Immagine che contiene oggetto

Descrizione generata automaticamente

Università

degli Studi

di Catania

**UNIBOOK**

Progetto di Ingegneria del software.

Flavio Vincenzo Condemi

Francesco Maria Coco

INDICE

[1. IDEAZIONE E ANALISI DEI REQUISITI 5](#_Toc64675348)

[1.1. Analisi dei requisiti 5](#_Toc64675349)

[1.2. Diagramma dei Casi d’uso 6](#_Toc64675350)

[1.3. Modello dei casi d’uso in formato dettagliato 7](#_Toc64675351)

[UC1: Registrazione studente 7](#_Toc64675352)

[UC8: Registrazione professore 9](#_Toc64675353)

[UC5: Creazione ricevimento 12](#_Toc64675354)

[UC3: Prenotazione di un ricevimento 14](#_Toc64675355)

[UC6: Creazione esame 16](#_Toc64675356)

[UC2: Prenotazione per un esame 18](#_Toc64675357)

[1.4. Altri casi d’uso in formato breve 20](#_Toc64675358)

[UC4: Gestione prenotazioni 20](#_Toc64675359)

[UC7: Gestione eventi 20](#_Toc64675360)

[1.5. Regole di Dominio 21](#_Toc64675361)

[1.6. Glossario 22](#_Toc64675362)

[1.7. Analisi Iterativa 22](#_Toc64675363)

[1a Iterazione 23](#_Toc64675364)

[2a Iterazione 23](#_Toc64675365)

[3a Iterazione 23](#_Toc64675366)

[4a Iterazione 23](#_Toc64675367)

[5a Iterazione 23](#_Toc64675368)

[6a Iterazione 23](#_Toc64675369)

[1.8. Modello di dominio 24](#_Toc64675370)

[1.9. Diagrammi di sequenza di sistema e contratti delle operazioni 25](#_Toc64675371)

[Diagramma di sequenza di Sistema UC1 25](#_Toc64675372)

[Contratti delle operazioni UC1 25](#_Toc64675373)

[Diagramma di sequenza di sistema UC8 27](#_Toc64675374)

[Contratti delle operazioni UC8 27](#_Toc64675375)

[Contratti delle operazioni UC5 29](#_Toc64675376)

[Contratti delle operazioni UC3 31](#_Toc64675377)

[Diagramma di sequenza di sistema UC6 33](#_Toc64675378)

[Contratti delle operazioni  UC6 33](#_Toc64675379)

[Contratti delle operazioni UC2 35](#_Toc64675380)

[2. PROGETTAZIONE 37](#_Toc64675381)

[2.1. Contratti delle operazioni UC2 38](#_Toc64675382)

[UC1 38](#_Toc64675383)

[UC8 39](#_Toc64675384)

[UC5 41](#_Toc64675385)

[UC6 42](#_Toc64675386)

[UC2 44](#_Toc64675387)

[UC3 46](#_Toc64675388)

[UC7 47](#_Toc64675389)

[UC4 49](#_Toc64675390)

[3. TESTING 51](#_Toc64675391)

[3.1. Test Iterazione 1 e 2 51](#_Toc64675392)

[TestRegistraStudente 51](#_Toc64675393)

[TestRegistraProfessore 53](#_Toc64675394)

[3.2. Test Iterazione 3 e 4 54](#_Toc64675395)

[TestCreaEsame 55](#_Toc64675396)

[TestCreaRicevimento 56](#_Toc64675397)

[3.3. Test Iterazione 5 57](#_Toc64675398)

[TestPrenotaEsame 58](#_Toc64675399)

[TestPrenotaRicevimento 59](#_Toc64675400)

[3.4. Test Iterazione 6 60](#_Toc64675401)

[TestGestioneEsameProfessore 60](#_Toc64675402)

[TestGestioneEsameStudente 62](#_Toc64675403)

[TestGestioneRicevimentoProfessore 63](#_Toc64675404)

[TestGestioneEsameStudente 64](#_Toc64675405)

# IDEAZIONE E ANALISI DEI REQUISITI

In UP, Unified Process, l’ideazione è la fase iniziale di creazione di un progetto, in cui si stabilisce una visione comune e di base del progetto stesso, andando ad evidenziare alcuni dei requisiti del progetto. Con requisito intendiamo una capacità o una condizione a cui il sistema deve essere essere conforme per offrire un determinato servizio, andando a considerare inoltre una stima dei tempi e delle risorse necessarie per la realizzazione. All’interno di questa fase sono stati realizzati: il modello dei casi d’uso, il diagramma dei casi d’uso e il glossario.

## Analisi dei requisiti

UniBook offre un servizio di gestione esami e ricevimenti per studenti e professori universitari. Gli studenti al momento della registrazione devono inserire l'università, il corso di studi ai quali sono iscritti mentre i professori devono inserire l'università presso la quale lavorano. UniBook garantisce:

1. La prenotazione e la gestione delle prenotazioni di un esame o di un ricevimento da parte degli studenti.
2. L'organizzazione e la gestione di un esame o di un ricevimento da parte di un professore.
3. La visualizzazione di tutti gli esami di una data materia e di un dato professore.
4. La visualizzazione di tutti i ricevimenti organizzati da un dato professore.
5. La visualizzazione di una lista di studenti prenotati ad un ricevimento o ad un esame.
6. Una gestione dinamica dell'interfaccia in base all'utenza registrata.
7. Una gestione degli avvisi su tutta l'utenza.

Di seguito viene proposto una serie di casi d'uso riassunti nella seguente tabella, elencando specificatamente attori, obiettivi e casi d'uso associati.

|  |  |  |
| --- | --- | --- |
| Attore | Obiettivo | Casi d’uso |
| Studente | Registrazione dello studente al servizio Unibook | **UC2: Registrazione studente** |
| Studente | Prenotare un esame in un data prestabilita da un professore | **UC3: Prenotazione esame** |
| Studente | Prenotare un ricevimento in base agli orari scelti dal professore | **UC4: Prenotazione ricevimento** |
| Studente | Gestire le prenotazione di  esami  e di ricevimenti | **UC5: Gestione prenotazioni** |
| Professore | Creare un ricevimento in uno specifico giorno | **UC6: Creazione ricevimento** |
| Professore | Rendere disponibile la prenotazione di un esame in una specifica data | **UC7: Creazione esame** |
| Professore | Gestire, modificare, eliminare, ordinare prenotazioni e ricevimenti | **UC8: Gestione eventi** |
| Professore | Registrazione al servizio unibook | **UC9: Registrazione professore** |

## Diagramma dei Casi d’uso

Viene riportato in basso il diagramma dei casi d’uso

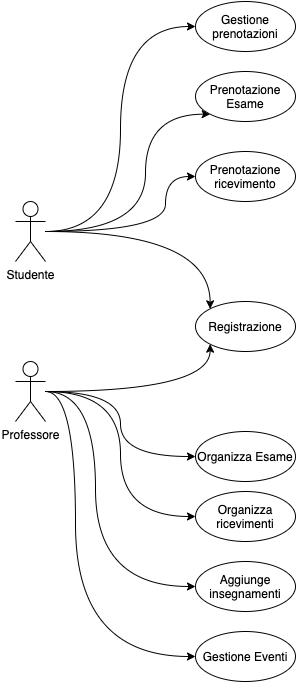


Figura 1: diagramma dei casi d’uso.

## Modello dei casi d’uso in formato dettagliato

Si considerino i seguenti casi d’uso in formato dettagliato, di cui verrà posto in evidenza lo scenario principale ed eventuali scenari alternativi.

### UC1: Registrazione studente

***Portata***: Applicazione UniBook

**Livello**: Obiettivo studente

***Attore primario***: Studente universitario.

***Parti interessate e interessi***:

*Studente*: vuole che l’applicazione consenta una registrazione rapida e sicura.

***Pre-condizioni***: L’applicazione deve essere resa disponibile ad un potenziale utente. L’applicazione UniBook deve interfacciarsi con un Verificatore esterno.

***Garanzia di successo (o post-condizioni)***: Lo studente viene registrato all’interno dell’applicazione.

***Scenario principale di successo***:

1. L'utente apre l'applicazione UniBook.
2. L'utente clicca l'opzione "Registrati qui se sei uno studente".
3. Il Sistema, mostra un form all'utente in cui può immettere i suoi dati.
4. L'utente inserisce tutti i suoi dati: nome, cognome, anno di nascità, l'email, la matricola universitaria, il codice fiscale dell'università e il corso di studi di appartenenza e una password di 8 lettere.
5. Il sistema mostra una schermata di riepilogo dati dopo aver verificato che lo studente non sia già iscritto ad UniBook.
6. Dopo che l'utente ha confermato i dati, Unibook verifica la correttezza dei dati inseriti tramite un Verificatore che verificherà se lo studente è veramente iscritto all'Università e al Corso di studi inserito.
7. Una volta verificati i dati, l'applicazione procede alla registrazione alla piattaforma.

***Estensioni (o Scenari alternativi)***:

1. L'utente leggendo i dati di riepilogo si accorge di un errore, quindi deve modificarlo:
   1. L'utente torna al punto 4 dello scenario principale di successo.
   2. Una volta corretti i dati, l'utente conferma la modifica effettuata e torna al punto 5 dello scenario principale di successo.
2. Nel momento in cui l'utente si sta registrando, la connessione ad internet viene a mancare:
   1. Il sistema mostra a schermo un alert in cui evidenzia il problema della mancata connessione internet.
   2. I dati inseriti dall'utente vengono salvati automaticamente, in modo tale che essi non debbano essere riscritti nuovamente.
   3. Nel momento in cui ritorna la connessione ad internet, la bozza viene pubblicata.
3. Il verificatore notifica ad UniBook la non validit√† del codice fiscale dell'Universit√† o del corso di studi inserito dall'utente:
   1. Il sistema notifica tramite schermata all'utente la non correttezza dei dati inseriti.
   2. L'utente torna al punto 4 dello scenario principale di successo e inserisce nuovamente il codice fiscale dell'Universit√† di appartenenza e del corso di studi di appartenenza.

***Requisiti speciali***:

Lo studente una volta confermati i dati deve poter essere registrato all’applicazione entro e non oltre i 30 secondi dalla conferma. Altrimenti l’applicazione notifica via email un problema nella registrazione.

***Elenco delle varianti tecnologiche e dei dati***:

Lo studente deve essere identificato dalla sua matricola.

***Frequenza di ripetizione***: Tale operazione può essere ripetuta una volta per ogni utente che ha l’intenzione di registrarsi.

### UC8: Registrazione professore

***Portata***: Applicazione UniBook

**Livello**: Obiettivo professore

***Attore primario***: Professore universitario.

***Parti interessate e interessi***:

*Professore*: vuole che l’applicazione consenta una registrazione rapida e sicura.

***Pre-condizioni***: L’applicazione deve essere resa disponibile ad un potenziale utente. L’applicazione UniBook deve interfacciarsi con un Verificatore esterno.

***Garanzia di successo (o post-condizioni)***: Il professore viene registrato all’interno dell’applicazione.

***Scenario principale di successo***:

1. L'utente apre l'applicazione UniBook.
2. L'utente clicca l'opzione "Registrati qui se sei un professore".
3. Il Sistema, mostra un form all'utente in cui può immettere i suoi dati.
4. L'utente inserisce tutti i suoi dati: nome, cognome, anno di nascità, l'email, la matricola universitaria, il codice fiscale dell'Università nella quale egli insegna e la password di 8 lettere utilizzata per accedere all'applicazione.
5. Il sistema mostra una schermata di riepilogo dati dopo aver verificato che il professore non sia già iscritto ad UniBook.
6. Dopo aver riletto i dati il professore procede alla conferma. UniBook confermerà se il Professore è presente realmente come docente dell'Università inserita.
7. UniBook mostrerà al professore un form in cui inserire tutte le materie da lui insegnate.
8. Una volta verificati i dati, ossia se le materie sono realmente presenti all'interno dell'Università e insegnate dal Docente l'applicazione procede alla conferma. L'applicazione procede alla registrazione alla piattaforma.

***Estensioni (o Scenari alternativi)***:

1. L'utente leggendo i dati di riepilogo nel punto 5 si accorge di un errore, quindi deve modificarlo:
   1. L'utente torna al punto 4 dello scenario principale di successo.
   2. Una volta corretti i dati, l'utente conferma la modifica effettuata e torna al punto 5 dello scenario principale di successo.
2. Nel momento in cui l'utente si sta registrando, la connessione ad internet viene a mancare:
   1. Il sistema mostra a schermo un alert in cui evidenzia il problema della mancata connessione internet.
   2. I dati inseriti dall'utente vengono salvati automaticamente, in modo tale che essi non debbano essere riscritti nuovamente.
   3. Nel momento in cui ritorna la connessione ad internet, la bozza viene pubblicata.
3. Il verificatore notifica ad UniBook la non validità del codice fiscale dell'Università inserito dall'Utente:
   1. Il sistema notifica tramite schermata all'utente la non correttezza dei dati inseriti.
   2. L'utente torna al punto 4 dello scenario principale di successo e inserisce nuovamente il codice fiscale dell'Universit√† di appartenenza.
4. Il verificatore notifica ad UniBook un errore con l'inserimento di una materia o più materie inserite dall'utente:
   1. UniBook notifica tramite schermata all'utente la non correttezza dei dati inseriti riguardo una o più materie.
   2. L'utente torna al punto 7 dello scenario principale di successo e inserisce nuovamente i dati.

***Requisiti speciali***:

Il professore una volta confermati i dati deve poter essere registrato all’applicazione entro e non oltre i 15 secondi dalla conferma. Altrimenti l’applicazione notifica via email un problema nella registrazione.

***Elenco delle varianti tecnologiche e dei dati***:

Il professore deve essere identificato univocamente dalla sua matricola.

***Frequenza di ripetizione***: Tale operazione può essere ripetuta una volta per ogni utente che ha l’intenzione di registrarsi.

### UC5: Creazione ricevimento

***Portata***: Applicazione UniBook

**Livello**: Obiettivo professore

***Attore primario***: Professore universitario

***Parti interessate e interessi***:

*Professore*: vuole un inserimento preciso ed intuitivo dei dati relativi al ricevimento. Inoltre, vuole essere certo che i propri ricevimenti non entrino in conflitto con altri eventi istituzionali, ricevimenti o esami, da lui stesso organizzati.

*Studente*: vuole che i dati dei ricevimenti vengano scritti nel modo più chiaro possibile, affinché non ci siano equivoci.

***Pre-condizioni***: Il professore deve essere registrato al servizio e deve aver effettuato l’accesso alla piattaforma.

***Garanzia di successo (o post-condizioni)***: Il ricevimento organizzato deve essere salvato e soprattutto disponibile nell’ immediato a tutti gli studenti che vogliono visualizzare i ricevimenti di un dato professore.

***Scenario principale di successo***:

1. Il Professore apre l'applicazione UniBook.
2. Il Professore inserisce le proprie credenziali.
3. Il Sistema, una volta che ha riconosciuto il Professore, apre la schermata iniziale.
4. Il Professore clicca l'opzione 'Inserimento nuovo Ricevimento'.
5. Il Professore inserisce tutti i dati relativi al nuovo Ricevimento: codice identificativo, orario di inizio e di fine, deadline della prenotazione del ricevimento, luogo del Ricevimento e il numero massimo di partecipanti.
6. Il Professore una volta controllati i dati, procede alla conferma di questi. Il Sistema si occupa di inserire il nuovo Ricevimento e invia un'email di conferma al Professore.
7. Il Professore una volta ricevuta l'email, chiude l'applicazione

***Estensioni (o Scenari alternativi)***:

1. Il professore leggendo i dati di riepilogo della creazione del ricevimento si accorge di un errore, quindi deve modificarlo:
   1. Il professore torna al punto 5 dello scenario principale di successo.
   2. Una volta corretti i dati, il professore conferma la modifica effettuata e torna al punto 6 dello scenario principale di successo.
2. Nel momento in cui il professore organizza il ricevimento, la connessione ad internet viene a mancare:
   1. Il sistema mostra a schermo un alert in cui evidenzia il problema della mancata connessione internet.
   2. I dati inseriti dal professore vengono salvati automaticamente, in modo tale che essi non debbano essere riscritti nuovamente.
   3. Nel momento in cui ritorna la connessione ad internet, la bozza viene pubblicata.
3. Il sistema notifica al professore la presenza di un ricevimento o di un esame già esistente, avente la stessa data e la stessa fascia oraria di quello appena inserito:
   1. Il professore torna al punto 5 dello scenario principale di successo e corregge i dati.

***Requisiti speciali***:

* Il ricevimento deve essere disponibile entro e non oltre 5 secondi dal salvataggio. Se il ricevimento non risulta disponibile entro il limite richiesto, il sistema lancia un alert e invia una notifica email al professore.

***Elenco delle varianti tecnologiche e dei dati***:

* Il ricevimento deve avere un proprio codice identificativo, in modo tale da differenziarlo dagli altri.
* Ogni ricevimento deve essere “firmato” dal professore.

***Frequenza di ripetizione***: Tale operazione può essere ripetuta più volte dal professore.

### UC3: Prenotazione di un ricevimento

***Portata***: applicazione UniBook

**Livello**: livello Studente.

***Attore primario***: studente universitario.

***Parti interessate e interessi***:

*Professore*: Il professore deve essere notificato nel momento in cui uno studente si prenota a quel preciso ricevimento

*Studente*: Lo studente deve poter essere in grado di prenotare il ricevimento in maniera intuitiva e soprattutto deve poter avere la possibilità di poter disdire il ricevimento entro un giorno fissato o di default o dal professore

***Pre-condizioni***: Lo studente deve essere sia registrato alla piattaforma sia loggato per poter effettuare eventuali prenotazioni a dei ricevimenti.

***Garanzia di successo (o post-condizioni)***: Nel momento in cui lo studente effettua la prenotazione, il professore verrà avvertito tramite notifica su UniBook.

***Scenario principale di successo***:

1. Lo Studente apre UniBook.
2. Lo Studente inserisce le proprie credenziali di accesso: la sua matricola e il codice identificativo dell’Università al quale appartiene.
3. Il Sistema dopo aver verificato lo studente, apre la schermata iniziale.
4. Lo Studente clicca sull’opzione ‘Prenota un Ricevimento’. Inserisce il nome e cognome del Professore con il quale effettuare un Ricevimento.
5. Il Sistema mostra a schermo i ricevimenti aperti e disponibili del professore allo studente.
6. Lo Studente una volta visualizzati i ricevimenti disponibili tra questi ne sceglie uno.
7. Una volta confermata la prenotazione, il Sistema si occupa di inserire lo studente nell’elenco degli Studenti di quel dato ricevimento e invia una email di conferma allo studente
8. Lo Studente una volta ricevuta conferma via email, chiude l’applicazione.

***Estensioni (o Scenari alternativi)***:

1. Lo studente, una volta visionato i dati di riepilogo relativi alla

prenotazione del ricevimento, si accorge di essersi prenotato in una data diversa:

* 1. Lo studente ritorna al punto 5 dello scenario principale di successo.

***Requisiti speciali***: La prenotazione del ricevimento dello studente deve avvenire senza alcun tipo di ritardo nella ricezione. Inoltre, la prenotazione del ricevimento deve essere notificata al professore tramite l’applicazione e non via email.

***Elenco delle varianti tecnologiche e dei dati***:

* Lo studente deve essere identificato tramite la sua matricola, il suo nome e il suo cognome

***Frequenza di ripetizione***: Lo studente è in grado di potersi prenotare a più ricevimenti che siano tra loro coerenti con l’orario di inizio e di fine.

