, quelli specifici per la versione RIA

Model objects (beans):
- User
- Course
- Appeal
- Grade
- Report

Data Access Objects (DAO):
- UserDAO
- CourseDAO
- AppealDAO
- GradeDAO
    - getNotEnteredGradesByAppealID (appealId)
- ReportDAO
- SecurityDAO

Controllers (servlets):
- GetLogin
- GetAccessRIA
- GetCoursesRIA
- GetAppealsRIA
- GetSubscribersRIA
- GetResultRIA
- GetModifyRIA
- EditRIA
- PublishRIA
- ReportRIA
- GetNotEnteredRIA
- MultipleEditRIA
- GetReportsRIA
- RefuseGradeRIA
- LogoutRIA

Views (templates):
- LoginRIA
    - Login form
- ProfessorHome
    - List of courses
    - List of appeals
    - List of subscribers
    - Edit buttons
    - Publish button
    - Report button
    - Multiple edit button
    - Edit form
    - Report details
    - Modal window (multiple edit form)

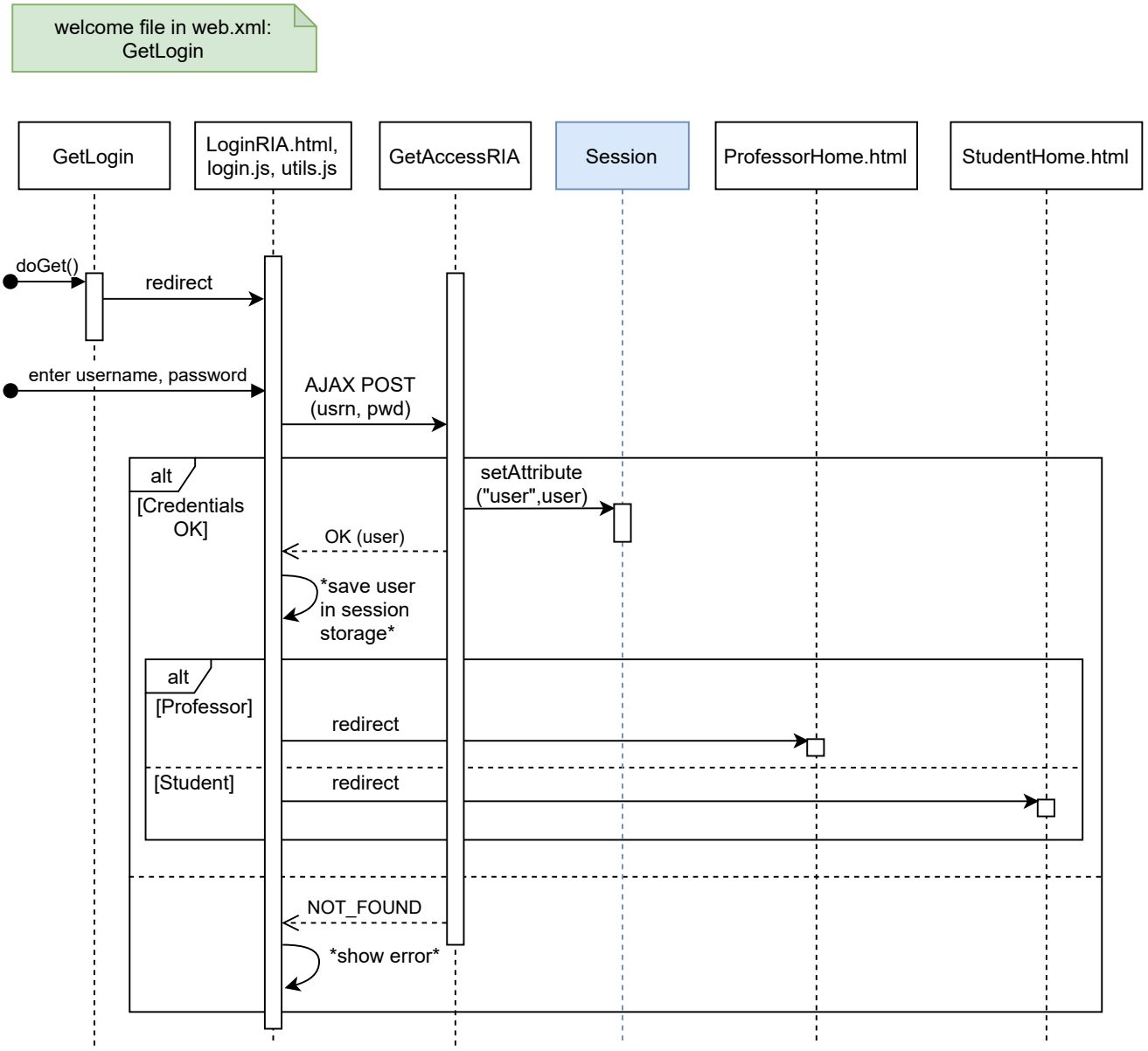
- StudentHome
    - List of courses
    - List of appeals
    - Grade details
    - Refuse button

Filters:
- SessionChecker
- ProfessorFilter
- StudentFilter
- GetMethodFilterRIA
- PostMethodFilterRIA
- AppealCheckerRIA

