# Progetto di laboratorio

Corso di Progettazione Model Driven del Software

https://github.com/gabrielestentella/project

Gabriele Stentella

# 1 Descrizione del problema

## 1.1 Analisi e specifica dei requisiti

Viene richiesto di realizzare un applicativo software che aiuti la gestione di attivazione e disattivazione dei tirocini associati alle tesi per i corsi di laurea triennali del Dipartimento di Informatica. Il sistema deve rispettare i seguenti requisiti:

- Requisiti funzionali

  1. Lo studente deve poter compilare una richiesta di attivazione del tirocinio che può essere interno o esterno
  2. Lo studente deve poter indicare il titolo del tirocinio
  3. Lo studente deve poter indicare i dati anagrafici relativi a se stesso, al relatore, e a eventuali correlatori
  4. Lo studente deve poter indicare la data di inizio e la data di fine del tirocinio
  5. Lo studente deve poter indicare i dati dell'azienda presso cui intende svolgere il tirocinio, in caso di tirocinio esterno
  6. Lo studente deve poter caricare un file pdf contenente l'autocertificazione degli esami sostenuti
  7. Lo studente deve avere la possibilità, direttamente dal modulo della richiesta di attivazione del tirocinio, di navigare verso il *servizio autocertificazioni* per generare il documento richiesto dal sistema
  8. Lo studente deve ricevere una notifica via mail quando la sua richiesta di tirocinio è stata approvata sia dal relatore sia dalla commissione tirocini
  9. Lo studente deve poter controllare lo *stato del tirocinio*, se il tirocinio è in corso deve essere visualizzato il tempo trascorso dalla data di approvazione
  10. Il relatore deve ricevere una notifica via mail quando uno studente compila una richiesta di tirocinio in cui compare come relatore
  11. Il relatore deve poter visualizzare i dati relativi a tutti i tirocini in cui è coinvolto e lo stato di ciascun tirocinio
  12. Il relatore deve poter approvare o rifiutare una richiesta di tirocinio
  13. Il relatore deve poter modificare i dati relativi a un tirocinio già approvato
  14. Il relatore deve poter terminare un tirocinio

15. La commissione tirocini deve poter visualizzare i dati relativi a tutti i tirocini approvati dai relatori
16. La commissione tirocini deve poter approvare o rifiutare una richiesta di tirocinio

- Requisiti non funzionali

1. Lo studente deve poter attivare un tirocinio soltanto se ha già superato tutti gli *esami obbligatori* dei primi due anni, oppure ha acquisito almeno 120 *CFU* e deve ancora superare al più un esame fra quelli obbligatori nei primi due anni
2. Lo studente deve indicare obbligatoriamente uno e un solo relatore, e da zero a due correlatori
3. Il relatore deve poter terminare un tirocinio solo se si è raggiunta o superata la data di fine del tirocinio
4. Il tirocinio deve durare almeno 14 settimane e non più di 24
5. Il relatore è obbligatoriamente un docente dell'ateneo
6. Se il tirocinio è esterno, deve essere indicato un tutor esterno oltre al relatore accademico
7. Un docente che fa parte della commissione tirocini non può approvare un tirocinio di cui è relatore accademico
8. Il correlatore non è da considerarsi un utente del sistema in quanto non può svolgere nessuna operazione sulla richiesta di tirocinio
9. Se il relatore o la commissione tirocini rifiutano la proposta di tirocinio, lo studente deve ripetere da capo la procedura, i dati della proposta rifiutata non vengono salvati
10. Requisiti di sviluppo:
    (a) Il sistema deve essere sviluppato in linguaggio Java
    (b) L'interfaccia grafica deve essere sviluppata con JavaFX
    (c) I dati persistenti devono essere gestiti con un database relazionale, l'accesso al database deve essere effettuato con la libreria JDBC

## 1.2 Glossario

- CFU: Crediti Formativi Universitari, unità di misura dell'impegno richiesto in termini di attività di studio o di apprendimento, un credito corrisponde convenzionalmente a 25 ore di impegno[1].

- esami obbligatori: gli insegnamenti elencati nelle tabelle delle "Attività formative obbligatorie" contenute nel manifesto degli studi reperibile a `https://apps.unimi.it/files/manifesti/ita_manifesto_F68of4_2024.pdf`.

- servizio autocertificazioni: servizio web messo a disposizione dall'ateneo, accessibile dall'url `https://studente.unimi.it/autocertificati/checkLogin.jsp?0`.