### UC6: Creazione esame

***Portata***: applicazione UniBook

**Livello**: obiettivo professore

***Attore primario***: professore universitario

***Parti interessate e interessi***:

*Professore*: vuole un inserimento preciso ed intuitivo dei dati relativi all’esame, in modo tale da non dover perdere troppo tempo. Inoltre vuole essere certo che i propri esami non entrino in conflitto con altri eventi istituzionali o con altri ricevimenti o esami.

*Studente*: vuole che i dati dell’esame vengano scritti nel modo più chiaro possibile, affinché non ci siano equivoci.

***Pre-condizioni***: Il professore deve essere sia registrato al servizio sia loggato alla piattaforma.

***Garanzia di successo (o post-condizioni)***: L’esame che viene creato deve essere salvato e soprattutto visibile nell’immediato a tutti gli studenti che partecipano a quella data materia associata all’esame.

***Scenario principale di successo***:

1. Il professore apre l'applicazione UniBook.
2. Il professore inserisce le proprie credenziali.
3. Il Sistema riconosciuto il professore apre la schermata iniziale.
4. Il professore clicca l'opzione Inserimento nuovo Esame.
5. Il sistema apre un form per l'inserimento di tutti i dati relativi agli esami.
6. Il professore inserisce tutti i dati relativi al nuovo esame: codice identificativo, orario di inizio e di fine, luogo, termine ultimo per registrarsi all'esame il numero massimo di partecipanti e la modalit√†.
7. Il professore una volta controllati i dati, procede alla conferma di questi.
8. L'applicazione mostra un form in cui chiede al professore di inserire la materia relativa all'esame.
9. Il professore inserisce uno dei codici univoci della materia da lui insegnate.
10. Il professore una volta controllati i dati, procede alla conferma di questi tramite firma. Il Sistema si occupa di inserire il nuovo Esame. Il sistema notifica al professore l'avvenuto inserimento tramite email.
11. Il professore una volta ricevuta l'email, chiude l'applicazione.

***Estensioni (o Scenari alternativi)***:

1. Il professore leggendo i dati di riepilogo della creazione dell’esame si accorge di un errore, quindi deve modificarlo:
   1. Il professore torna al punto 6 dello scenario principale di successo.
   2. Una volta corretti i dati, il professore conferma la modifica effettuata e torna al punto 7 dello scenario principale di successo.
2. Nel momento in cui il professore organizza l’esame, la connessione ad Internet viene a mancare.
   1. Il sistema mostra a schermo un alert in cui evidenzia il problema della mancata connessione internet.
   2. I dati inseriti dal professore vengono salvati automaticamente come una bozza, in modo tale che essi non debbano essere riscritti nuovamente.
   3. Nel momento in cui ritorna la connessione ad Internet, la bozza viene pubblicata.
3. Il sistema notifica al professore la presenza di un ricevimento o di un esame già esistent, avente la stessa data e la stessa fascia oraria di quello appena inserito:
   1. Il professore torna al punto 6 dello scenario principale di successo e corregge i dati.

***Requisiti speciali***: L’esame dovrà essere visibile a tutti gli studenti entro cinque secondi dal salvataggio. Se così non fosse il Sistema notificherebbe al professore con un’alert e via email un problema con l’inserimento del suddetto esame.

***Elenco delle varianti tecnologiche e dei dati***:

* L’esame deve avere un proprio codice identificativo, in modo tale da differenziarlo dagli altri.
* Ogni esame deve essere “firmato” dal professore.

***Frequenza di ripetizione:*** l’operazionepotrà essere ripetuta ma non ci potranno essere esami della stessa materia organizzati dallo stesso professore nella stessa data.

### UC2: Prenotazione per un esame

***Portata***: applicazione UniBook.

**Livello**: livello studente.

***Attore primario***: studente universitario.

***Parti interessate e interessi***:

*Professore*: il professore deve essere notificato nel momento in cui uno studente si prenota a quel preciso esame.

*Studente*: lo studente deve poter essere in grado di prenotare l’esame in maniera intuitiva e soprattutto deve poter avere la possibilità di poter disdire la propria partecipazione a quell’esame entro una soglia fissata fissata di default o dal professore.

***Pre-condizioni***: Lo studente deve essere sia registrato alla piattaforma sia loggato per poter effettuare eventuali prenotazioni a dei ricevimenti.

***Garanzia di successo (o post-condizioni)***: Una volta che lo studente ha effettuato la prenotazione all’esame, verra notificata la partecipazione dello Studente sia allo studente, sia al Professore che ha organizzato quell’esame, tramite notifica dell’applicazione e via email.

***Scenario principale di successo***:

1. Lo studente apre UniBook.
2. Lo studente inserisce le proprie credenziali di accesso: la sua matricola e il codice identificativo dell’università alla quale egli appartiene.
3. Il Sistema dopo aver verificato lo studente apre la schermata iniziale.
4. Lo studente clicca sull’opzione prenota un esame. Inserisce la materia della quale vuole effettuare un esame.
5. Il Sistema mostra a schermo gli esami aperti e disponibili della materia allo studente.
6. Lo studente una volta visualizzati gli esami disponibili tra questi sceglie quello del professore con cui segue la materia, inserendo il proprio nome, il proprio cognome e la matricola.
7. Una volta confermati i dati, il Sistema si occupa di inserire lo studente nell’elenco di studenti per quel dato esame e invia una email allo studente e al Professore utilizzando nel caso dello studente quella di registrazione.
8. Lo studente una volta ricevuta conferma via email, chiude l’applicazione.

***Estensioni (o Scenari alternativi)***:

1. Lo studente leggendo i dati di riepilogo della prenotazione dell’esame si accorge del fatto che abbia prenotato un appello di una data differente:
   1. Lo studente torna al punto 5 dello scenario principale di successo.
   2. Una volta corretti i dati, lo studente procede nella fase di prenotazione dell’esame

***Requisiti speciali***: La prenotazione dell’esame da parte dello studente deve avvenire senza alcun tipo di ritardo nella ricezione. Inoltre, la prenotazione dell’esame deve essere notificata al professore tramite l’applicazione e non via email.

***Elenco delle varianti tecnologiche e dei dati***:

* Lo studente deve essere identificato tramite la sua matricola, il suo nome e il suo cognome.

***Frequenza di ripetizione***: Lo studente è in grado di potersi prenotare a più esami che siano tra loro coerenti con l’orario di inizio e di fine.

## Altri casi d’uso in formato breve

Vengono riportati in questo paragrafo i restanti casi d’uso in formato breve.

### UC4: Gestione prenotazioni

Attore primario: Studente

All’interno di questo caso d’uso lo studente può visualizzare tutte le prenotazioni da lui effettuate, che siano ricevimenti o esami, ed eventualmente eliminarle. In casodi cancellazione di una prenotazione di un esame o di un ricevimento, il professore che organizza quell’evento viene notificato dall’applicazione tramite email.

### UC7: Gestione eventi

Attore primario: Professore

All’interno di questo caso d’uso il professore può visualizzare tutti gli eventi da lui creati, che siano ricevimenti o esami ed eventualmente modificarli o eliminarli. In caso di modifica o di cancellazione di un esame o di un ricevimento, ciascun studente che si è iscritto a quell’evento viene notificato dall’applicazione tramite email dell’avvenuta modifica o cancellazione di quell’evento.

## Regole di Dominio

L’applicazione UniBook durante la registrazione degli utenti, studenti o professori deve poter verificare che i dati inseriti siano corretti. L’Università viene identificata da un codice fiscale univoco, costituito da una serie di undici caratteri numerici. Il corso di studio da un codice univoco numerico di massimo undici caratteri, mentre la materia viene identificata da un codice univoco anch’esso numerico di sei caratteri. Lo studente viene identificato da una matricola, assegnata dal Corso di Studio alla quale è iscritto, che è un codice alfanumerico la cui grandezza dipende dal Corso di studio di competenza. Ogni studente iscritto ad Unibook viene identificato da tale matricola che risulta univoca. Allo stesso modo un professore sarà identificato da una matricola personale. Ogni utente, che sia professore o studente al momento della registrazione inserisce una password di 8 caratteri necessaria per verificare la sua identità al momento del login.

L’applicazione si interfaccia con un Sistema esterno, che si occupa di andare a verificare la correttezza dei dati inseriti dall’utente al momento della registrazione al servizio.

Ricevimenti ed esami al momento della creazione assumono un ID, di riferimento. Non vi possono essere conflitti legati ad eventi che posseggono lo stesso ID. Inoltre, non ci potranno essere esami e ricevimenti organizzati dallo stesso professore durante la stessa data e la stessa fascia oraria.

Per gestire la persistenza dei dati l’applicazione sfrutta un database interno. Le tecnologie utilizzate all’interno del progetto e quindi dell’applicazione sono di tipo software proprietario.

## Glossario

Prenotazione: termine con cui ci si riferisce alla possibilità da parte di uno studente di poter iscriversi ad un determinato esame o ricevimento.

Ricevimento: termine con cui si riferisca ad un incontro tra professore e studente o gruppo di studenti affinchè gli studenti possano confrontarsi col professore in merito ad argomenti universitari.

Esame: prova a cui vengono sottoposti gli studenti per verificare il grado di preparazione.

Corso di Studi: ciclo di studi alla cui conclusione si ottiene un titolo di studio. In questo documento, il termine si applica ai Corsi di Laurea, ai Corsi di Laurea Magistrale e ai Corsi di Laurea Magistrale a Ciclo Unico e, per il terzo ciclo, ai Corsi di Specializzazione e di Dottorato di Ricerca.

Gestione: con questo termine all’interno dell’elaborato si intende una funzione di controllo, di amministrazione degli eventi organizzati nel caso dei professori, degli eventi prenotati nel caso degli studenti.

## Analisi Iterativa

Seguendo un approccio iterativo di tipo incrementale ed evolutivo, si è deciso di articolare lo sviluppo del progetto in più iterazioni, ognuna delle quali permette di analizzare uno o più scenari di casi d’uso. Le iterazioni scelte sono sei e coinvolgono tutti i casi d’uso descritti in formato dettagliato, ed anche i casi d’uso 4 e 7 descritti invece in formato breve.

### 1a Iterazione

In questa iterazione sono rappresentati lo scenario principale di successo del caso d’uso 1 e 8. In particolare viene creato un’istanza della classe Professore, e un’istanza della classe Studente.

### 2a Iterazione

In questa iterazione viene riportato lo scenario alternativo .c del caso d’uso 1 e viene inoltre riportato lo scenario alternativo .d del caso d’uso 8. Viene quindi presentato un errore nella creazione dell’istanza di uno studente e nell’inserimento di una materia da associare ad un professore al momento della registrazione di questi al servizio.

### 3a Iterazione

In questa iterazione viene valutato lo scenario principale di successo dei casi d’uso 5 e 6. Verranno infatti create: un’istanza della classe Ricevimento e un’istanza della classe Esame.

### 4a Iterazione

In questa iterazione consideriamo gli scenari alternativi .c relativi ai casi d’uso 5 e 6. Si manifesterà pertanto un problema nella creazione di un esame e di un Ricevimento di un Professore. Viene inoltre mostrato una possibile modifica ed eliminazione di un esame e di un Ricevimento da parte di un Professore così come evidenziato all’interno del caso d’uso UC7.

### 5a Iterazione

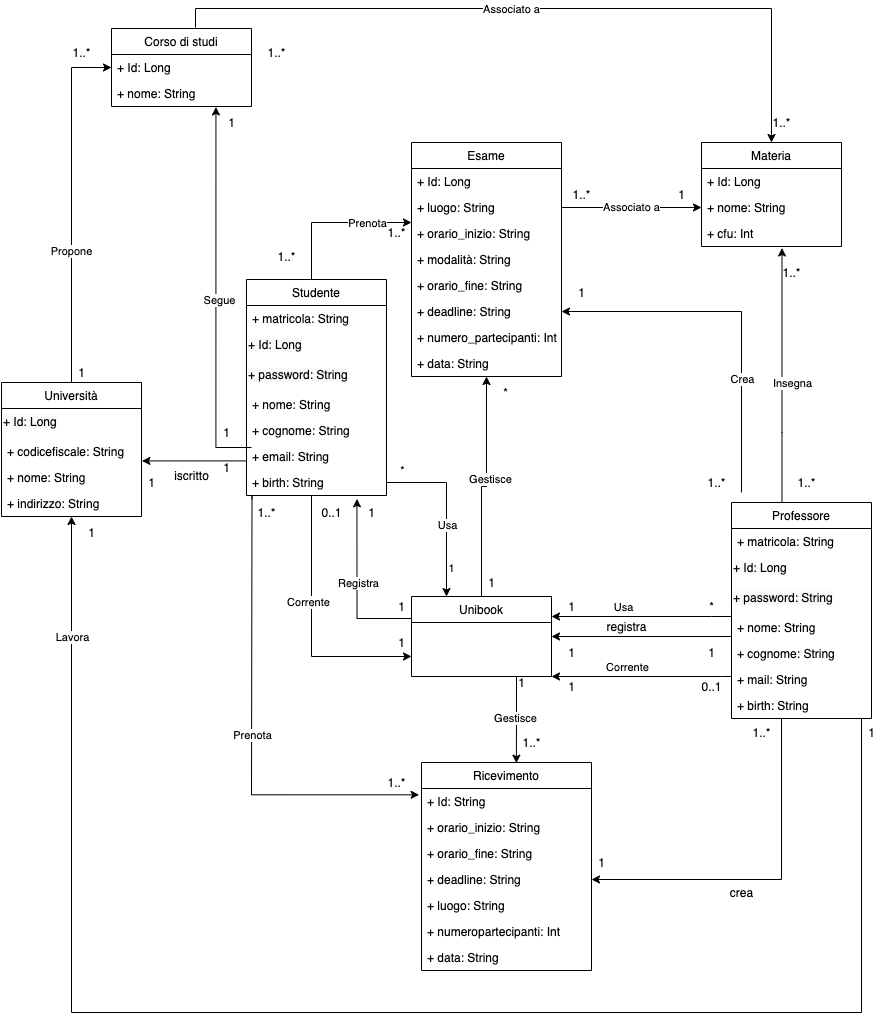
In questa iterazione consideriamo gli scenari di successo relativi ai casi d’uso 2 e 3 relativi rispettivamente alla prenotazione di un esame e di un ricevimento da parte dello studente.

### 6a Iterazione

In questa iterazione consideriamo gli scenari alternativi .a relativi ai casi d’uso 2 e 3, evidenziando un problema nella prenotazione da parte di uno studente ad un ricevimento e ad un esame. Viene inoltre mostrato un’eliminazione della prenotazione di un esame e di un Ricevimento da parte di uno studente così come evidenziato all’interno del caso d’uso 4.

## Modello di dominio

Alla fine delle sei iterazioni si è reso necessario andare a definire un modello di dominio come segue.



Sono state identificate le seguenti classi concettuali:

UniBook:  rappresenta il sistema UniBook.

Studente:  utente che utilizza UniBook per prenotare Esami o Ricevimenti.

Professore: utente che utilizza UniBook creando Esami e Ricevimenti.

Ricevimento evento gestito da UniBook.

Esame: evento gestito da UniBook.

Università: ente istituzionali in cui è iscritto lo studente e in cui lavora il Professore.

Corso di Studio; ente istituzionale che  uno studente segue.

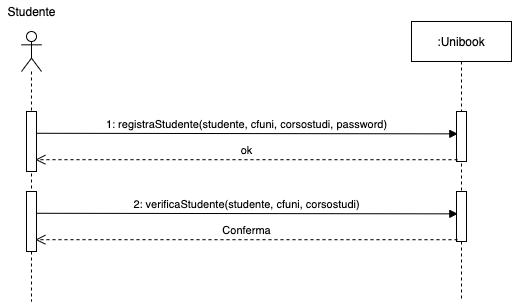
Materia: disciplina insegnata da un Professore e associata all’esame.

## Diagrammi di sequenza di sistema e contratti delle operazioni

Di seguito vedremo:

### Diagramma di sequenza di Sistema UC1

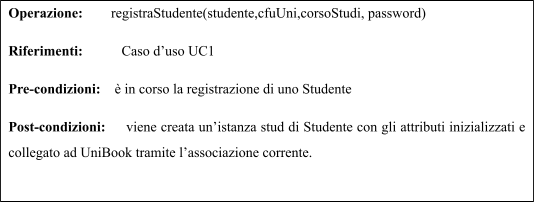
A seguire il diagramma di sequenza di sistema del caso d’uso UC1:



### Contratti delle operazioni UC1

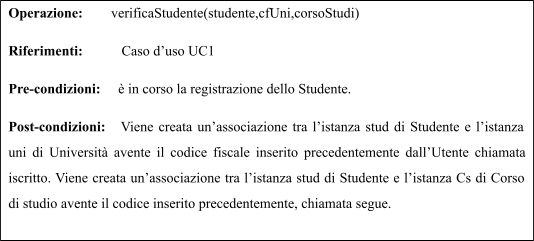
#### RegistraStudente

L’operazione di sistema *registraStudente* consente la definizione di un nuovo Studente.

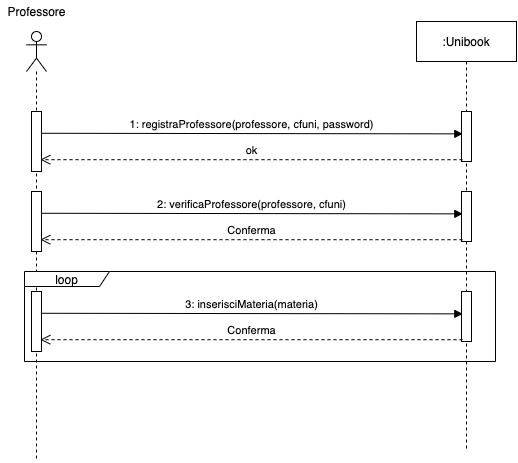


#### VerificaStudente

Con questa operazione lo studente verifica i dati inseriti relativi al Corso di Studio e Università che segue.



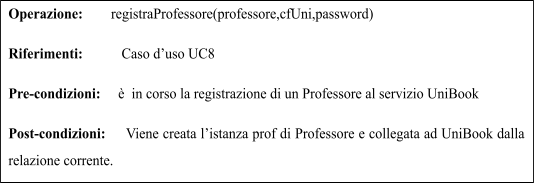
### Diagramma di sequenza di sistema UC8



### Contratti delle operazioni UC8

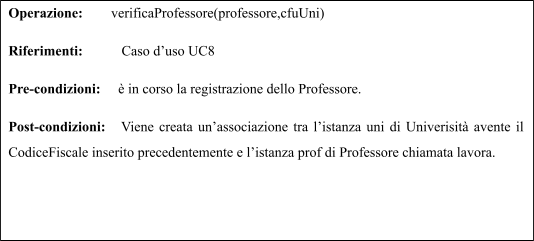
#### RegistraProfessore

L’operazione di sistema *registraProfessore* consente la definizione di un nuovo Professore.



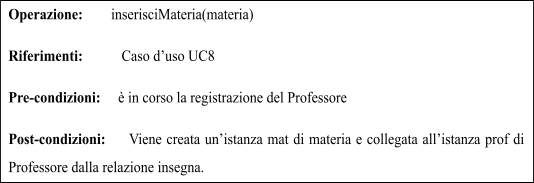
#### VerificaProfessore

Con questa operazione il professore verifica i dati inseriti.