Scripts:
- login.js
- utils.js
- professor.js
- student.js

SEQUENCE DIAGRAM RIA

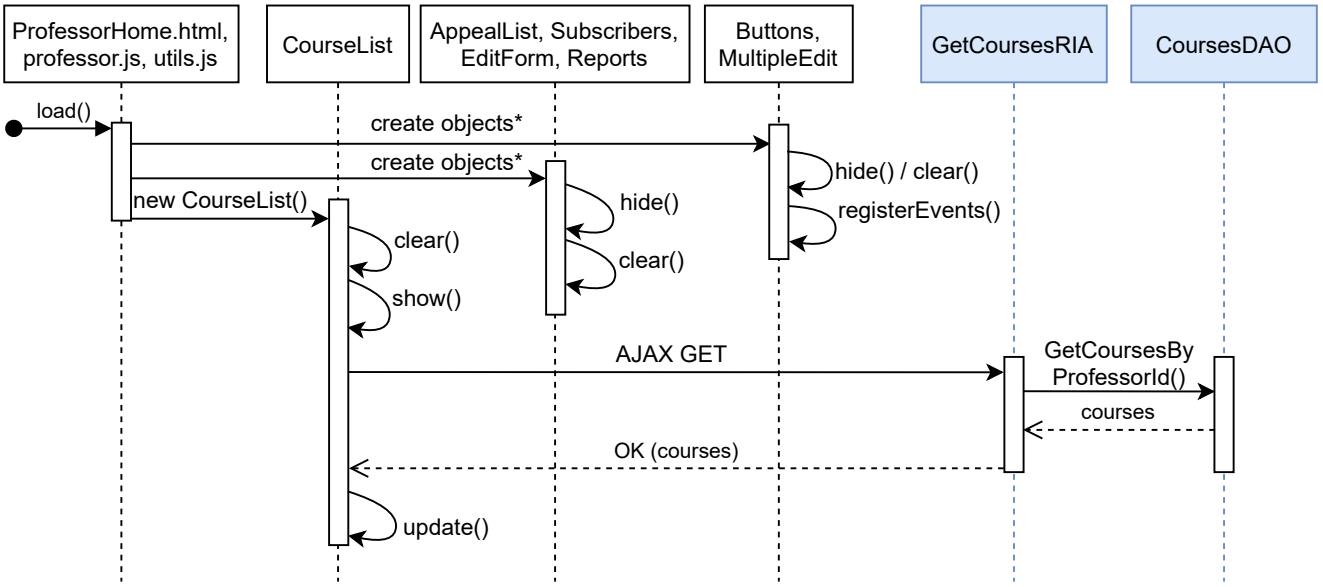
Login



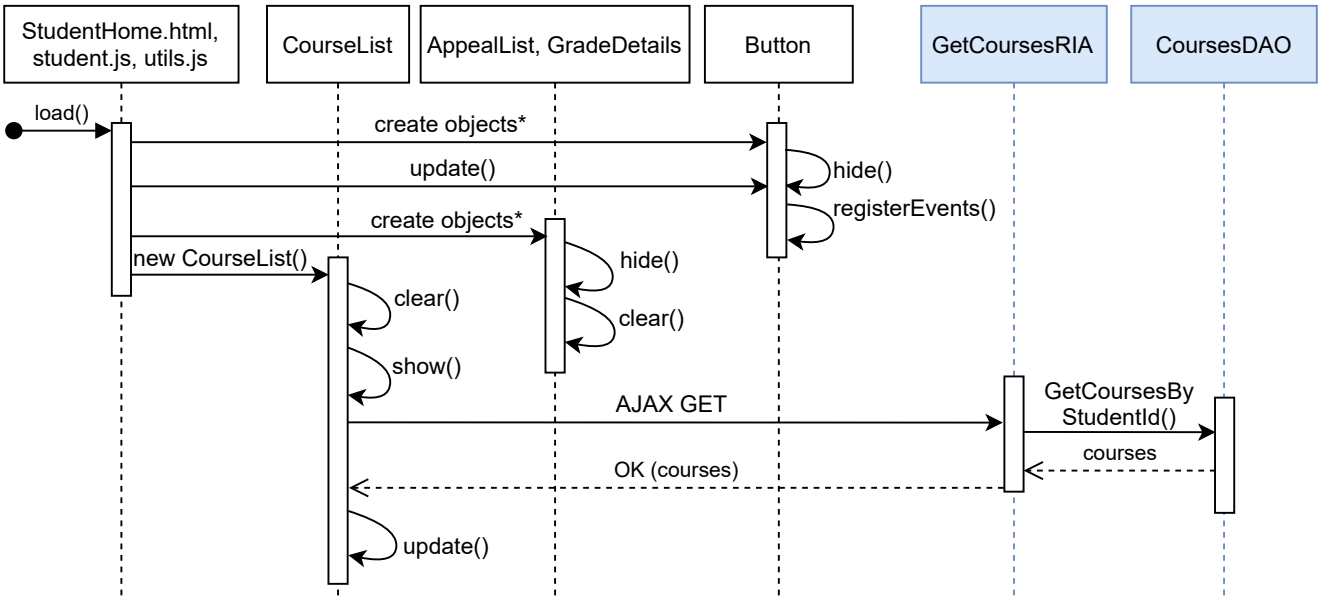
*During object initialization DOM elements are fetched