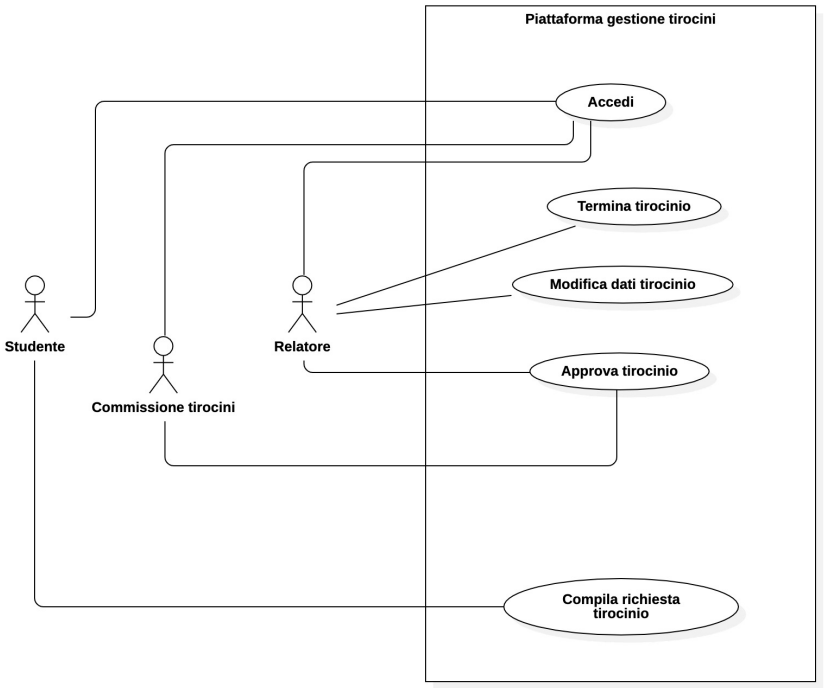
[1] fonte: https://www.unimi.it/it/corsi/orientarsi-e-scegliere/il-sistema-universitario

- stato del tirocinio[2]: un tirocinio è in *attesa di approvazione del relatore* (`attesa1`) dopo che la domanda è stata inviata dallo studente e prima dell'approvazione da parte del relatore; in *attesa di approvazione della commissione* (`attesa2`) dopo che il relatore lo ha approvato e prima che lo approvi la commissione tirocini; *approvato* dopo l'approvazione da parte della commissione e prima della data di inizio tirocinio; *in corso* (`in_corso`) dalla data di inizio tirocinio al momento in cui il relatore termina il tirocinio; *terminato* dopo che il relatore ha terminato il tirocinio.

# 2 Progettazione del Sistema

## 2.1 Diagramma dei casi d'uso

La prima versione del diagramma dei casi d'uso prevedeva il caso d'uso "Accedi" collegato a tutti gli utenti del sistema, tuttavia, secondo la notazione UML, se più di un attore è collegato a un caso d'uso vuol dire che per portarlo a completamento serve necessariamente la partecipazione di tutti gli attori, quindi una possibile correzione sarebbe stata quella di creare un attore astratto che fosse una generalizzazione di tutti gli utenti del sistema, ma a fronte delle sostanziali differenze presenti fra i diversi utenti e del fatto che il caso d'uso in questione ha un funzionamento elementare, si è scelto di indicare l'autenticazione dell'utente come precondizione a tutti gli altri casi d'uso ed eliminare il caso d'uso. Nella seconda versione, oltre ad essere stato eliminato il caso d'uso "Accedi", è stato inserito un nuovo caso d'uso che estende "Compila domanda tirocinio" e consente allo studente di navigare al servizio autocertificazioni.




Prima versione del diagramma dei casi d'uso

[2]gli stati del tirocinio sono stati ampliati rispetto alla specifica di progetto


Seconda versione del diagramma dei casi d'uso

## 2.2 Descrizione degli scenari

Si riporta di seguito la descrizione degli scenari più significativi per ciascun caso d'uso riportato del diagramma.