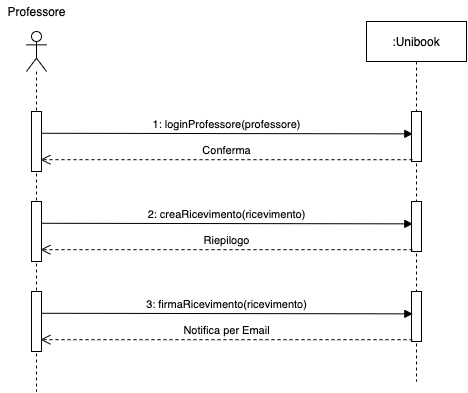


#### InserisciMateria(materia)

Con questa operazione il professore inserisce la materia da lui insegnata.



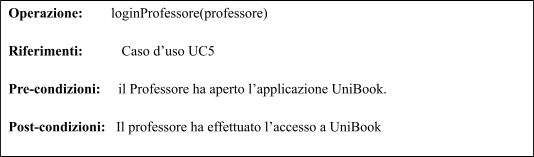
**Diagramma di sequenza di sistema  UC5**



### Contratti delle operazioni UC5

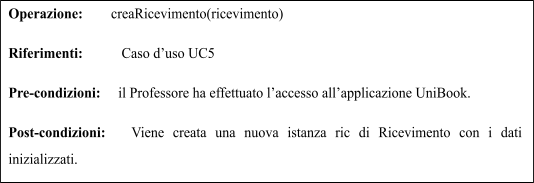
#### LoginProfessore

All’interno di questa operazione il Professore effettua il Login all’interno dell’applicazione



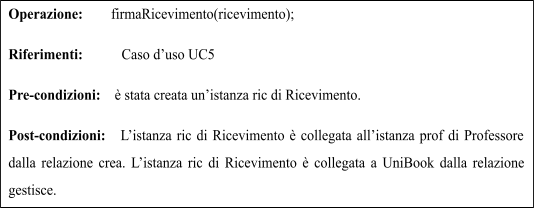
#### CreaRicevimento

All’interno di questo caso d’uso il Professore inserisce i dati del nuovo ricevimento.

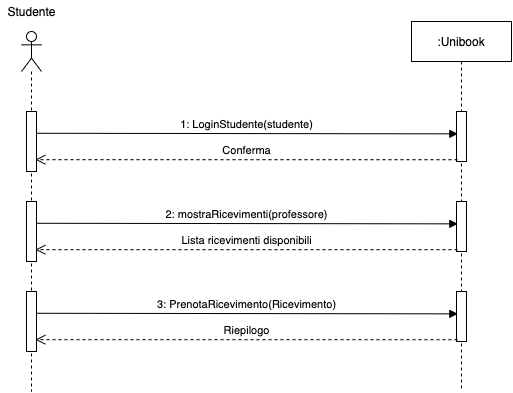


#### FirmaRicevimento

All’interno di questa operazione avviene la conferma dei dati da parte del Professore.



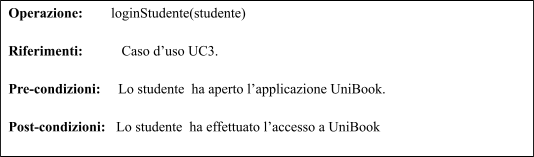
**Diagramma di sequenza di sistema  UC3**



### Contratti delle operazioni UC3

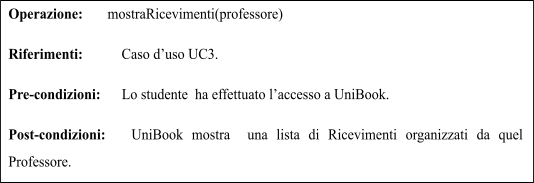
#### LoginStudente

All’interno di questa operazione lo studente effettua il Login all’interno dell’applicazione.



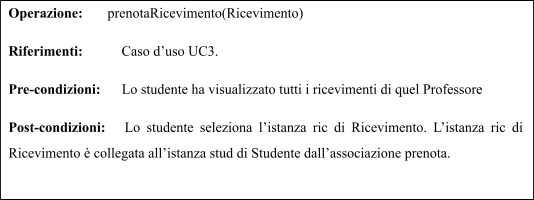
#### MostraRicevimenti

All’interno di questa operazione lo studente inserisce il Professore con il quale vuole visualizzare i ricevimenti

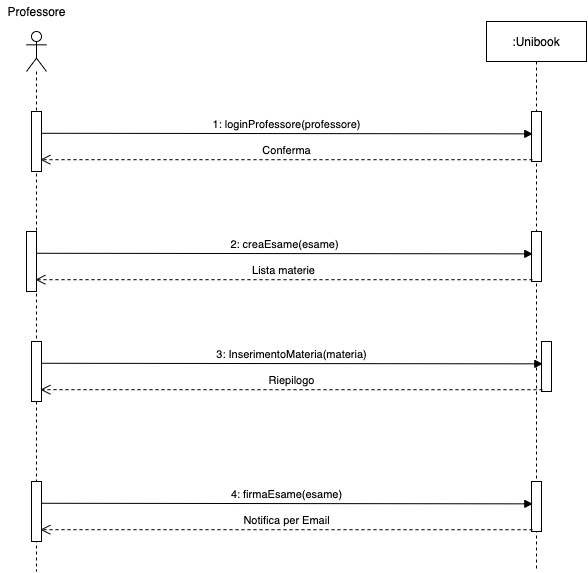


##### PrenotaRicevimento

All’interno di questa operazione lo studente seleziona il ricevimento al quale vuole partecipare.



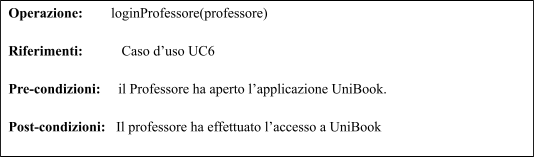
### Diagramma di sequenza di sistema UC6



### Contratti delle operazioni  UC6

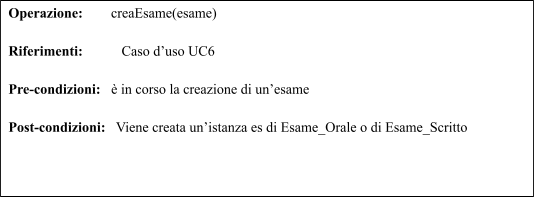
#### LoginProfessore

All’interno di questa operazione il Professore effettua il Login all’interno dell’applicazione



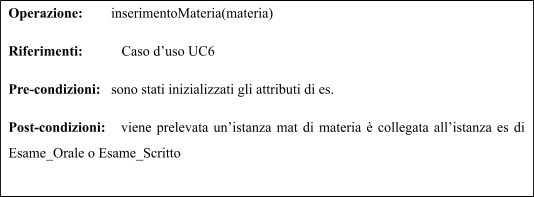
#### CreaEsame

All’interno di questa operazione il Professore inserisce i dati dell’Esame



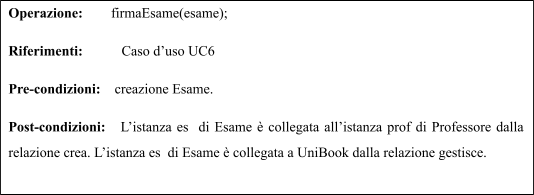
#### InserimentoMateria

All’interno di questa operazione il Professore inserisce la materia dell’Esame.

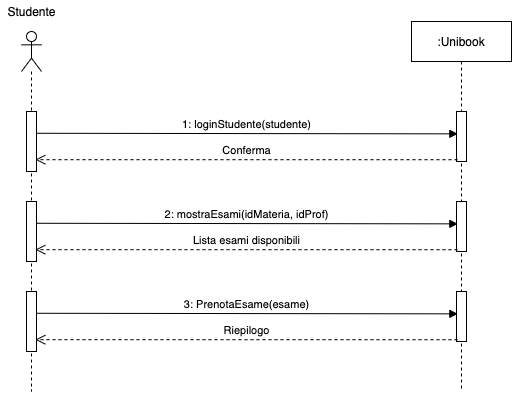


#### FirmaEsame

All’interno di questa operazione avviene la conferma dei dati da parte del Professore e la firma all’Esame appena creato



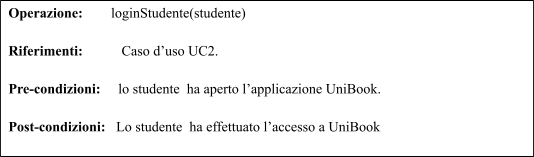
**Diagramma di sequenza di sistema  UC2**



### Contratti delle operazioni UC2

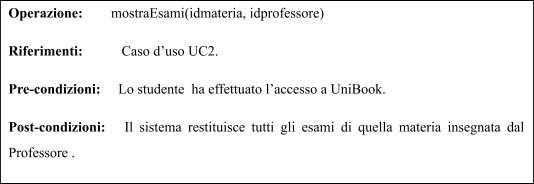
#### LoginStudente

All’interno di questa operazione lo studente effettua il Login all’interno dell’applicazione



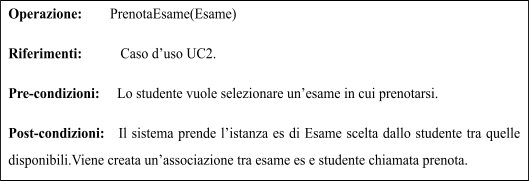
#### MostraEsami

Attraverso questa operazione lo studente seleziona la materia della quale vuole effettuare un’esame.



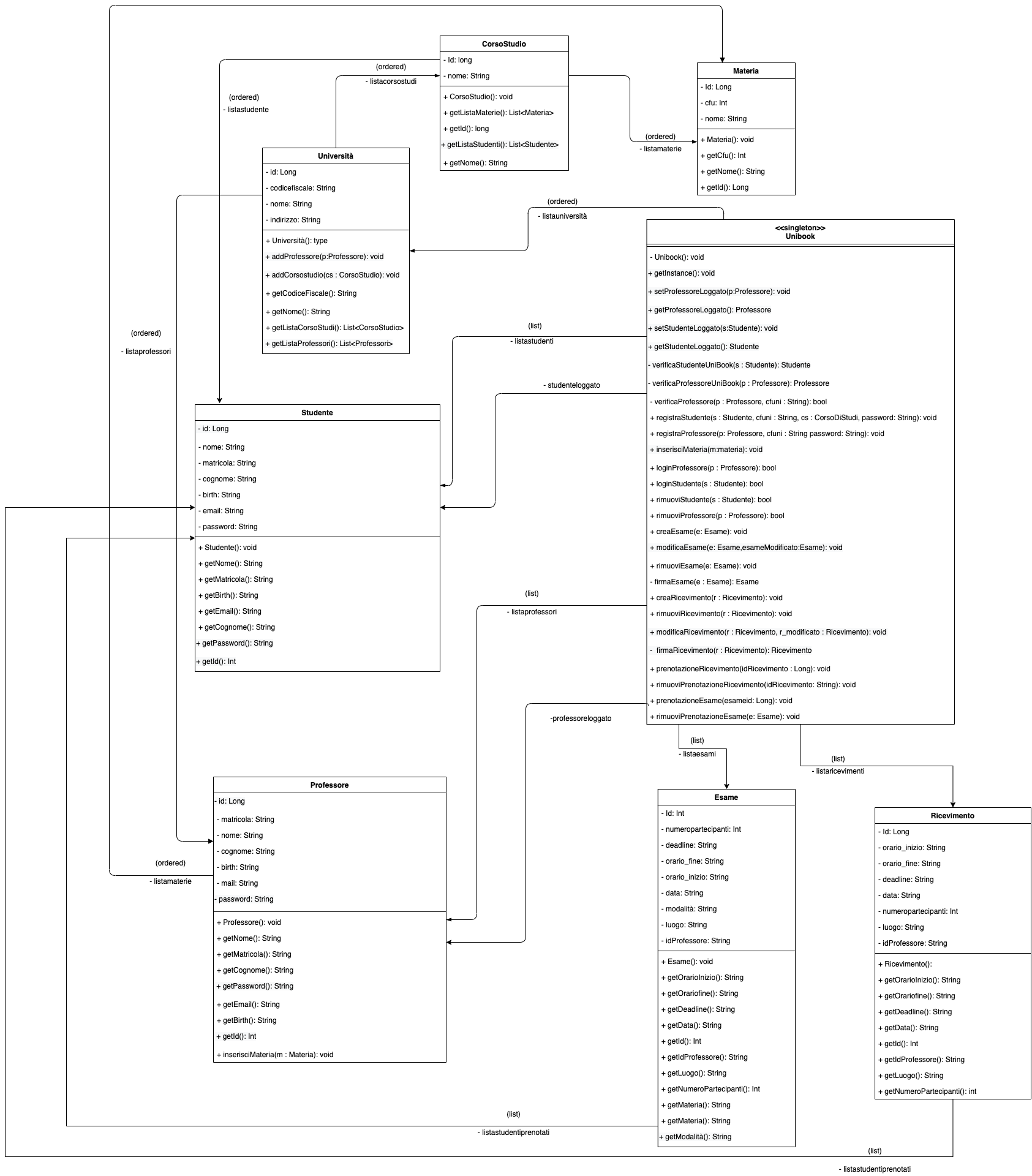
#### PrenotaEsame

Attraverso questa operazione avviene la prenotazione dell’esame selezionato dallo studente.



# PROGETTAZIONE

Durante la fase di progettazione verrà illustrato inizialmente il salto rappresentazionale dal **modello di dominio** cioè il **diagramma delle classi**, in modo tale da definire gli oggetti software, le loro responsabilità e come essi collaborino tra loro. In seguito verranno descritti i diagrammi di interazione in particolare quelli di sequenza per tutti i casi d’uso analizzati.   
Il **diagramma delle Classi** dunque sarà:



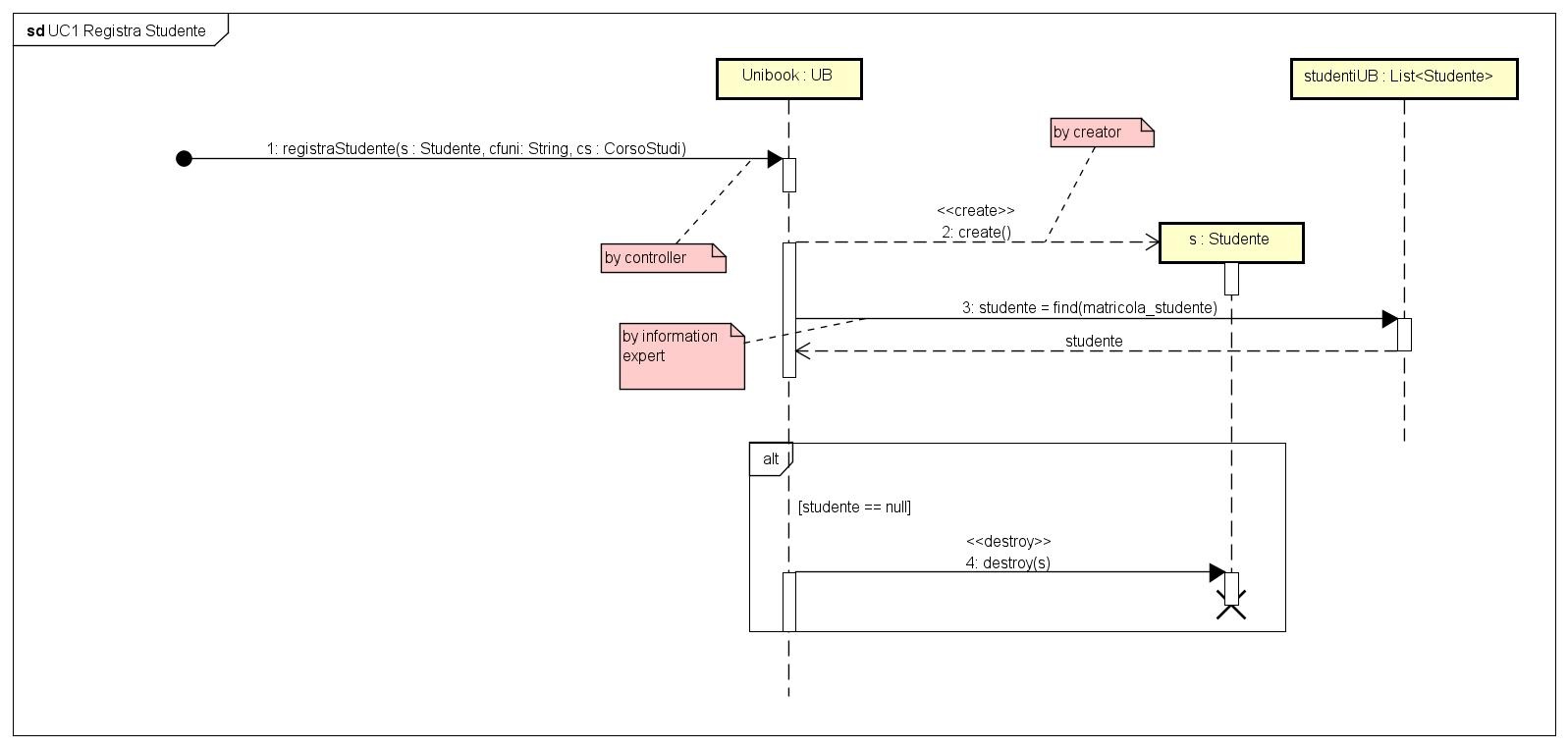
## Contratti delle operazioni UC2

Il diagramma di sequenza descrive fondamentalmente come gli oggetti (e le istanze) scambiano messaggi in un ordine particolare. Risultano inoltre utili se si desidera rappresentare graficamente operazioni complicate per renderle più comprensibili.

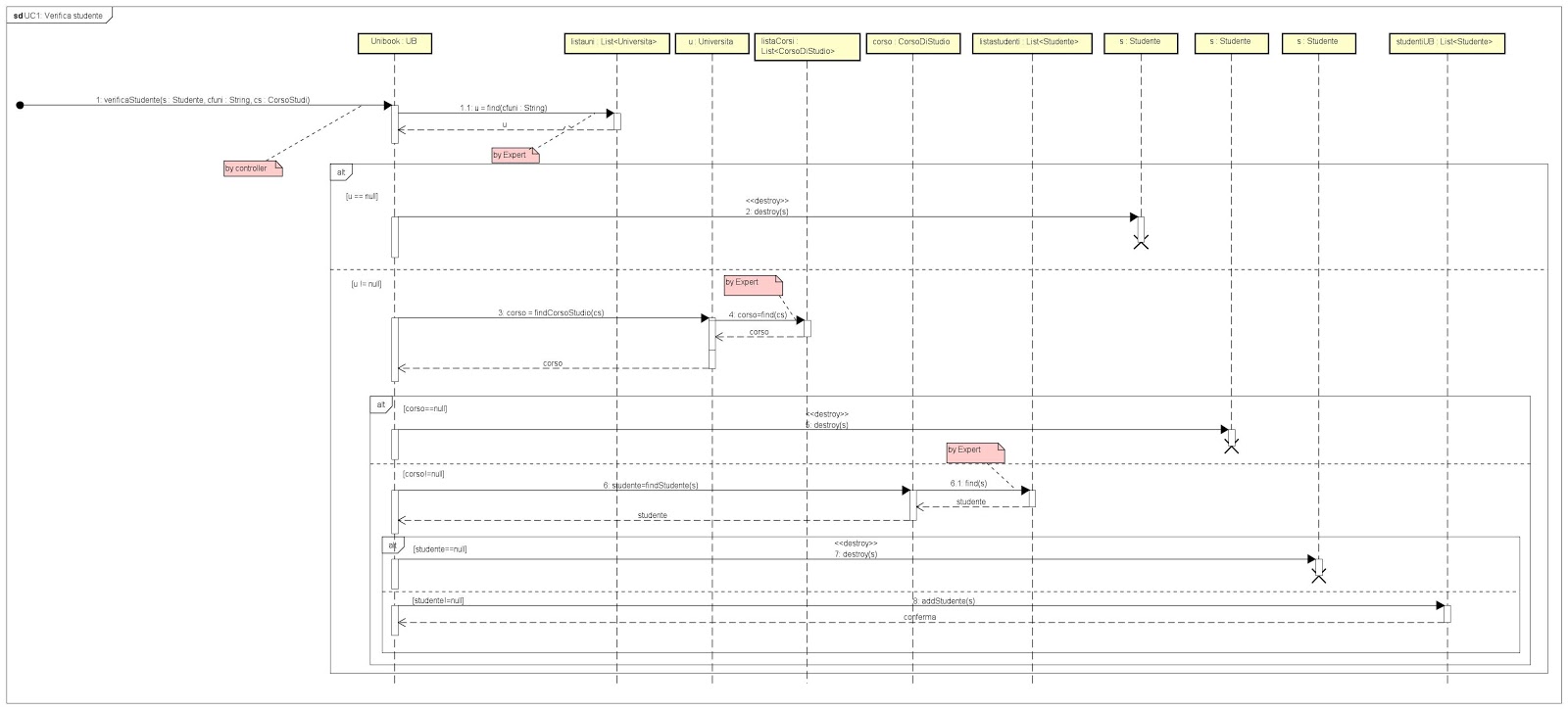
Vengono di seguito riportati i diagrammi di sequenza di tutte le operazioni di tutti i casi d’uso analizzati.