Person Id field is always inferred from session object

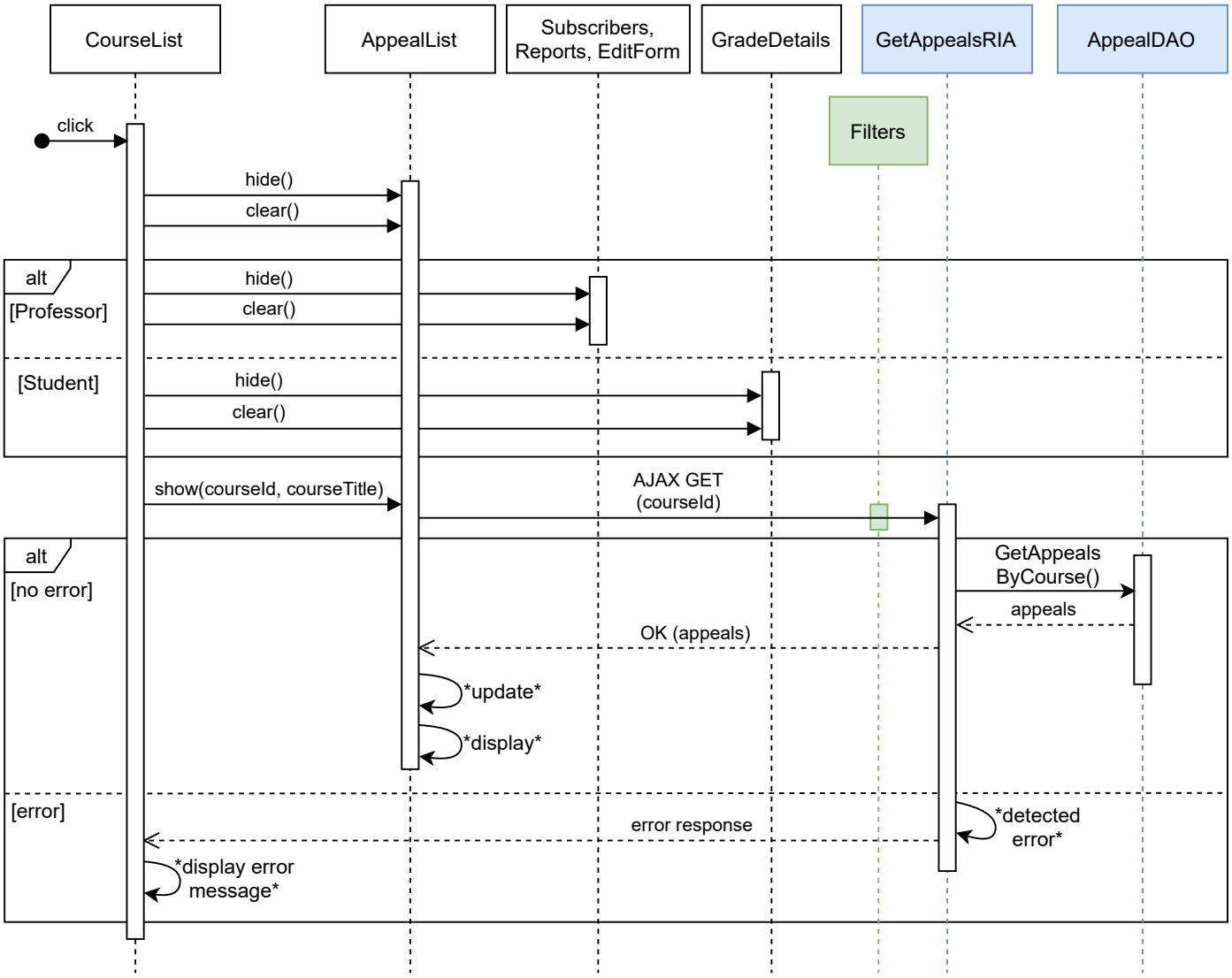
Professor's page load



Student's page load

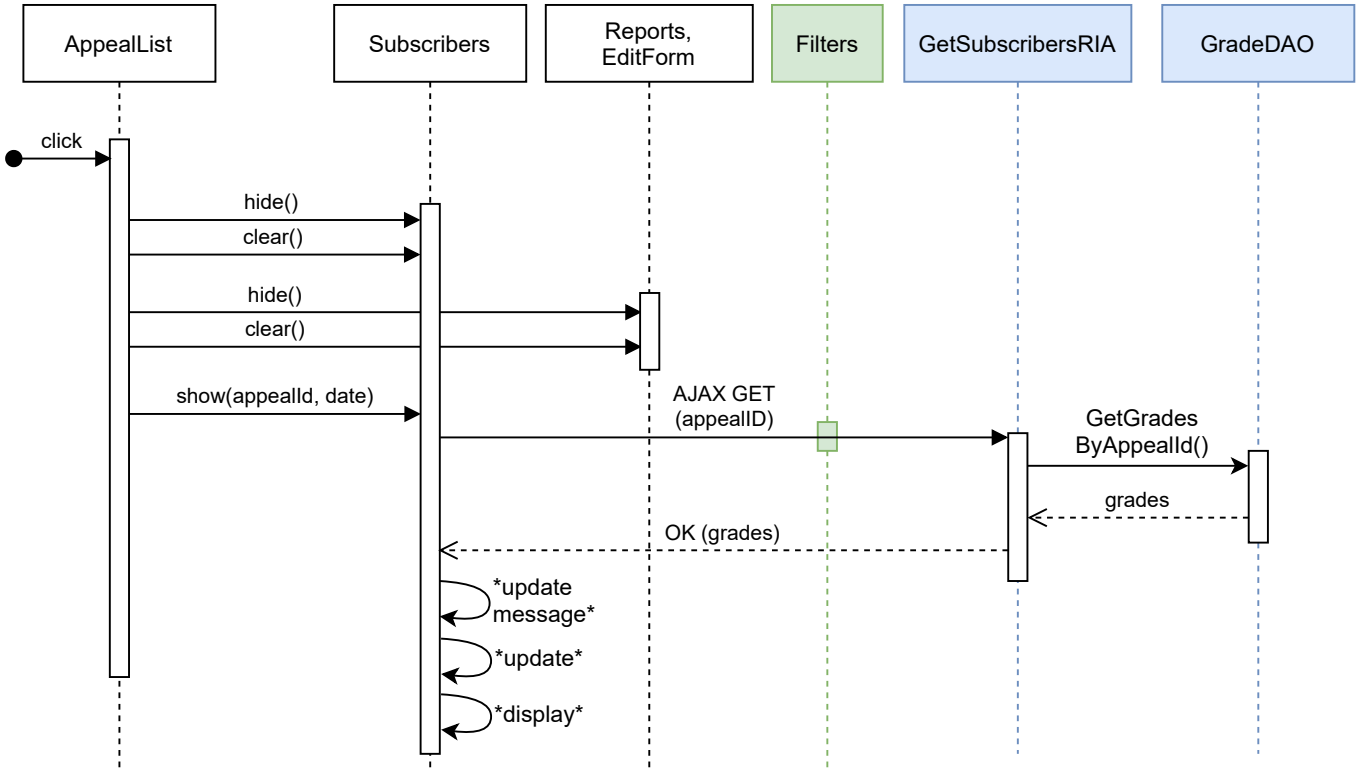