| | |
|---|---|
| **Nome** | Compila domanda tirocinio |
| **Scopo** | Lo studente fornisce al sistema la documentazione e i dati necessari all'avvio del tirocinio |
| **Attore** | Studente |
| **Pre-condizioni** | L'utente è autenticato con un account studente |
| **Trigger** | L'utente clicca sul tasto "Nuovo tirocinio" |

| | |
|---|---|
| **Descrizione sequenza eventi** | (1) Lo studente inserisce il titolo del tirocinio<br>(2) Lo studente sceglie la tipologia di tirocinio (interno/esterno)<br>(3) Lo studente inserisce il nome del relatore<br>(4) Lo studente inserisce la mail di contatto del relatore<br>(5) Lo studente carica il pdf dell'autocertificazione degli esami sostenuti<br>(6) Lo studente inserisce la data di inizio tirocinio<br>(7) Lo studente inserisce la data di fine tirocinio<br>(8) Lo studente invia la richiesta<br>(9) Il sistema chiede conferma dei dati inseriti<br>(10) Lo studente conferma i dati<br>(11) Il sistema visualizza una schermata di conferma<br>(12) Il sistema invia una mail di notifica allo studente e al relatore<br>(13) Il sistema scrive i dati della richiesta di tirocinio nel database |
| **Alternative** | (5') Lo studente inserisce il nome del correlatore<br>(6') Lo studente inserisce la mail di contatto del correlatore<br>(7') Il processo riprende da (5') se è presente un correlatore aggiuntivo, altrimenti riprende da (5)<br>---<br>(5') Lo studente clicca sul tasto "Servizio autocertificazioni"<br>(6') Lo studente utilizza il servizio per generare il documento richiesto<br>(7') Il processo riprende da (5) |

| | |
|---|---|
| **Post-condizioni** | La richiesta di tirocinio viene registrata nel database dedicato |

| | |
|---|---|
| **Nome** | Approva tirocinio |
| **Scopo** | L'utente approva una proposta di tirocinio effettuata da uno studente |
| **Attore** | Relatore, Commissione tirocini |
| **Pre-condizioni** | L'utente è autenticato con un account relatore o commissione tirocini, esiste almeno un tirocinio che si trova in uno dei due stati di attesa dell'approvazione |
| **Trigger** | L'utente seleziona una richiesta di tirocinio che vede nella propria homepage |
| **Descrizione sequenza eventi** | (1) L'utente visualizza i dati relativi alla richiesta di tirocinio che ha selezionato<br>(2) L'utente clicca su "Approva tirocinio"<br>(3) Il sistema richiede di confermare l'azione<br>(4) L'utente conferma<br>(5) Il sistema torna all'homepage<br>(6) Il sistema riporta l'approvazione nel database<br>(7) Il sistema controlla se il tirocinio è stato approvato sia dal Relatore sia dalla Commissione tirocini, in caso positivo invia una mail di notifica allo studente |
| **Alternativa** | (2') L'utente clicca su "Rifiuta tirocinio"<br>(3') (3', 4' e 5') uguali alla sequenza principale<br>(6') Il sistema riporta il rifiuto nel database<br>(7') Il sistema invia una mail di notifica allo studente |
| **Post-condizioni** | La richiesta di tirocinio viene marcata come approvata o rifiutata dall'utente nel database dedicato |

| | |
|---|---|
| **Nome** | Modifica dati tirocinio |
| **Scopo** | Il relatore può modificare i dati relativi a un tirocinio approvato |
| **Attore** | Relatore |
| **Pre-condizioni** | L'utente è autenticato con un account relatore |

| | |
|---|---|
| **Trigger** | L'utente seleziona una richiesta di tirocinio che vede nella propria homepage |
| **Descrizione sequenza eventi** | (1) L'utente effettua una modifica sui dati del tirocinio che ha selezionato<br>(2) L'utente conferma le modifiche<br>(3) Il sistema torna all'homepage<br>(4) Il sistema riporta le modifiche nel database dedicato |
| **Post-condizioni** | I dati relativi a un tirocinio sono modificati nel database |

| | |
|---|---|
| **Nome** | Termina tirocinio |
| **Scopo** | Il relatore può marcare un tirocinio come terminato |
| **Attore** | Relatore |
| **Pre-condizioni** | L'utente è autenticato con un account relatore, la data indicata come fine del tirocinio è pari o inferiore a quella in avviene questo caso d'uso |
| **Trigger** | L'utente seleziona una richiesta di tirocinio che vede nella propria homepage |
| **Descrizione sequenza eventi** | (1) L'utente clicca sul tasto "Termina tirocinio"<br>(2) Il sistema richiede di confermare la chiusura del tirocino<br>(3) L'utente conferma<br>(4) Il sistema torna all'homepage<br>(5) Il sistema marca il tirocinio come terminato nel database |
| **Post-condizioni** | Il tirocinio è marcato come terminato nel database |