### UC1

#### RegistraStudente

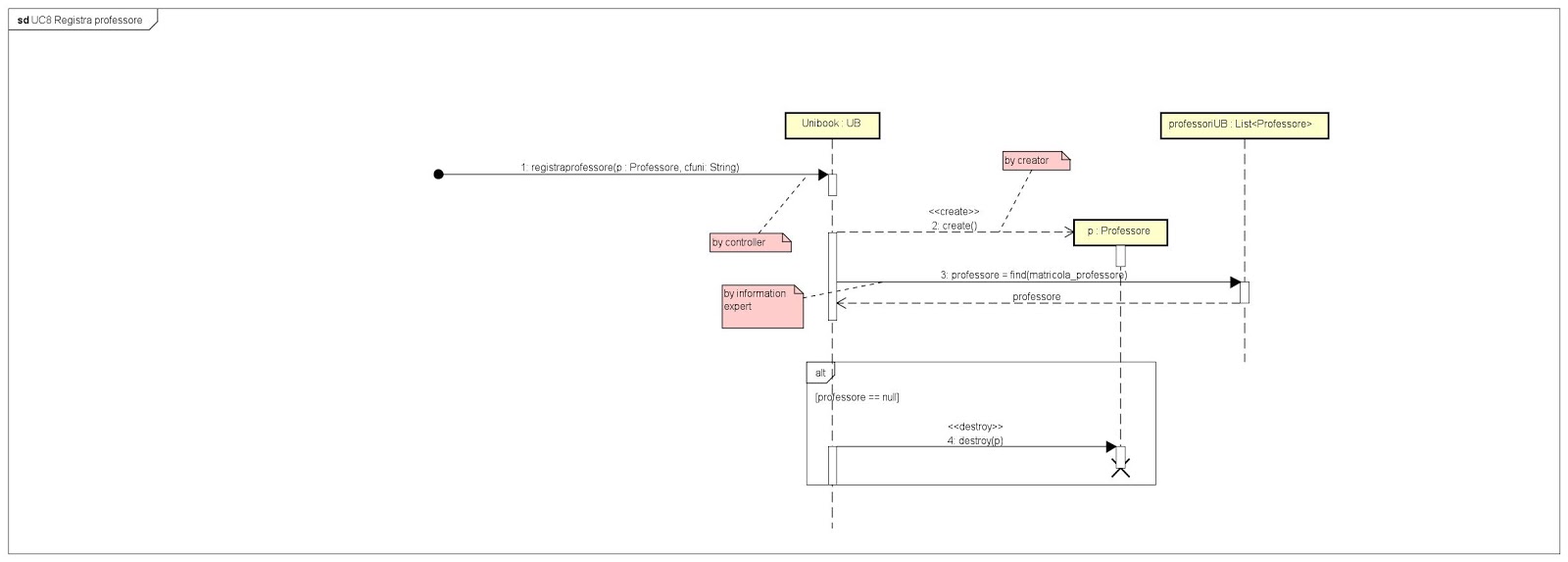


#### VerificaStudente

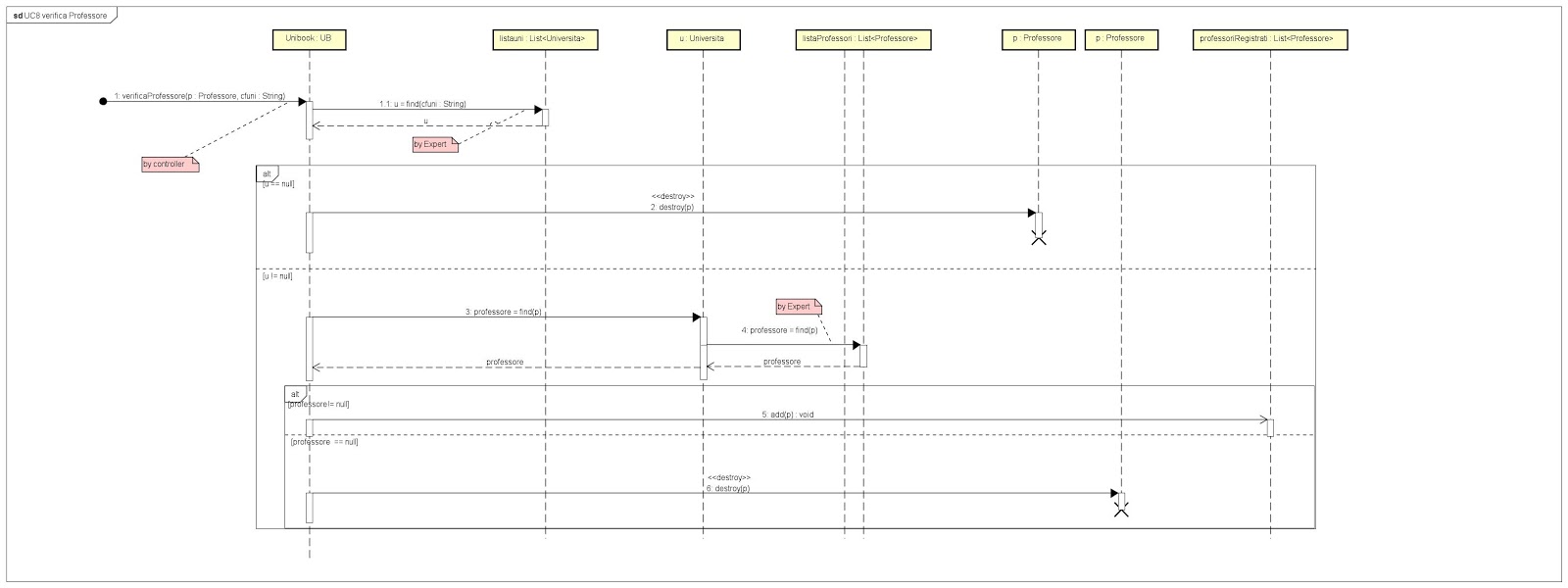


### UC8

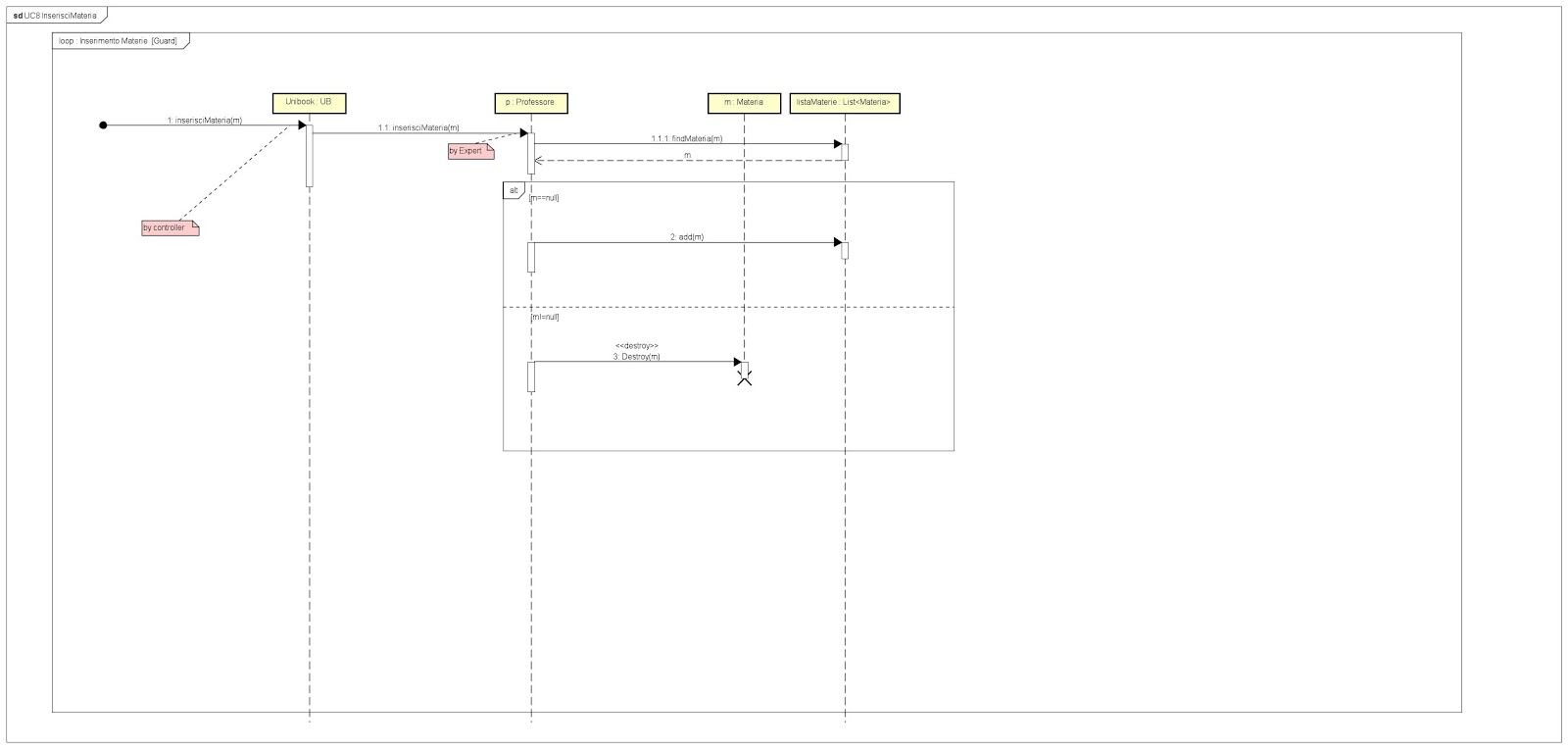
#### RegistraProfessore



#### VerificaProfessore

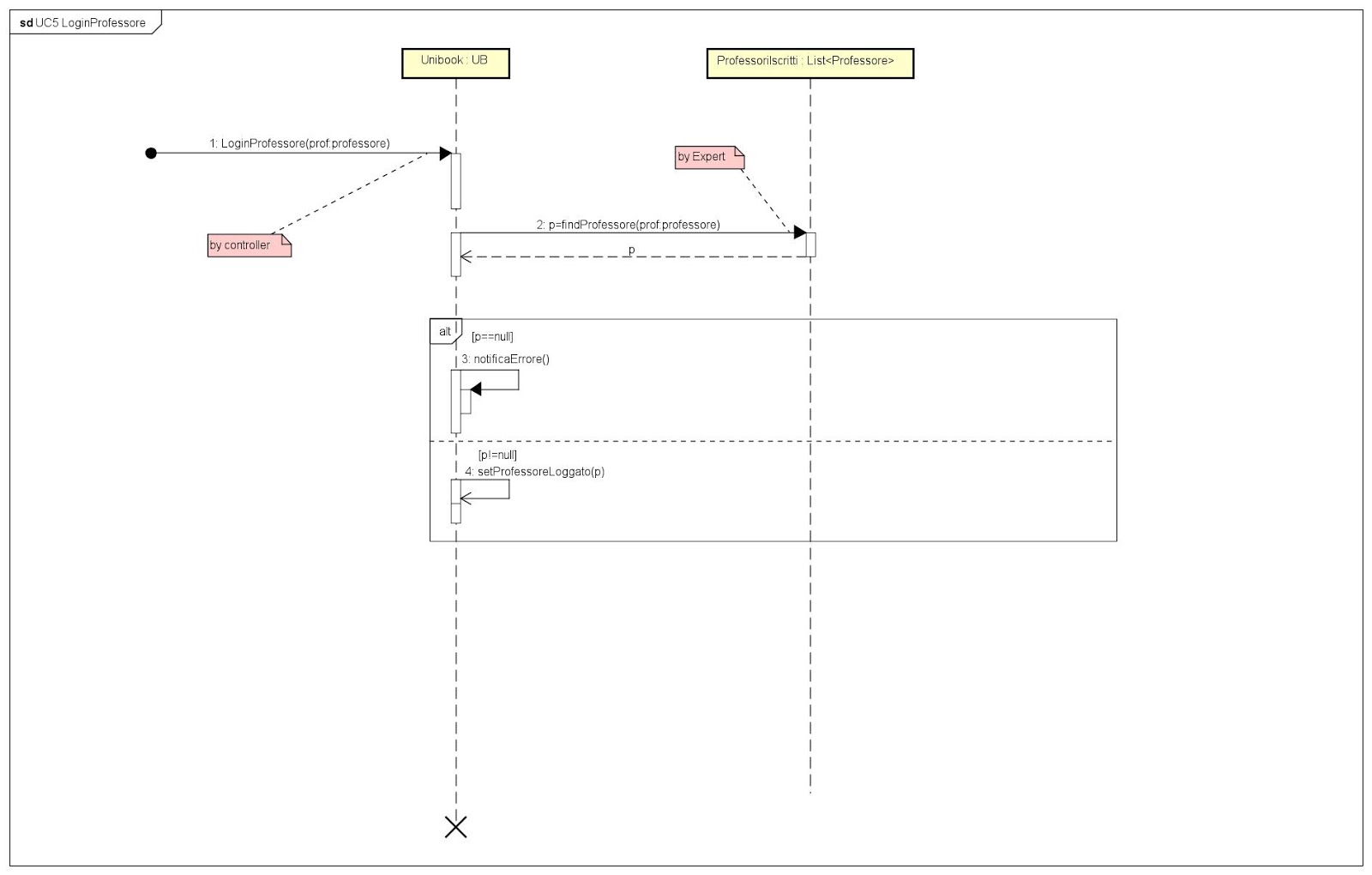


#### InserisciMateria

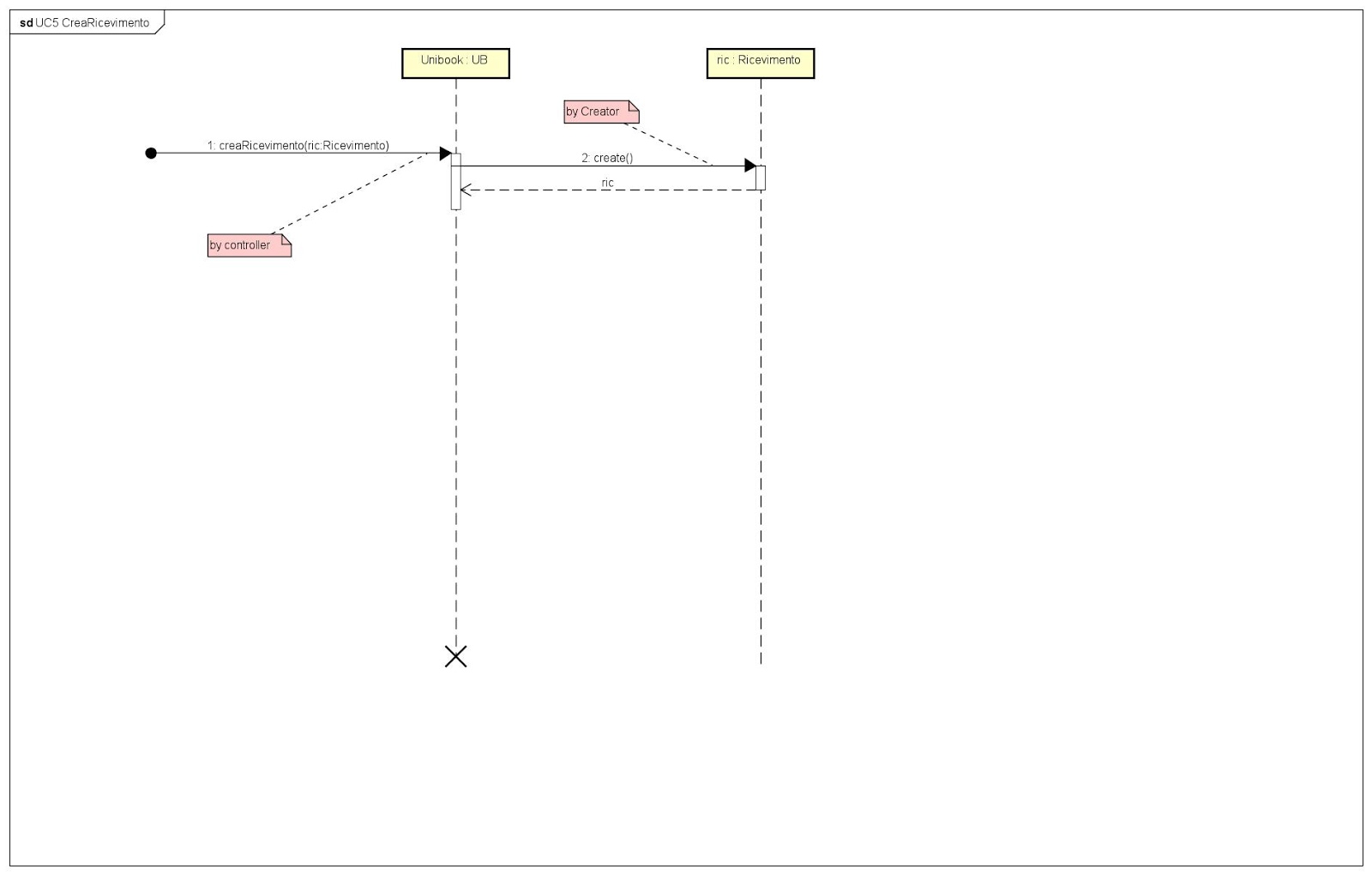


### UC5

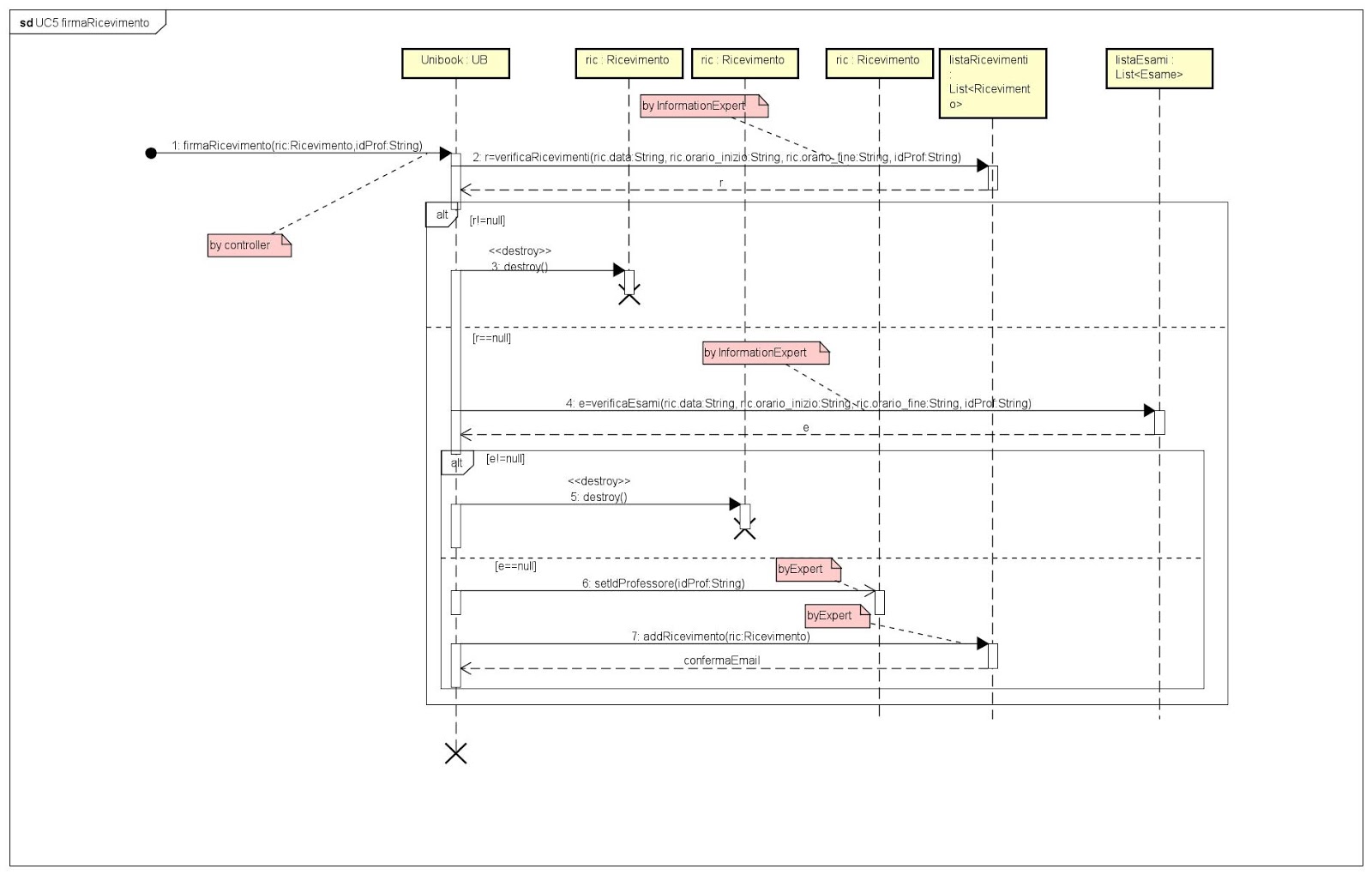
#### LoginProfessore



#### CreaRicevimento

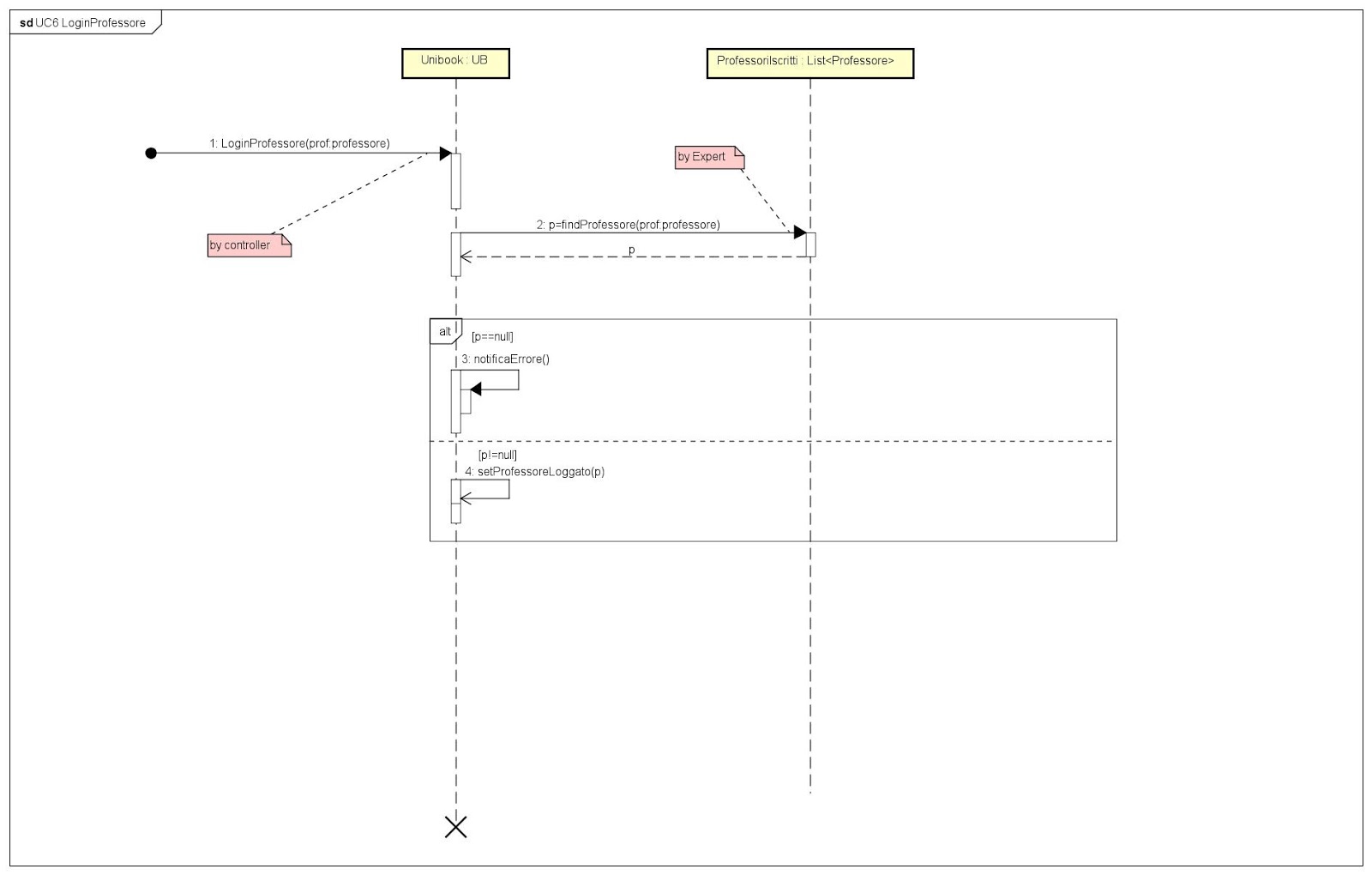


#### FirmaRicevimento

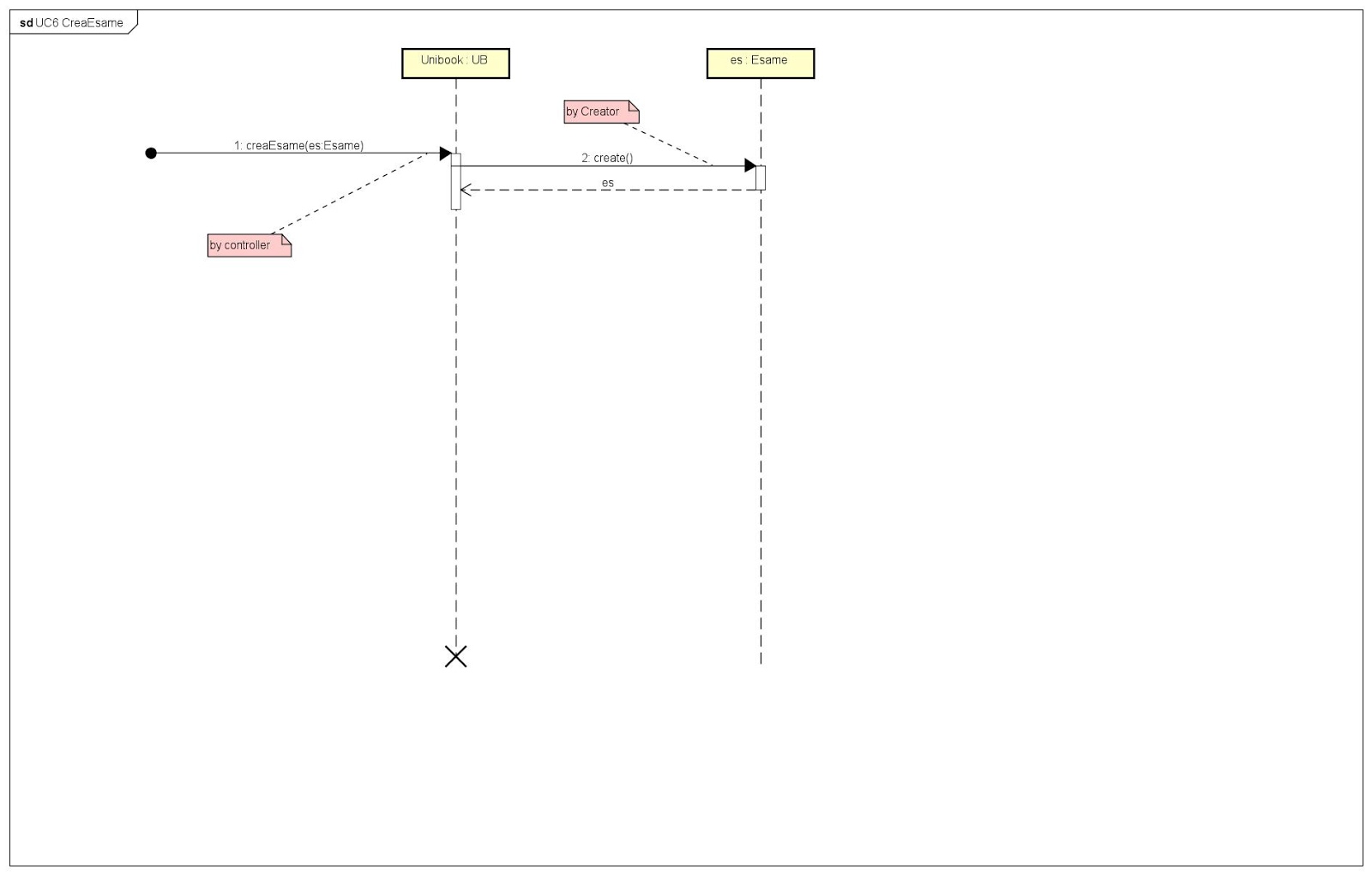


### UC6

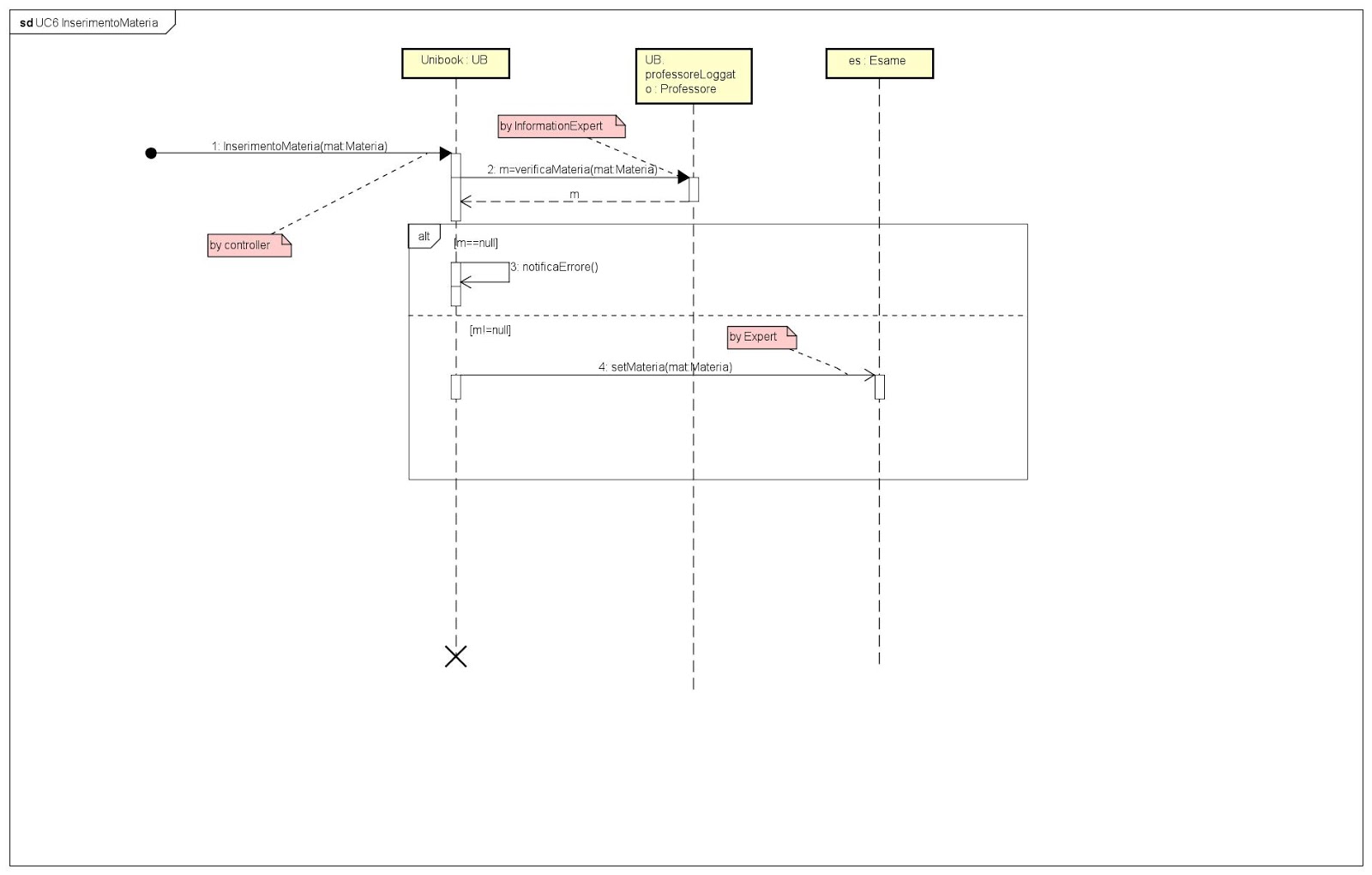
#### LoginProfessore



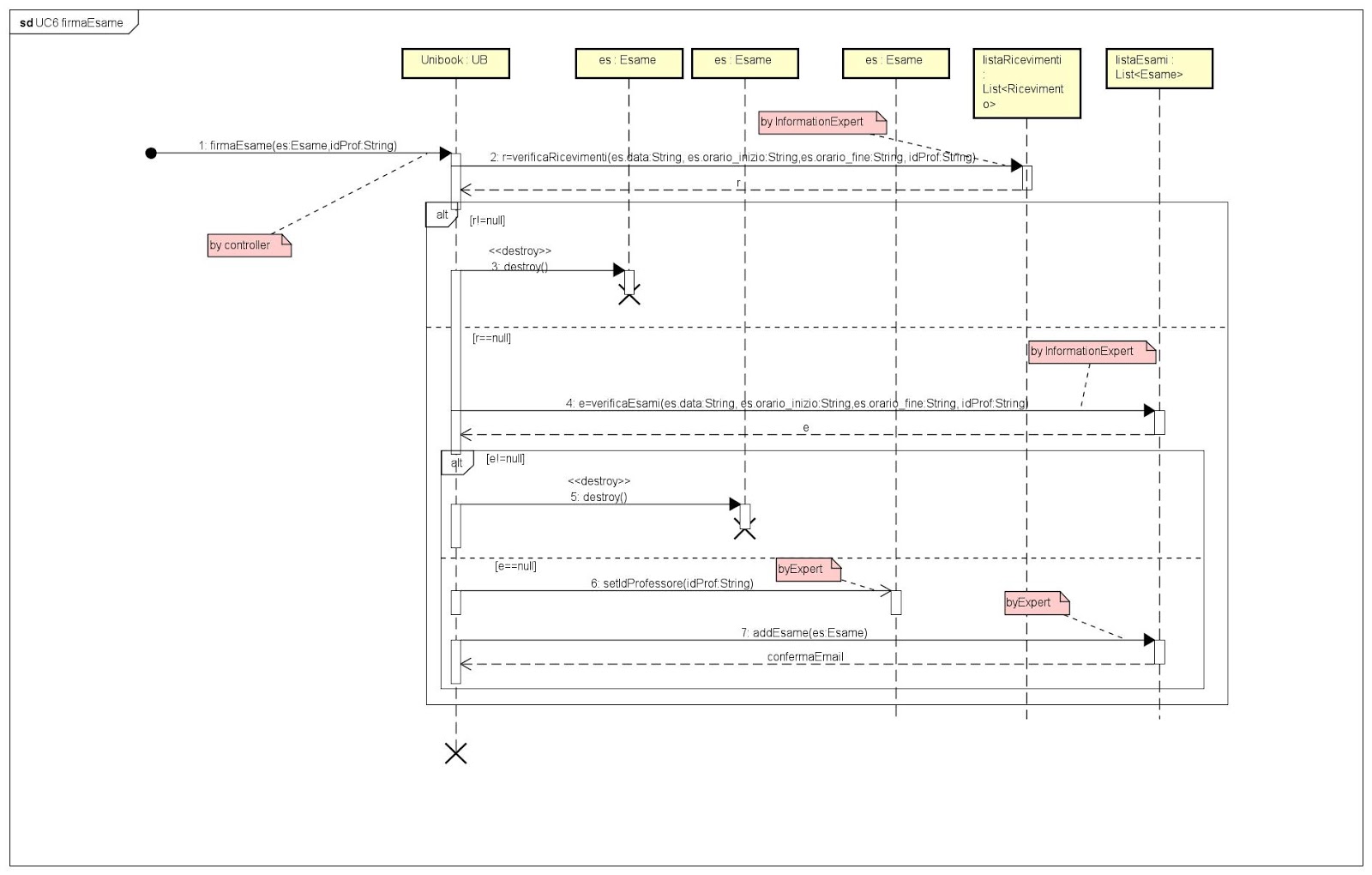
#### CreaEsame



#### InserimentoMateria

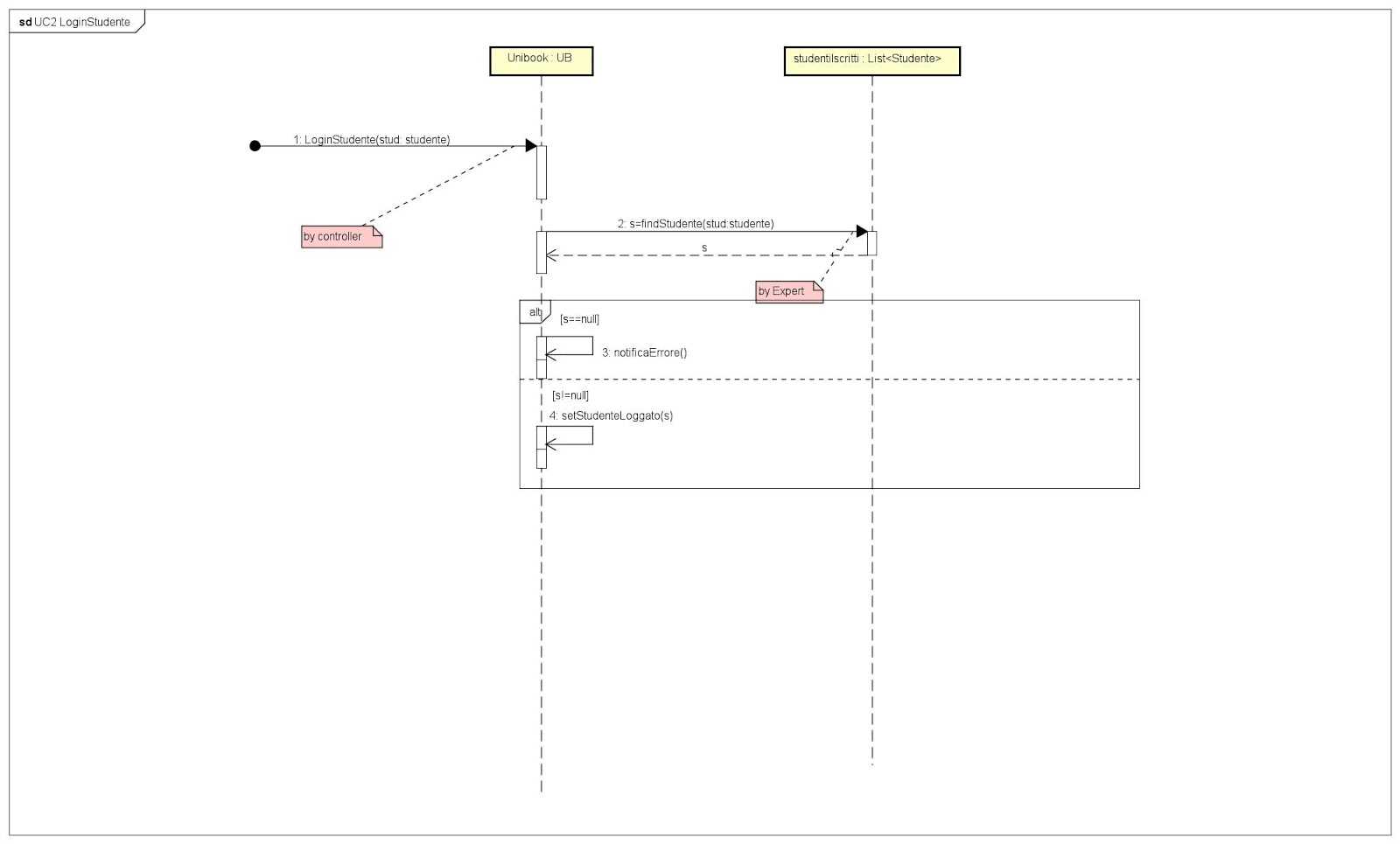


#### FirmaEsame

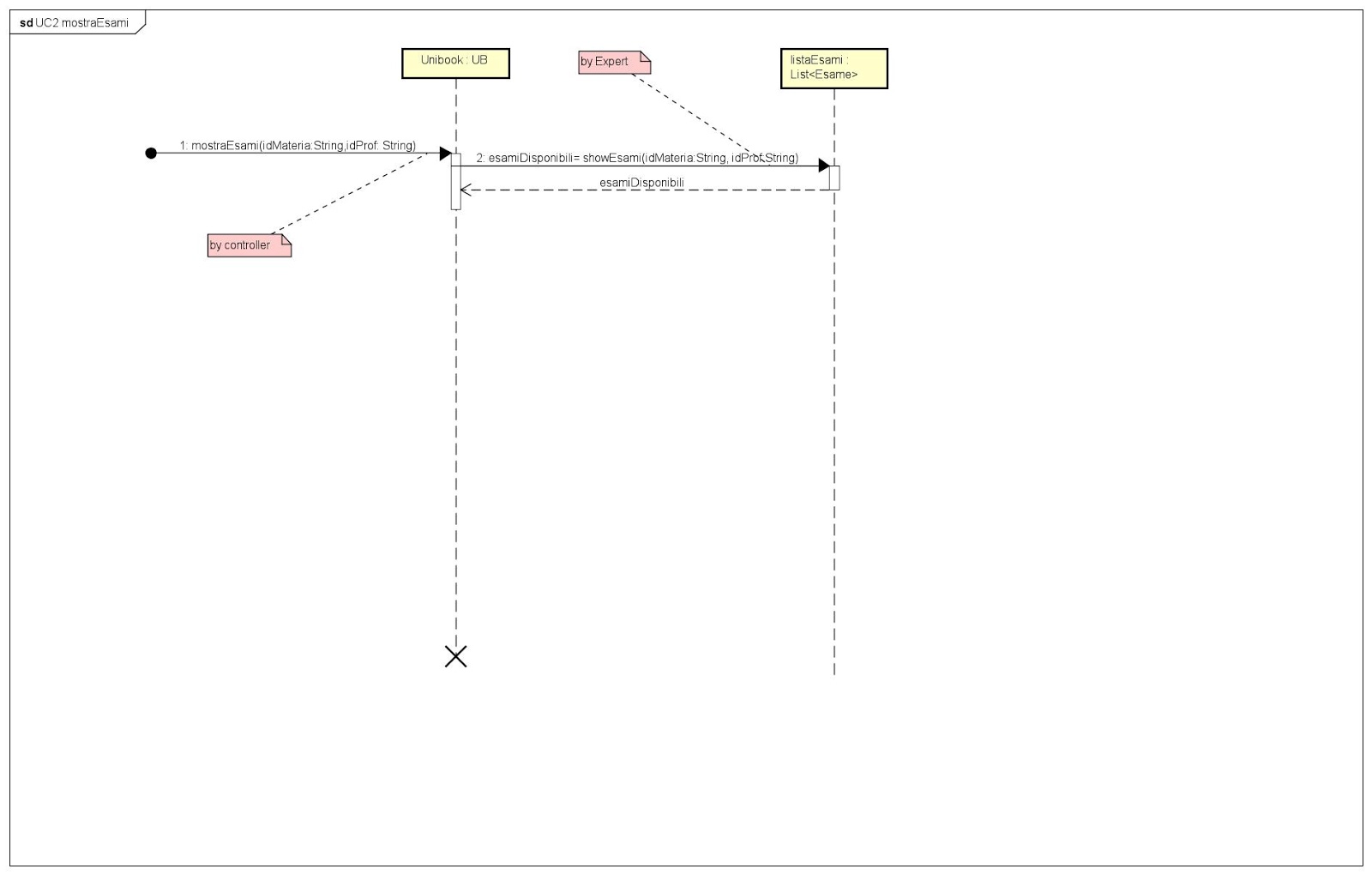


### UC2

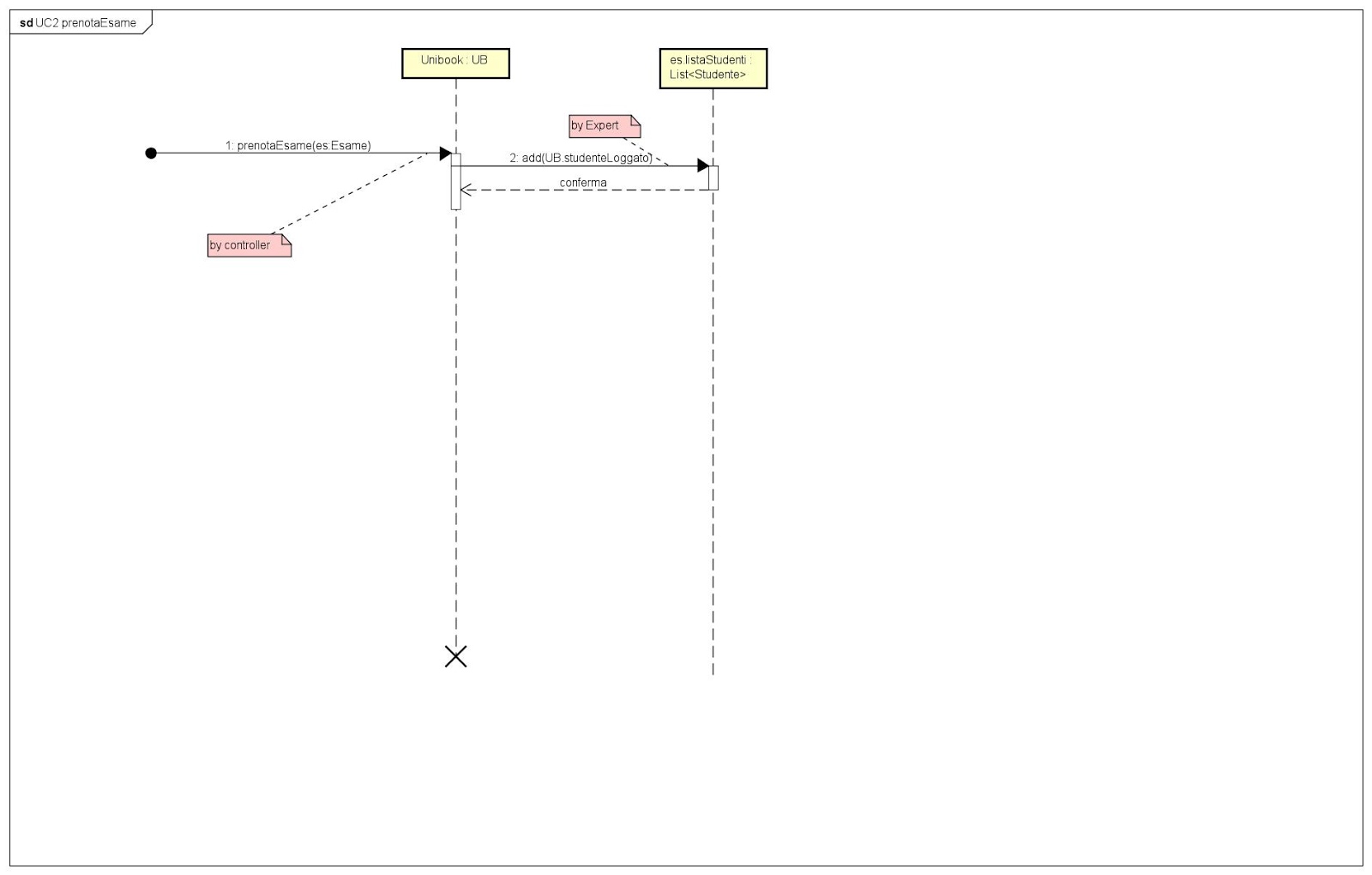
#### LoginStudente



#### MostraEsame

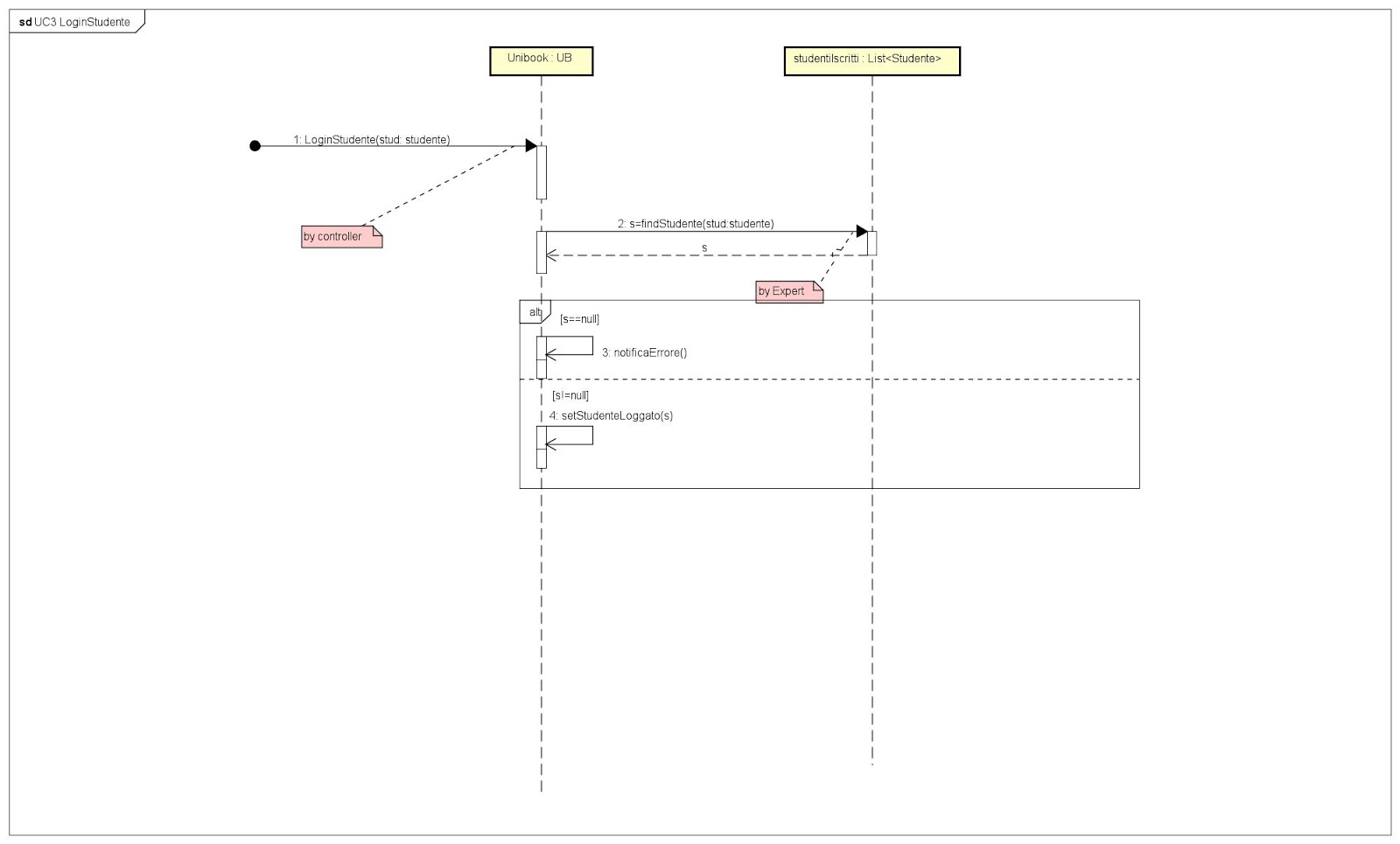


#### PrenotaEsame

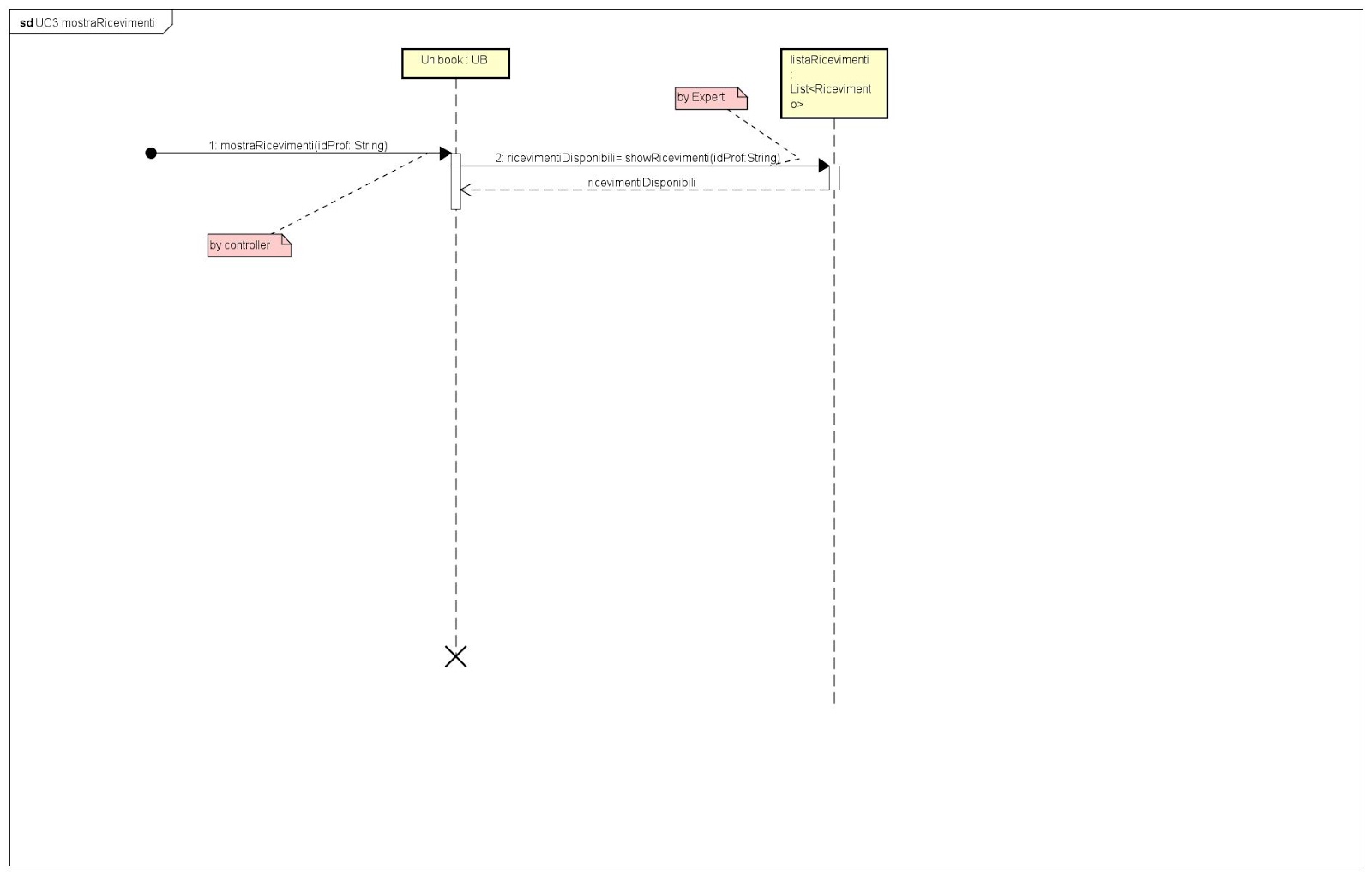


### UC3

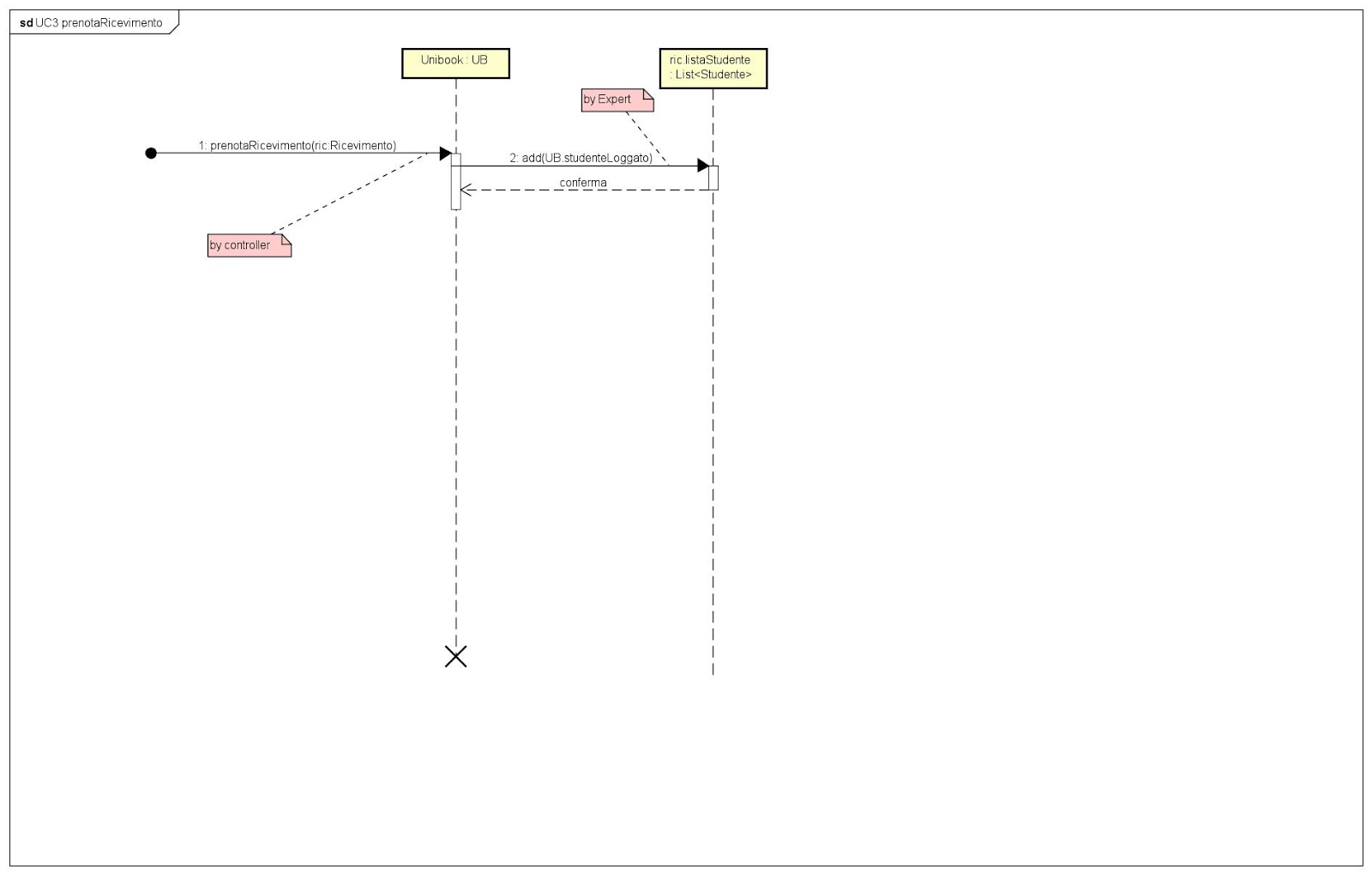
#### LoginStudente



#### MostraRicevimenti

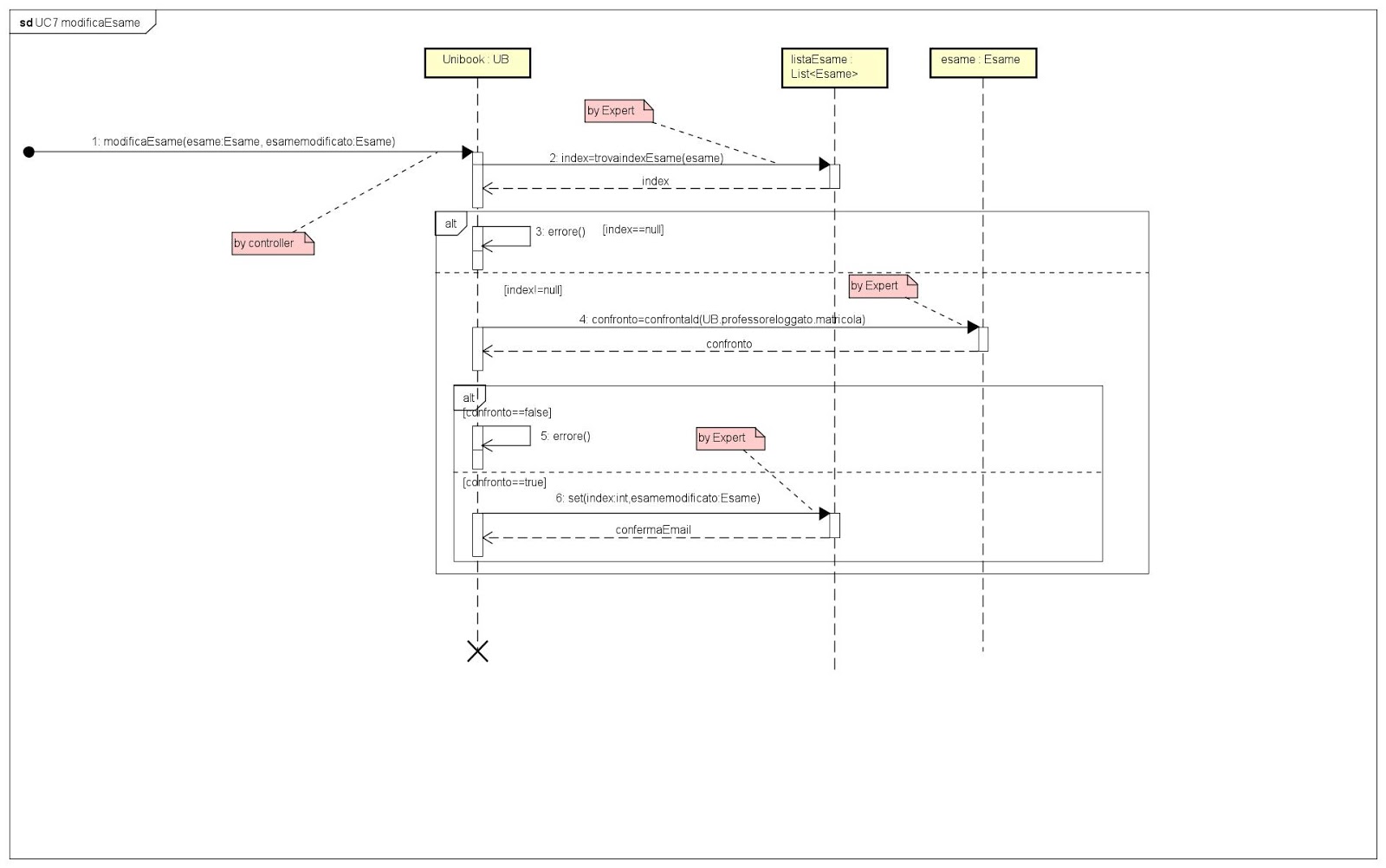


#### PrenotaRicevimento

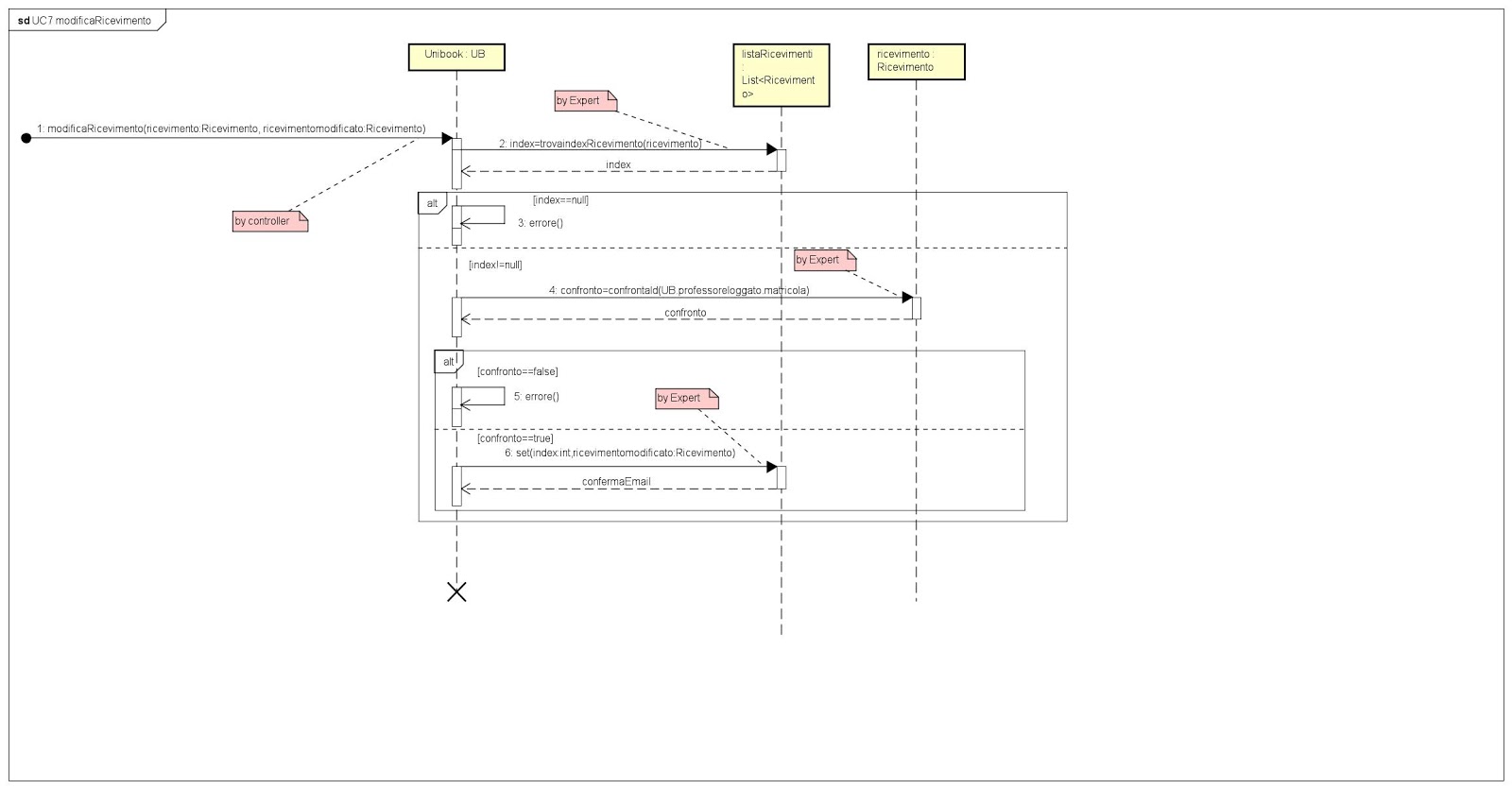


### UC7

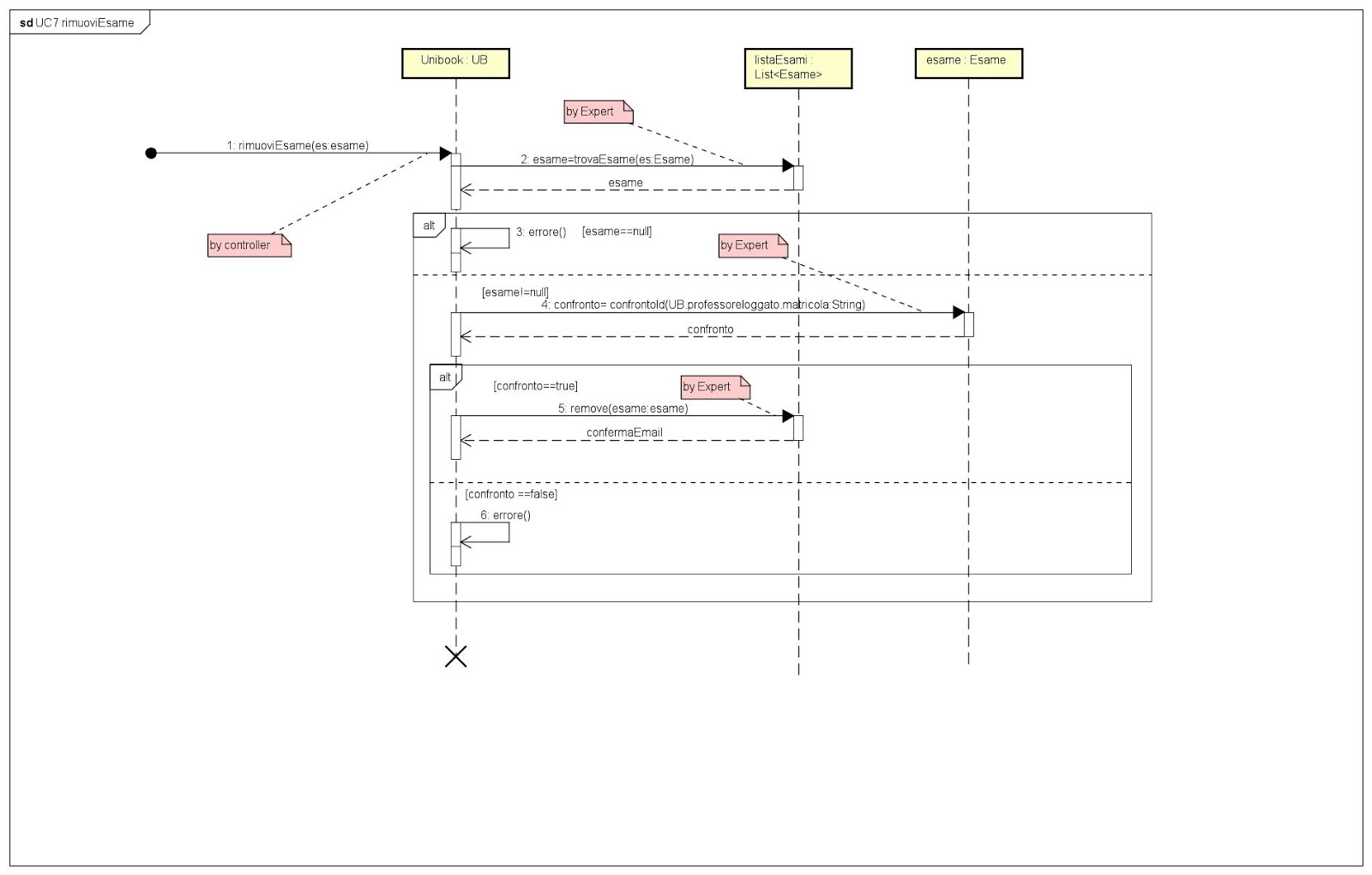
#### ModificaEsame



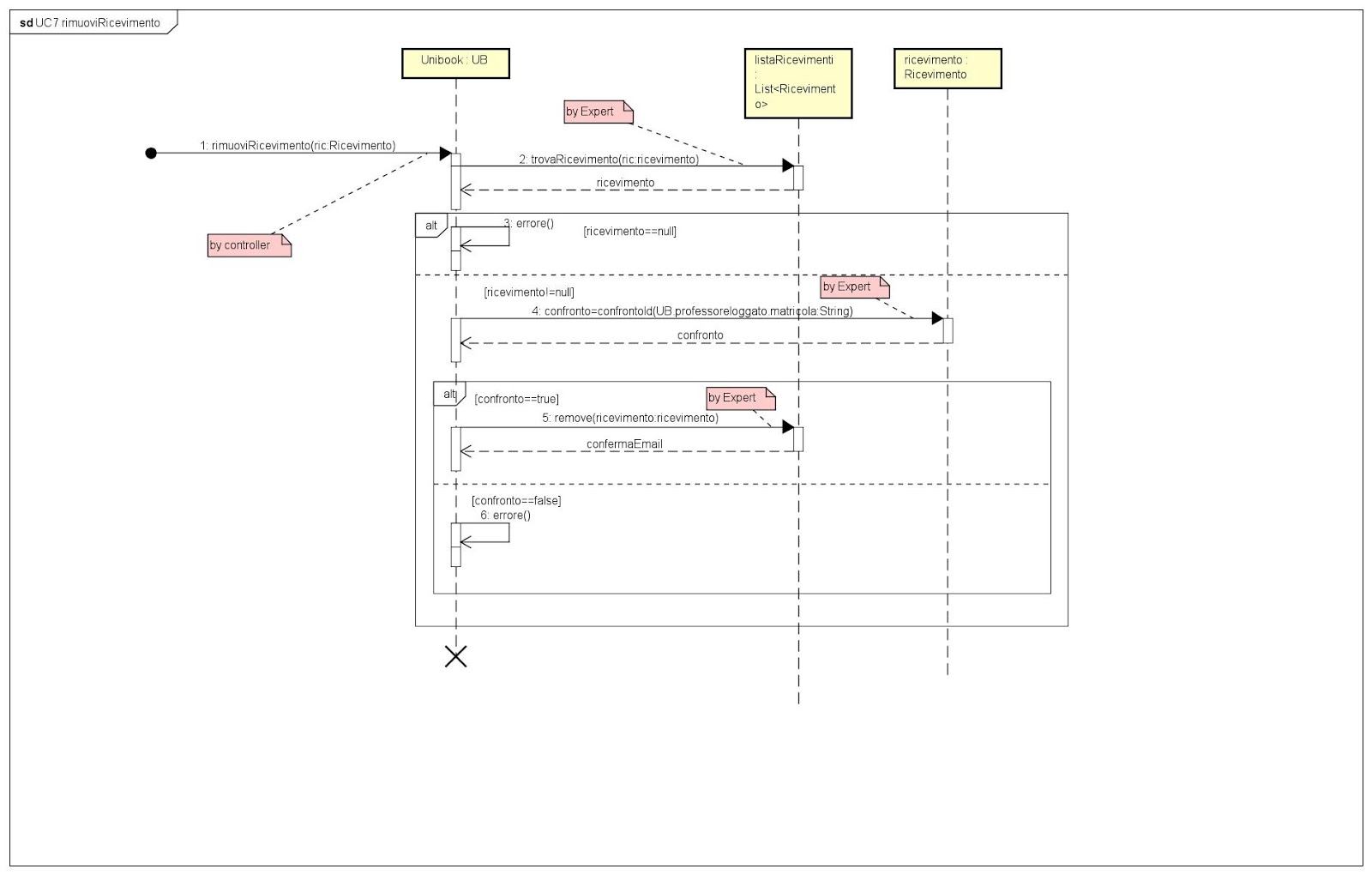
#### ModificaRicevimento



#### RimuoviEsame

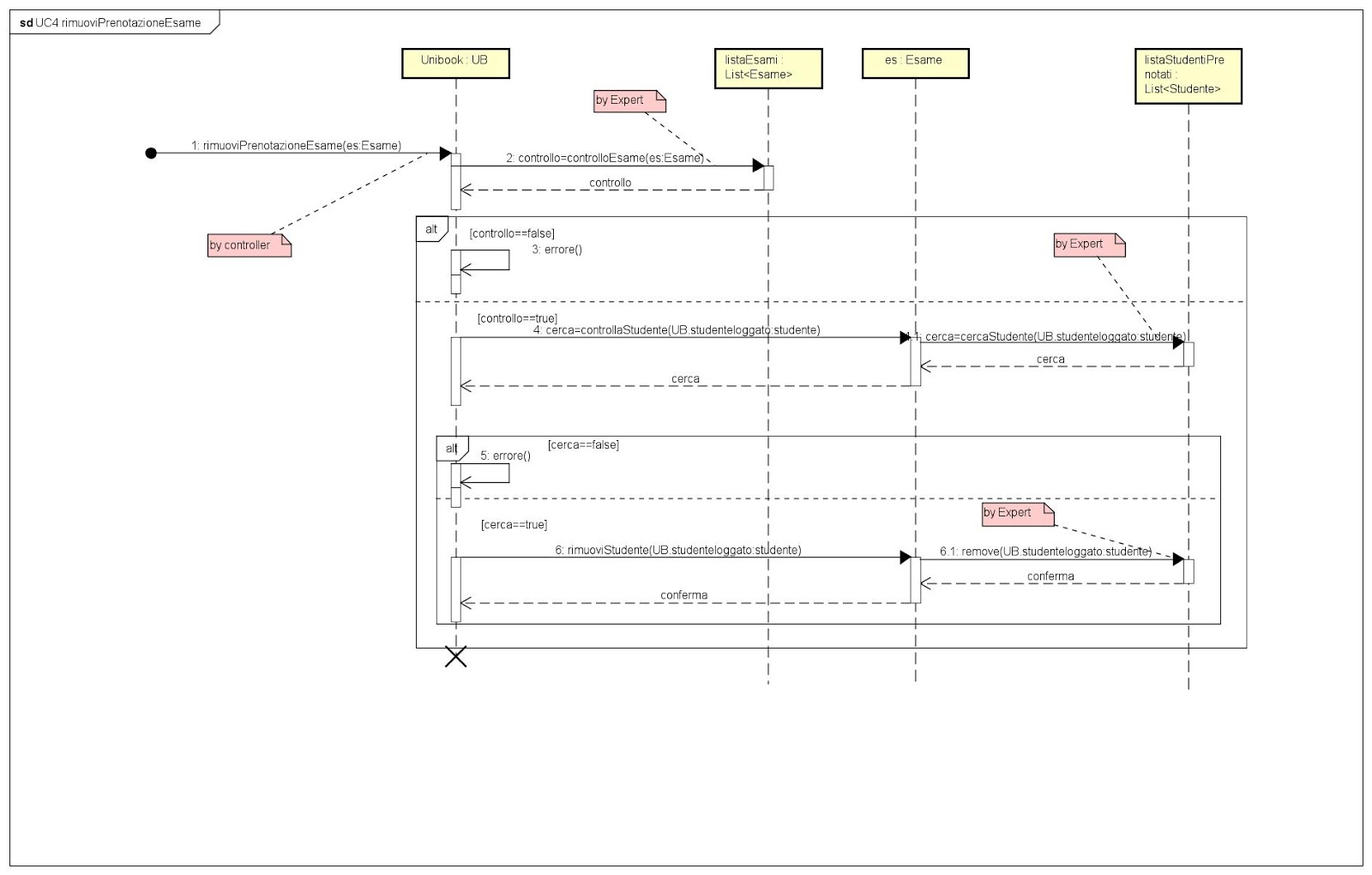


#### RimuoviRicevimento

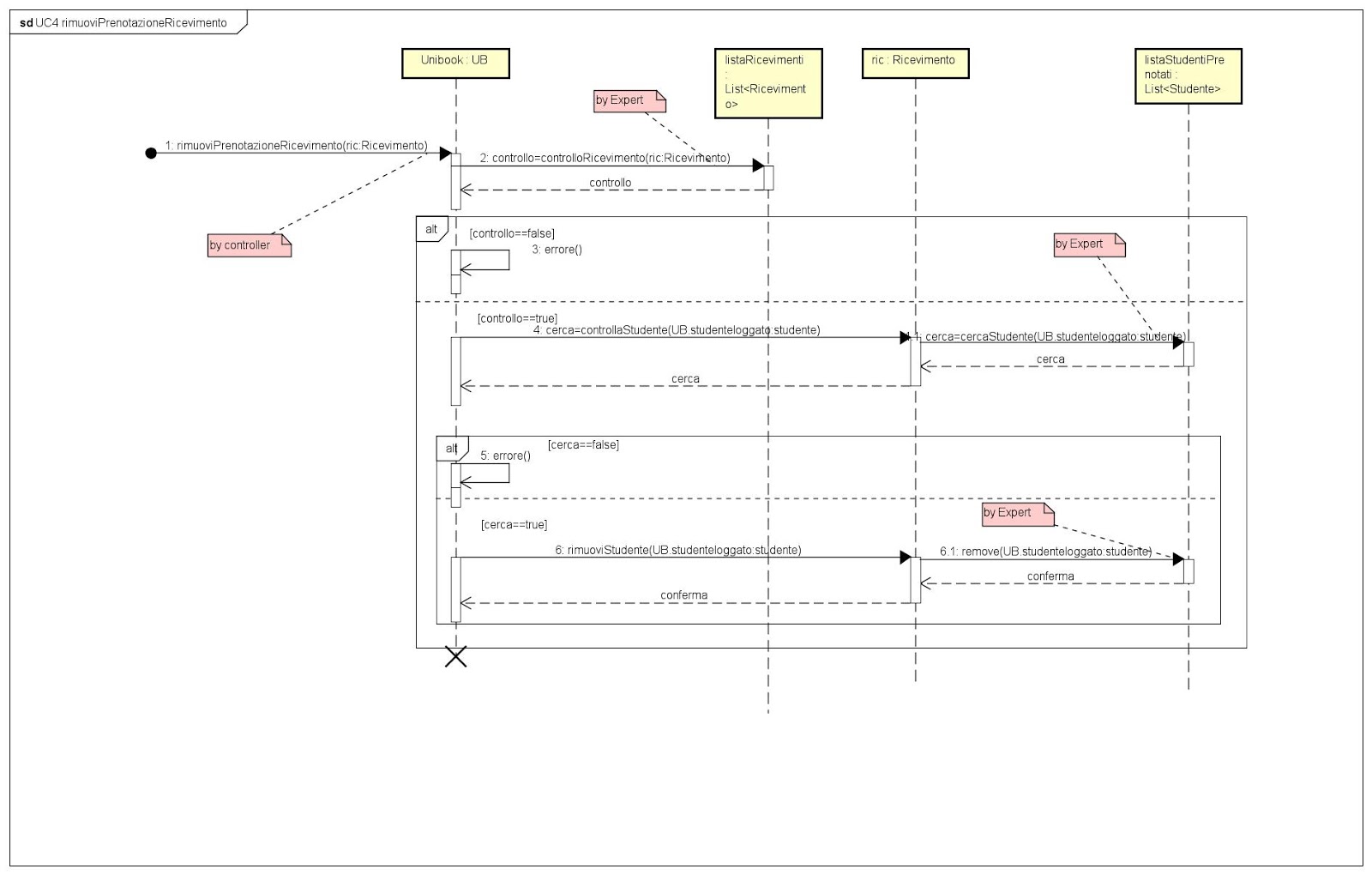


### UC4

#### RimuoviPrenotazioneEsame



#### RimuoviPrenotazioneRicevimento



# TESTING

Con la parola testing si intende una verifica sperimentale, fatta mediante l’esecuzione dell’applicativo, selezionando alcuni dati ingresso e valutando i risultati. Lo scopo è quello di verificare che il sistema sviluppato riesca ad assolvere le specifiche per cui è stato ideato. Inoltre l’obiettivo del test non è come si può pensare erroneamente, quello di mostrare l’assenza di difetti ma invece quello di trovare dei dati che possano causare la scoperta di errori durante l’esecuzione. E’ impossibile realizzare un testing in grado di coprire tutti i possibili casi ingresso, pertanto abbiamo selezionato alcune operazioni tra le più importanti nelle quali si possono riscontrare eventuali errori.

## Test Iterazione 1 e 2

Nella prima iterazione ci occupiamo di creare un’istanza della classe professore e un’istanza della classe Studente. Abbiamo realizzato due classi chiamati: TestRegistraStudente e TestRegistraProfessore. All’interno di entrambe le classi sono presenti diversi metodi che descrivono sia la prima che la seconda iterazione.

I metodi registraStudente e registraProfessore vanno a testare la prima iterazione mentre i metodi validitàCodiceFiscale, validitàCorsoDiStudi e testInserimentoMateria verificano la seconda iterazione.

### TestRegistraStudente

class testRegistraStudente {  
 Unibook unibook;  
 Universita u1;  
 Studente studente;  
 CorsoStudio cs;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void registraStudente() {  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 cs = new CorsoStudio("Ingegneria Informatica");  
 String cfuni = "eivwbtp394ut";  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String password = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
 index = unibook.getListastudenti().indexOf(studente);  