Course selection

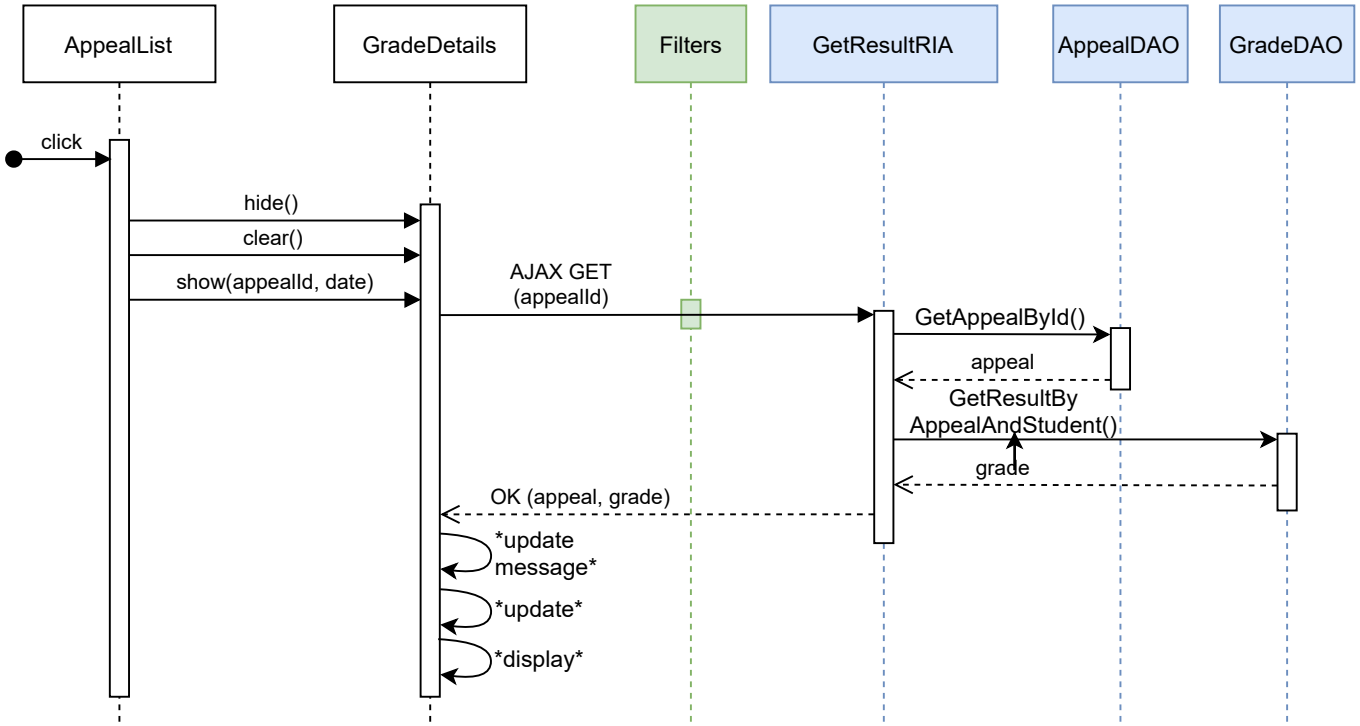


Error handling process is the same for all update() methods; thence it won't be repeated in next diagrams