## 2.3 Diagramma delle classi

Si riporta il diagramma delle classi di progetto. Le classi *Docente* e *Studente*, avendo molti attributi in comune, sono state legate con una relazione di ereditarietà alla classe astratta *UtenteUniversità*, inoltre il fatto che la *CommissioneTirocini* è composta da più docenti, è riportato del diagramma con l'utilizzo di una relazione di aggregazione (in quanto l'esistenza di un'istanza di *Docente* non è condizionata alla sua appartenenza alla commissione). La classe centrale è *Tirocinio*, contiene tutti gli attributi necessari (requisiti 1 — 6) alla

memorizzazione di una richiesta di tirocinio e a gestire il suo stato, e tutti i metodi legati alla variazione di stato.

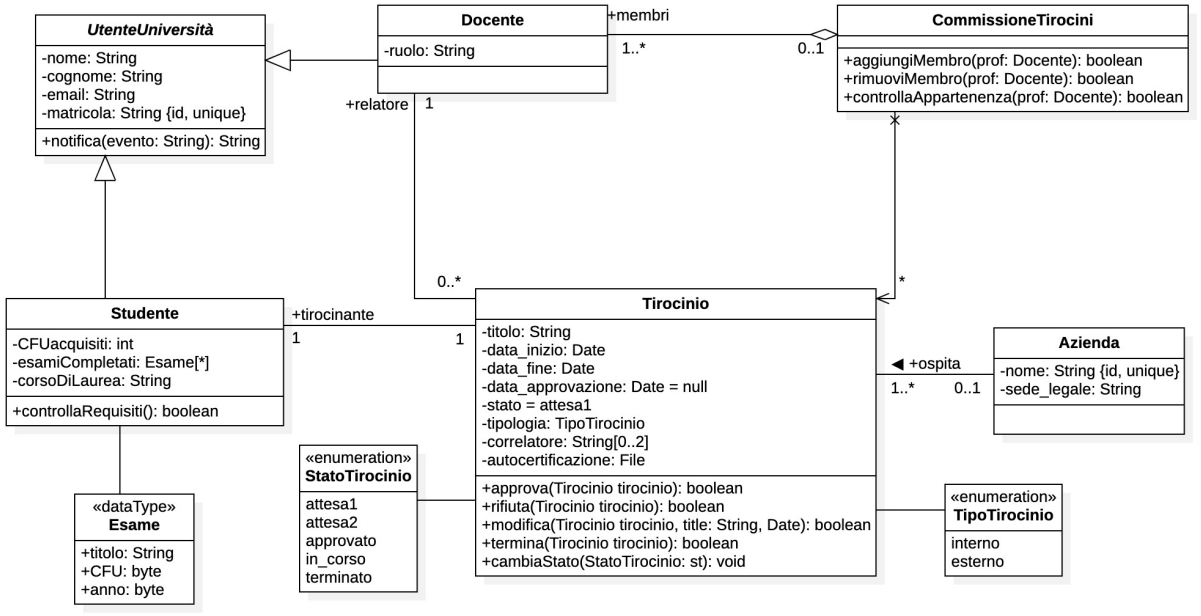



Diagramma delle classi di progetto

## 2.4 Diagrammi di sequenza

Per ogni caso d'uso, viene riportato il rispettivo diagramma di sequenza, le sequenze più significative sono legate ai casi d'uso *Compila richiesta tirocinio* e *Approva tirocinio*.