 *assertEquals*(studente, unibook.getListastudenti().get(index));  
 // *TODO review the generated test code and remove the default call to fail.* }  
  
 @Test  
 void validitàCodiceFiscale(){  
 /\*  
 Il verificatore notifica ad UniBook la non validità del codice fiscale  
 dell’Università inserito dall’utente. Questo significa  
 che l'utente che si vuole registrare ad unibook non è iscritto a quell'università  
 \*/  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 cs = new CorsoStudio("Ingegneria Informatica");  
  
 String cfuni = "43445245"; // ---> codice fiscale errato  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String password = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
 index = unibook.getListastudenti().indexOf(studente);  
 *assertEquals*(-1, index); // ---> la lista deve essere vuota  
  
 }  
  
 @Test  
 void validitàCorsoDiStudi(){  
 /\*  
 Il verificatore notifica ad UniBook la non validità del codice fiscale  
 del corso di studi inserito dall’utente. Questo significa  
 che l'utente che si vuole registrare ad unibook non è iscritto a quel corso di studi  
 \*/  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 cs = new CorsoStudio("Ingegneria meccanica");// ---> corso di studi sbagliato  
 String cfuni = "eivwbtp394ut";  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String password = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
 index = unibook.getListastudenti().indexOf(studente);  
 *assertEquals*(-1, index); // ---> la lista deve essere vuota  
  
 }  
}

### TestRegistraProfessore

class testRegistraProfessore {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
 @Test  
 void registraProfessore() {  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
 index = unibook.getListaprofessori().indexOf(professore);  
 *assertEquals*(professore, unibook.getListaprofessori().get(index));  
 }  
  
 @Test  
 void testInserimentoMateria(){  
 /\*  
 Il verificatore notifica ad UniBook un errore con l’inserimento di una materia  
 o più materie inserite dall’utente:  
 \*/  
  
 //---------FASE DI REGISTRAZIONE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
 index = unibook.getListaprofessori().indexOf(professore);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE AGGIUNTA MATERIA-----------  
 CorsoStudio cs = new CorsoStudio("Ingegneria Informatica");  
 Materia materia = new Materia(9, "analisi");  
 cs.addMateria(materia);  
 unibook.inserisciMateria(materia, cs);  
 int index = unibook.getProfessoreloggato().getListamaterie().indexOf(materia);  
  
 *assertEquals*(materia, unibook.getProfessoreloggato().getListamaterie().get(index));  
 }  
}

## Test Iterazione 3 e 4

Nella terza iterazione ci occupiamo di creare un’istanza della classe Esame e un’istanza della classe Ricevimento, mentre nella quarta iterazione verifichiamo gli scenari alternativi, in particolare la presenza di un ricevimento e di un esame (e viceversa) nello stesso giorno.

Affinchè vengano verificate le iterazioni 3 e 4, abbiamo realizzato due classi chiamati: TestCreaEsame e TestCreaRicevimento.

### TestCreaEsame

class testCreaEsame {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void creaEsame() {  
  
 //---------FASE DI REGISTRAZIONE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.loginProfessore(professore);  