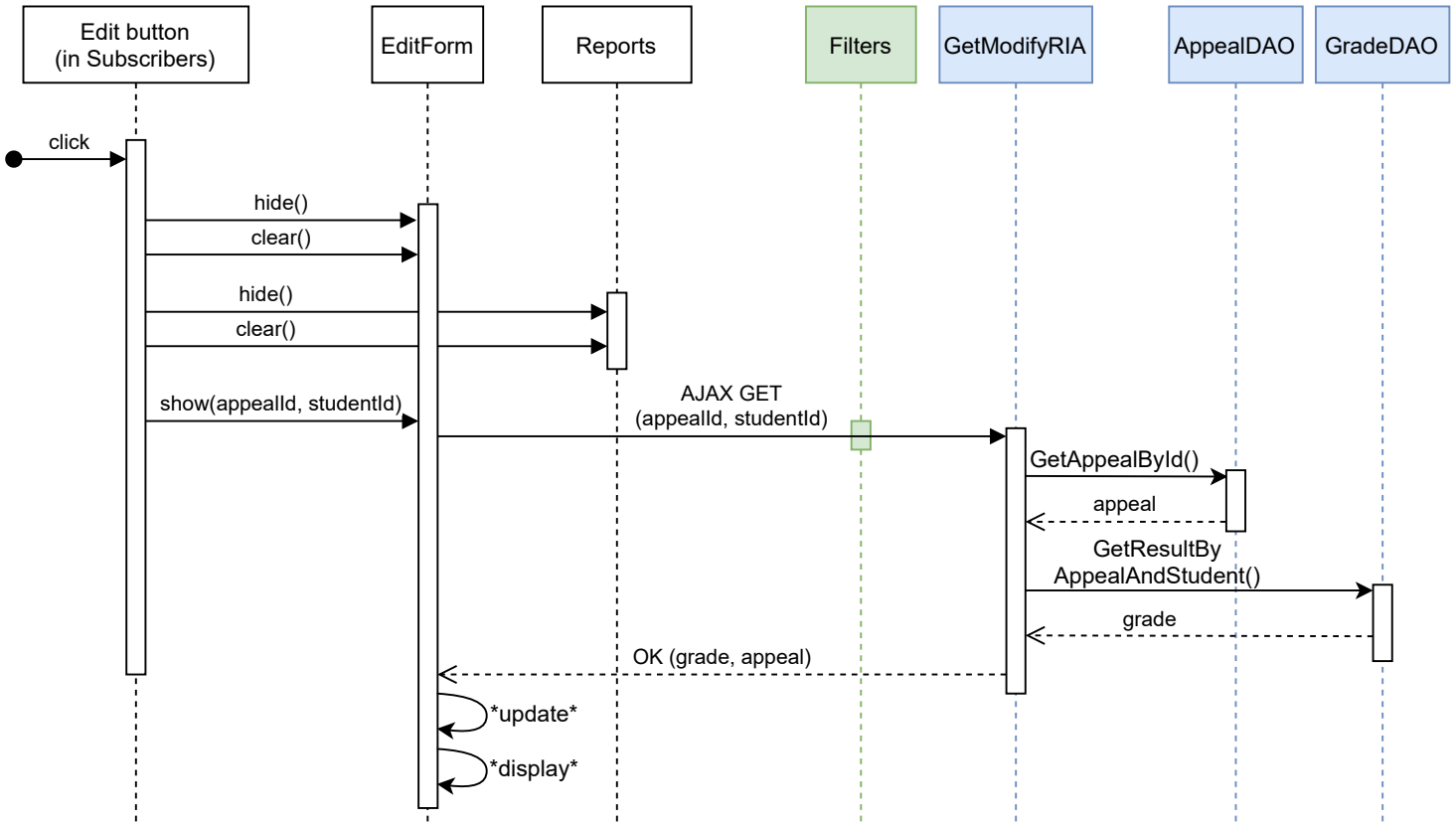
Professor's appeal selection



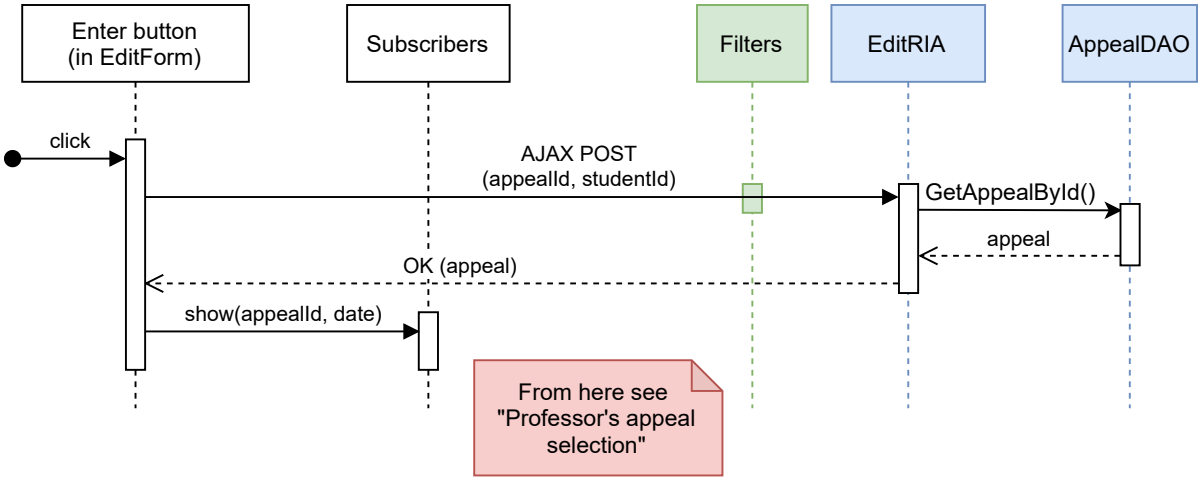
Student's appeal selection