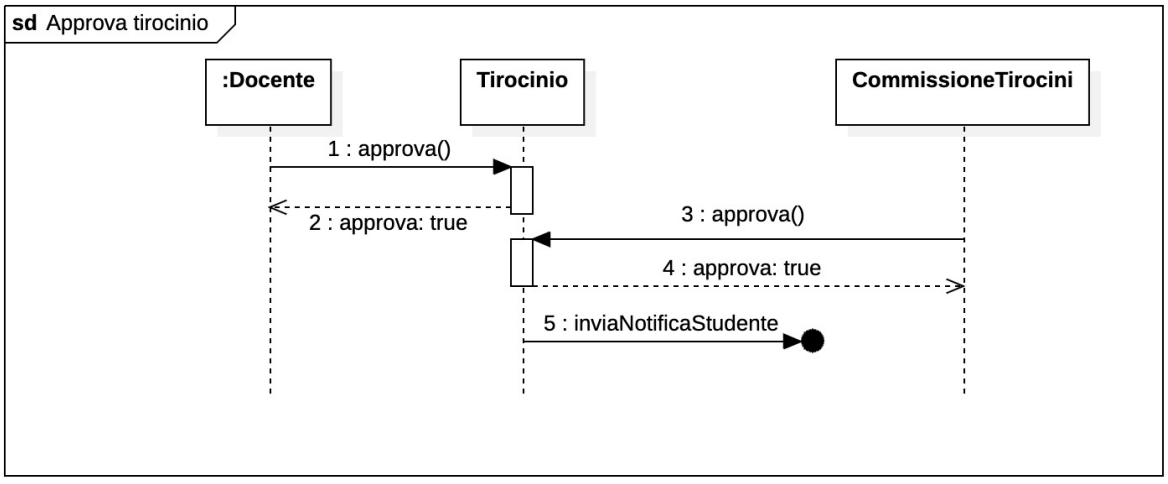



Diagramma di sequenza del caso d'uso *Approva tirocinio*

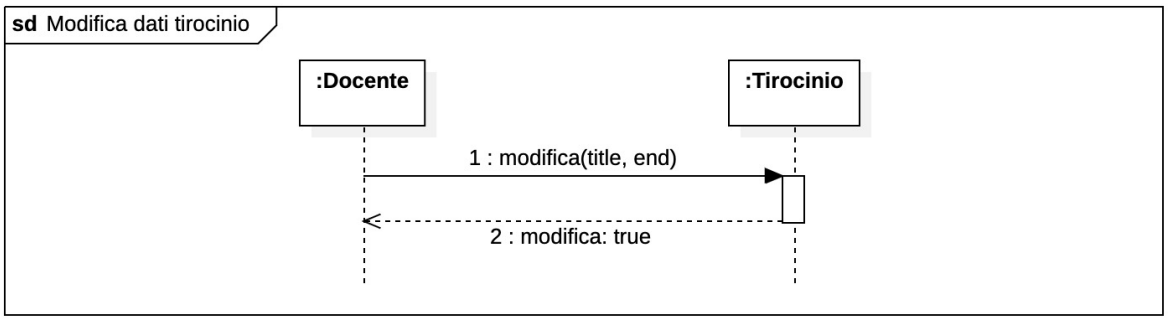



Diagramma di sequenza del caso d'uso *Modifica dati tirocinio*

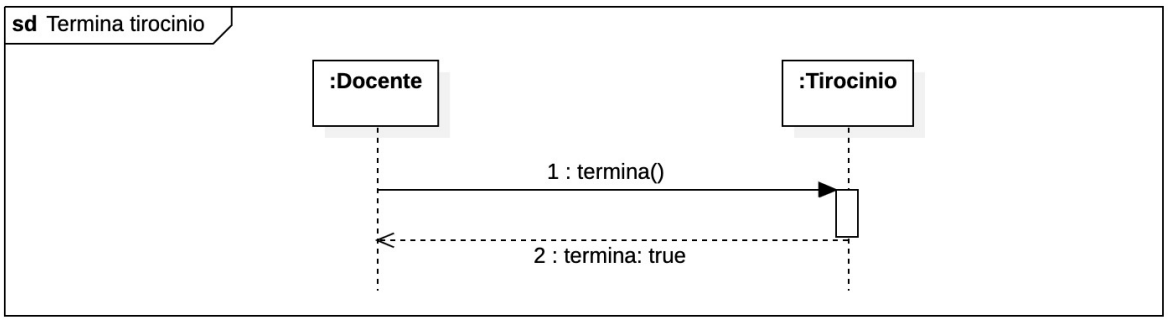



Diagramma di sequenza del caso d'uso *Termina tirocinio*

## 2.5 Diagrammi delle attività

Dal momento che il programma è completamente incentrato sulla gestione del tirocinio, il suo intero funzionamento è riassumibile in un'unica sequenza. Il ciclo di vita di un tirocinio comincia con la compilazione di una richiesta, poi evolve passando dai vari stati in base alle azioni compiute da *Docente* e *Commissione tirocini*. I dati che sono da memorizzare in maniera persistente sono inseriti in nodi *«datastore»* (si veda il diagramma delle classi di programma, sezione 3.1, per i dettagli relativi alla gestione del database).