 unibook.creaEsame(esame);  
  
 //---------FASE CREAZIONE ESAME-----------  
 int index = unibook.getListaesami().indexOf(esame);  
 *assertEquals*(esame, unibook.getListaesami().get(index));  
 *assertEquals*(professore.getMatricola(), unibook.getListaesami().get(index).getIdprofessore());  
 }  
  
 @Test  
 void validitàDataEsame(){  
  
 //---------FASE DI REGISTRAZIONE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN PROFESSORE-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE DI CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.creaRicevimento(ricevimento);  
  
 //---------FASE CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 int index = unibook.getListaesami().indexOf(esame);  
  
 *assertEquals*(-1, unibook.getListaesami().indexOf(esame));  
  
 }  
}

### TestCreaRicevimento

class testCreaRicevimento {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void creaRicevimento() {  
  
 //---------FASE DI REGISTRAZIONE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.loginProfessore(professore);  
 unibook.creaRicevimento(ricevimento);  
  
 int index = unibook.getListaricevimenti().indexOf(ricevimento);  
 *assertEquals*(ricevimento, unibook.getListaricevimenti().get(index));  
 *assertEquals*(professore.getMatricola(), unibook.getListaricevimenti().get(index).getIdprofessore());  
  
 }  
  
 @Test  
 void validitàDataRicevimento(){  
 //---------FASE DI REGISTRAZIONE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN PROFESSORE-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE DI CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 //---------FASE DI CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.creaRicevimento(ricevimento);  
  
 int index = unibook.getListaricevimenti().indexOf(ricevimento);  
  
 *assertEquals*(-1, unibook.getListaricevimenti().indexOf(ricevimento));  
 }  
}

## Test Iterazione 5

Nella quinta  iterazione ci occupiamo di testare la prenotazione di un’esame e di un ricevimento da parte di uno studente. Abbiamo realizzato due classi chiamati: TestPrenotazioneEsame e TestPrenotazioneRicevimento. 

### TestPrenotaEsame

class testPrenotaEsame {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void prenotazioneEsame() {  
  
 //---------FASE DI REGISTRAZIONE STUDENTE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Studente studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 CorsoStudio cs = new CorsoStudio("Ingegneria Informatica");  
 String cfuni = "eivwbtp394ut";  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String password = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 u1.addProfessore(professore);  
  
 String passwordProf = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, passwordProf);  
 index = unibook.getListaprofessori().indexOf(professore);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginStudente(studente);  
 unibook.loginProfessore(professore);  
  
 //---------FASE DI CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 //---------FASE DI PRENOTAZIONE ESAME-----------  
  
 unibook.prenotazioneEsame(esame.getId());  
 int index = esame.getListaStudentiprenotati().indexOf(studente);  
 *assertEquals*(studente, esame.getListaStudentiprenotati().get(index));  
 }  
}

### TestPrenotaRicevimento

class testPrenotaRicevimento {  