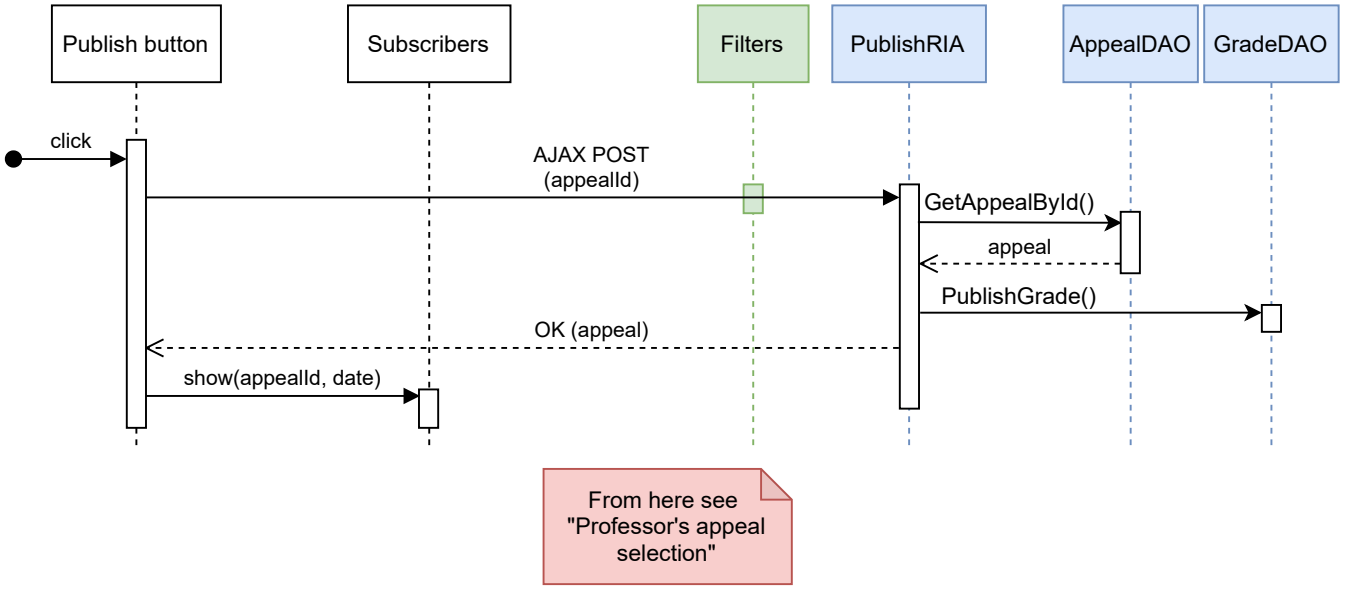
Editing grades (form request)



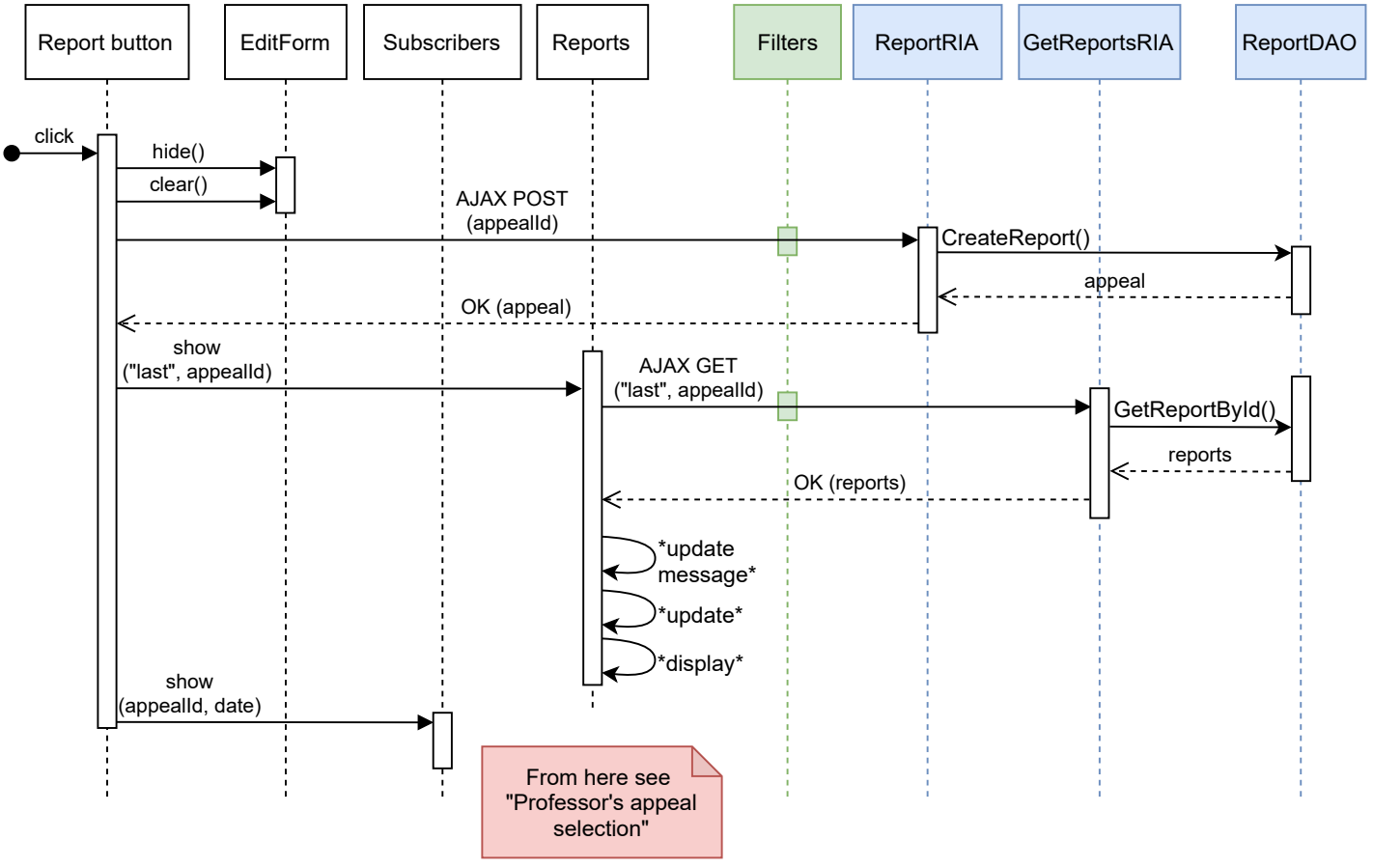
Editing grades (submit grade)