  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void prenotazioneRicevimento() {  
  
 //---------FASE DI REGISTRAZIONE STUDENTE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Studente studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 CorsoStudio cs = new CorsoStudio("Ingegneria Informatica");  
 String cfuni = "eivwbtp394ut";  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String password = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 u1.addProfessore(professore);  
  
 String passwordProf = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, passwordProf);  
 index = unibook.getListaprofessori().indexOf(professore);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginStudente(studente);  
 unibook.loginProfessore(professore);  
  
 //---------FASE CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.loginProfessore(professore);  
 unibook.creaRicevimento(ricevimento);  
  
 //---------FASE PRENOTAZIONE-----------  
 unibook.prenotazioneRicevimento(ricevimento.getId());  
  
  
 int index = ricevimento.getlistaStudentiprenotati().indexOf(studente);  
 *assertEquals*(studente, ricevimento.getlistaStudentiprenotati().get(index));  
  
 }  
}

## Test Iterazione 6

Nella quinta  iterazione ci occupiamo di testare la prenotazione di un’esame e di un ricevimento da parte di uno studente. Abbiamo realizzato quattro classi chiamati: TestGestioneEsameProfessore, TestGestioneEsameStudente, TestGestioneRicevimentoProfessore, TestGestionericevimentoStudente. In tutte e 4 le classi sono presenti rispettivamente 2 metodi di rimozione e modifica di un esame o di un ricevimento.

### TestGestioneEsameProfessore

public class testGestioneEsameProfessore {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void rimuoviEsame() {  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 //---------FASE RIMOZIONE RICEVIMENTO-----------  
 unibook.rimuoviEsame(esame);  
  
 int index = unibook.getListaricevimenti().indexOf(esame);  
  
 *assertEquals*(-1, index);  
  
 }  
  
 @Test  
 void modificaEsame() {  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 //---------FASE MODIFICA ESAME-----------  
 //proviamo a modificare la data dell' esame  
 Esame esamemodificato = new Esame("aula 5", 30, "20/06/2020", "17:30", "14:30", "Orale");  
 unibook.modificaEsame(esame, esamemodificato);  
  
 int index = unibook.getListaesami().indexOf(esamemodificato);  
 *assertEquals*("20/06/2020", unibook.getListaesami().get(index).getData());  
 }  
}

### TestGestioneEsameStudente

public class testGestioneEsameStudente {  
  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void rimuoviEsame() {  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI REGISTRAZIONE STUDENTE-----------  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Studente studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 CorsoStudio cs = new CorsoStudio("Ingegneria Informatica");  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String passwordStudente = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
  
 //---------FASE DI LOGIN-----------  
  
 unibook.loginProfessore(professore);  
 unibook.loginStudente(studente);  
  
 //---------FASE CREAZIONE ESAME-----------  
 Esame esame = new Esame("aula 5", 30, "15/06/2020", "17:30", "14:30", "Orale");  
 unibook.creaEsame(esame);  
  