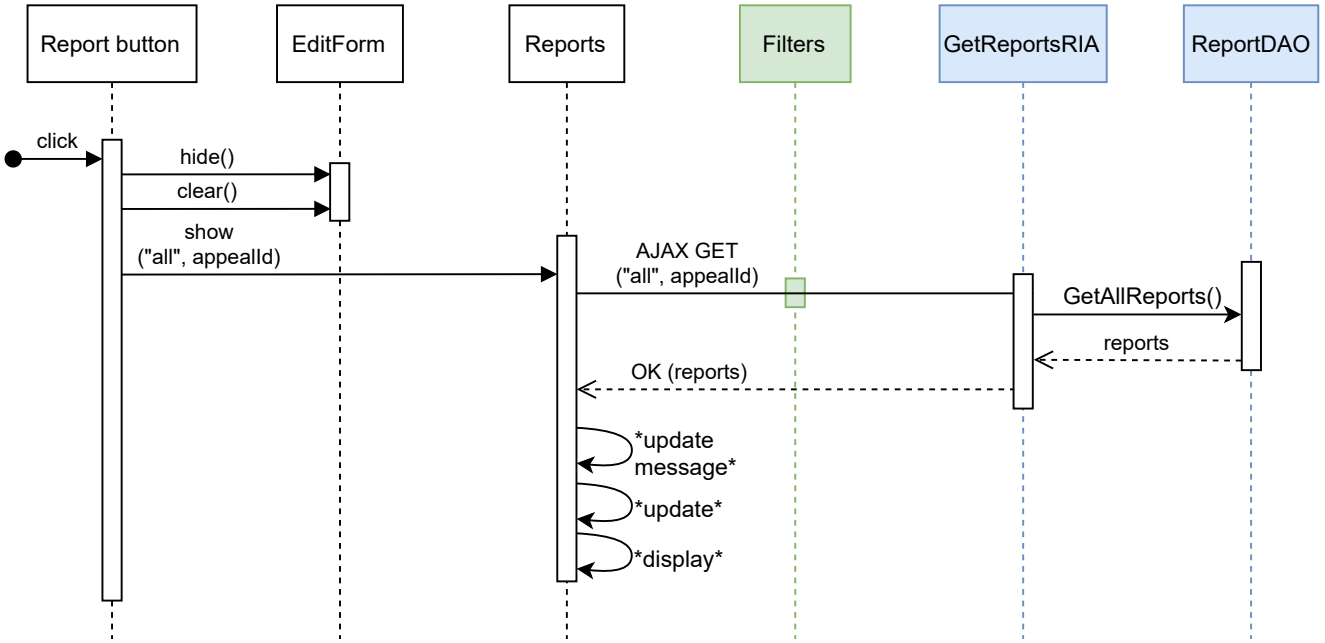
Publishing grades



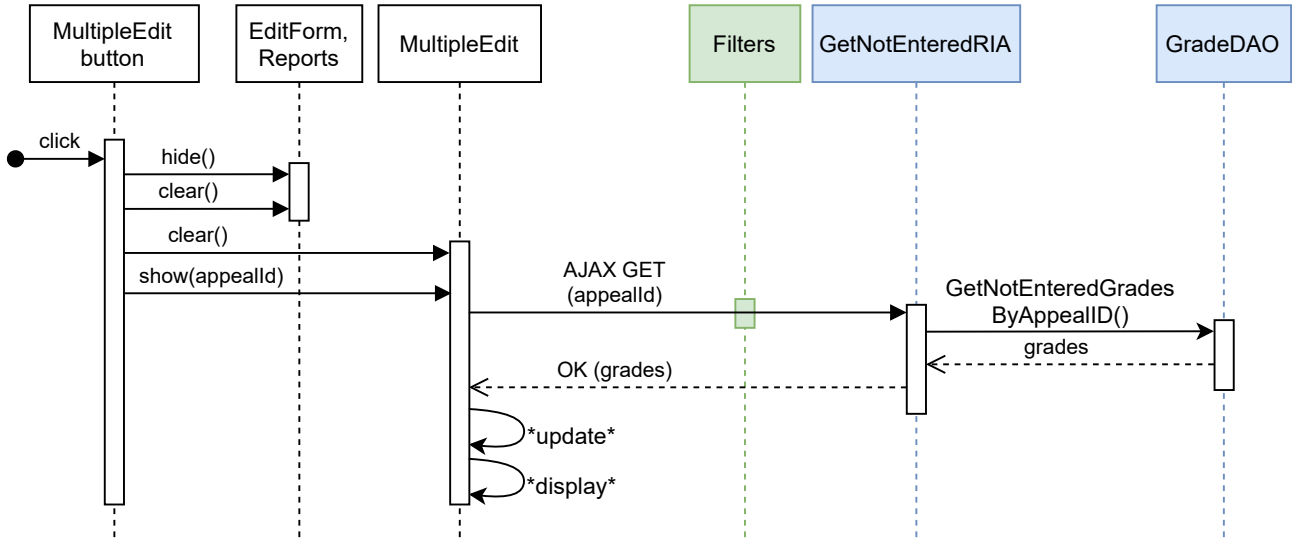
Reporting grades



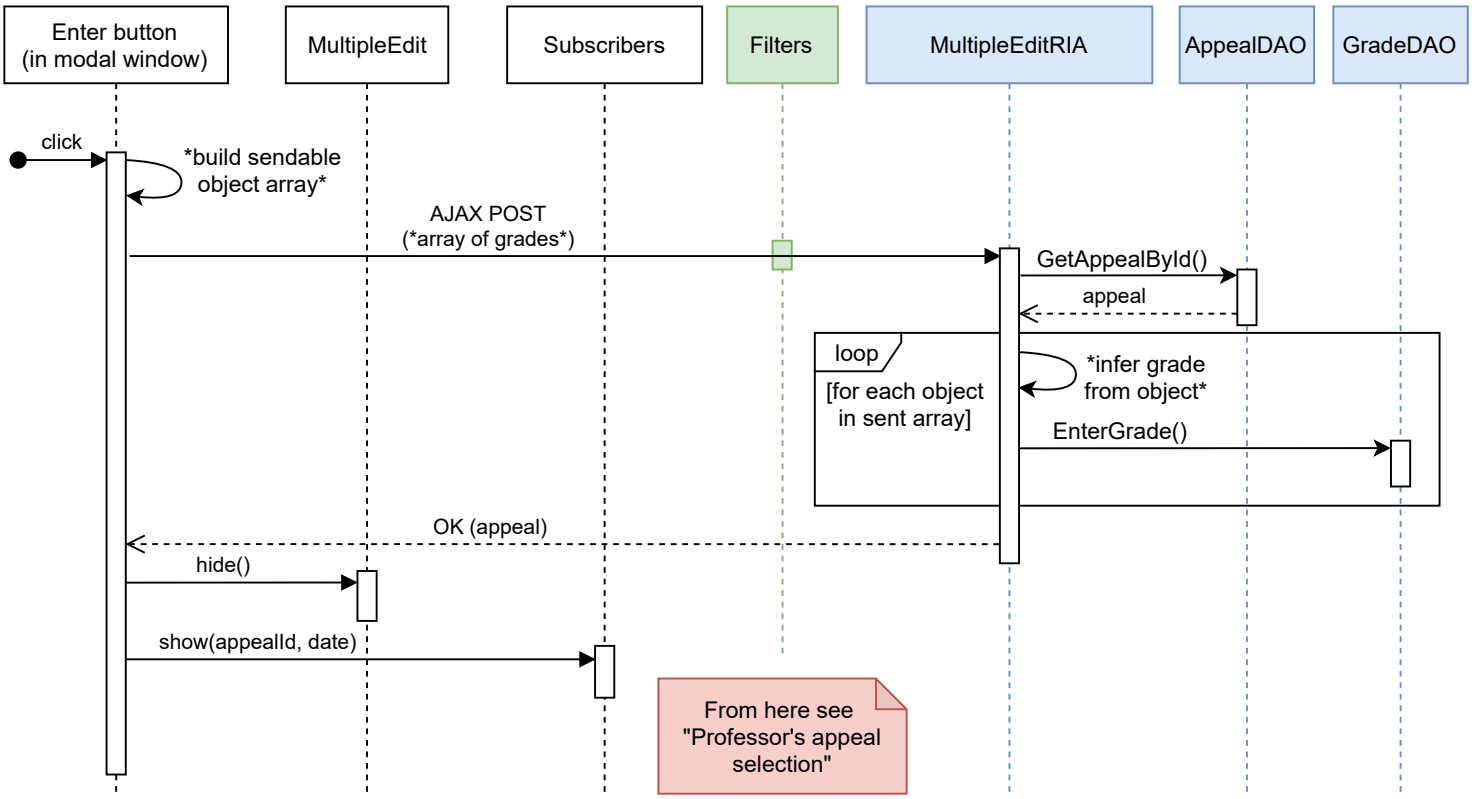
Showing all reports



Edit multiple grades
(fetch 'not entered' grades)



Edit multiple grades (enter grades)



Refusing grades