 //---------FASE PRENOTAZIONE ESAME-----------  
 unibook.prenotazioneEsame(esame.getId());  
  
 //---------FASE RIMOZIONE-----------  
 unibook.rimuoviPrenotazioneEsame(esame.getId());  
  
 int index = esame.getListaStudentiprenotati().indexOf(studente);  
  
 *assertEquals*(-1, index);  
  
 }  
  
}

### TestGestioneRicevimentoProfessore

class testGestioneRicevimentoProfessore {  
  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void rimuoviRicevimento() {  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.creaRicevimento(ricevimento);  
  
 //---------FASE RIMOZIONE RICEVIMENTO-----------  
 unibook.rimuoviRicevimento(ricevimento);  
  
 int index = unibook.getListaricevimenti().indexOf(ricevimento);  
  
 *assertEquals*(-1, index);  
  
 }  
  
 @Test  
 void modificaRicevimento() {  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI LOGIN-----------  
 unibook.loginProfessore(professore);  
  
 //---------FASE CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.creaRicevimento(ricevimento);  
  
 //---------FASE MODIFICA RICEVIMENTO-----------  
 //proviamo a modificare la data dell' esame  
 Ricevimento ricevimentomodificato = new Ricevimento("aula 5", 30, "15/06/2020", "18:30", "14:30");  
 unibook.modificaRicevimento(ricevimento, ricevimentomodificato);  
  
 int index = unibook.getListaricevimenti().indexOf(ricevimentomodificato);  
 *assertEquals*("18:30", unibook.getListaricevimenti().get(index).getOrariofine());  
 }  
}

### TestGestioneEsameStudente

package com.example.demo;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.*assertEquals*;  
  
public class testGestioneRicevimentoStudente {  
  
 Unibook unibook;  
 Universita u1;  
 int index;  
  
 @BeforeEach  
 void setUp() {  
 unibook = Unibook.*getInstance*();  
 }  
  
 @Test  
 void rimuoviRicevimento() {  
  
 //---------FASE DI REGISTRAZIONE PROFESSORE-----------  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Professore professore = new Professore("Pietro", "Zamboni", "T0987834", "pietro.zamboni@gmail.com");  
 String cfuni = "eivwbtp394ut";  
 u1.addProfessore(professore);  
  
 String password = "pietro";  
  
 unibook.addUniversità(u1);  
 unibook.registraProfessore(professore, cfuni, password);  
  
 //---------FASE DI REGISTRAZIONE STUDENTE-----------  
  
 u1 = new Universita("eivwbtp394ut", "unict", "via santa sofia");  
 Studente studente = new Studente("Flavio", "Condemi", "O46001668", "flavio.condemi@gmail.com");  
 CorsoStudio cs = new CorsoStudio("Ingegneria Informatica");  
  
 cs.addStudente(studente);  
 u1.addCorsostudio(cs);  
 String passwordStudente = "flavio";  
 unibook.addUniversità(u1);  
 unibook.registraStudente(studente, cfuni, cs, password);  
  
 //---------FASE DI LOGIN-----------  
  
 unibook.loginProfessore(professore);  
 unibook.loginStudente(studente);  
  
 //---------FASE CREAZIONE RICEVIMENTO-----------  
 Ricevimento ricevimento = new Ricevimento("aula 5", 30, "15/06/2020", "17:30", "14:30");  
 unibook.creaRicevimento(ricevimento);  
  
 //---------FASE PRENOTAZIONE ESAME-----------  
 unibook.prenotazioneRicevimento(ricevimento.getId());  
  
 //---------FASE RIMOZIONE-----------  
 unibook.rimuoviPrenotazioneRicevimento(ricevimento.getId());  
  
 int index = ricevimento.getlistaStudentiprenotati().indexOf(studente);  
  
 *assertEquals*(-1, index);  
 }  
  
}