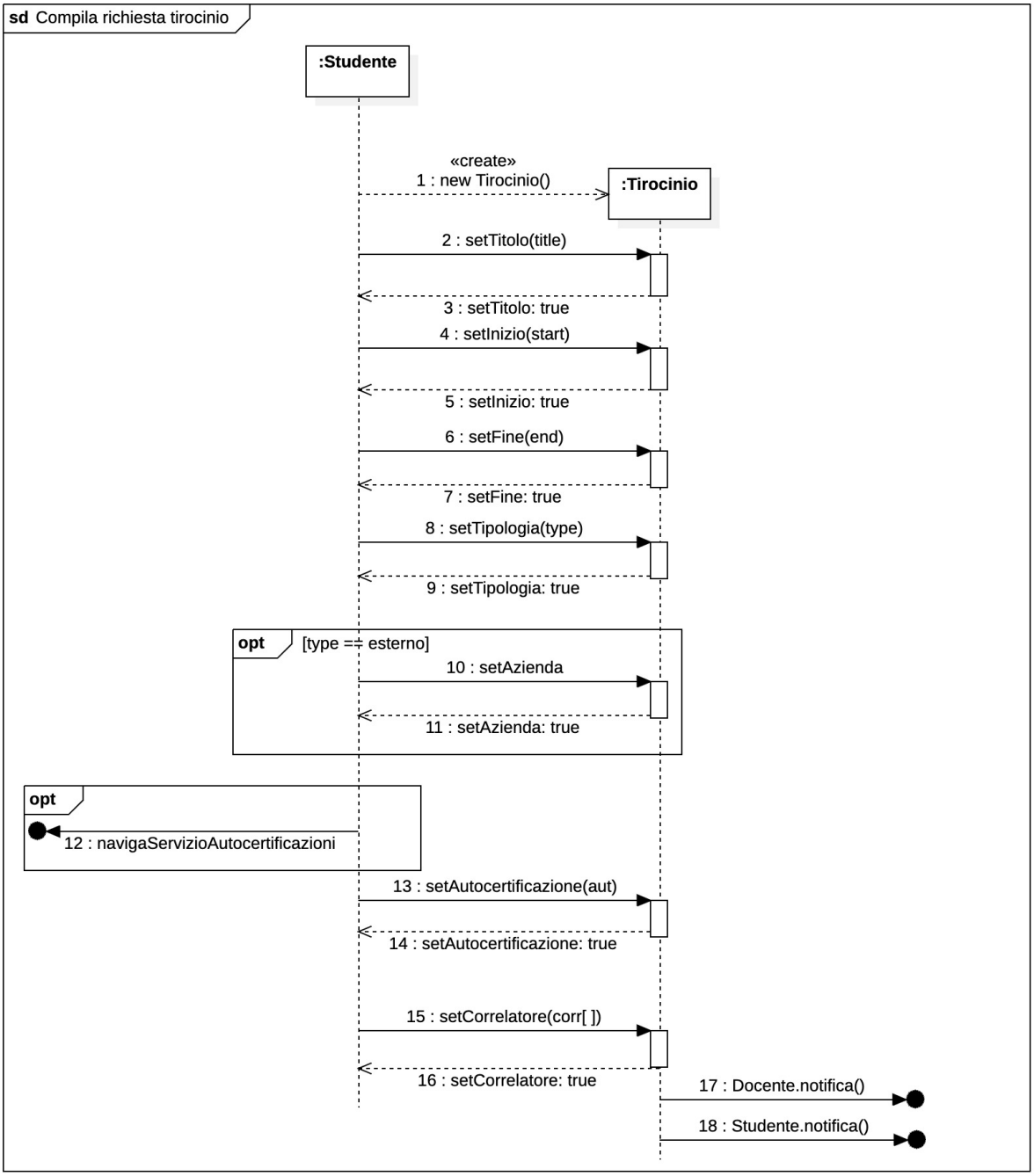



Diagramma di sequenza del caso d'uso *Compila richiesta tirocinio*

## 2.6 Macchine di stato

L'unica entità dell'applicativo che è caratterizzata da stati rilevanti per lo sviluppo del software è il Tirocinio. Come spiegato nel glossario (sezione 1), il Tirocinio si trova nello stato `attesa1` appena viene inserito nel sistema, poi passa nello stato `attesa2` quando viene approvato dal relatore, quando viene approvato anche dalla commissione tirocini passa nello stato `approvato`, a partire dalla data di inizio indicata nella richiesta (evento temporale) il Tirocinio si trova nello stato `in_corso`, quando il relatore decide di terminare il Tirocinio, lo stato passa a `terminato`. Sia quando il Tirocinio si trova nello stato `approvato`, sia quando è `in_corso`, il relatore può effettuare delle modifiche ai dati, questa operazione non

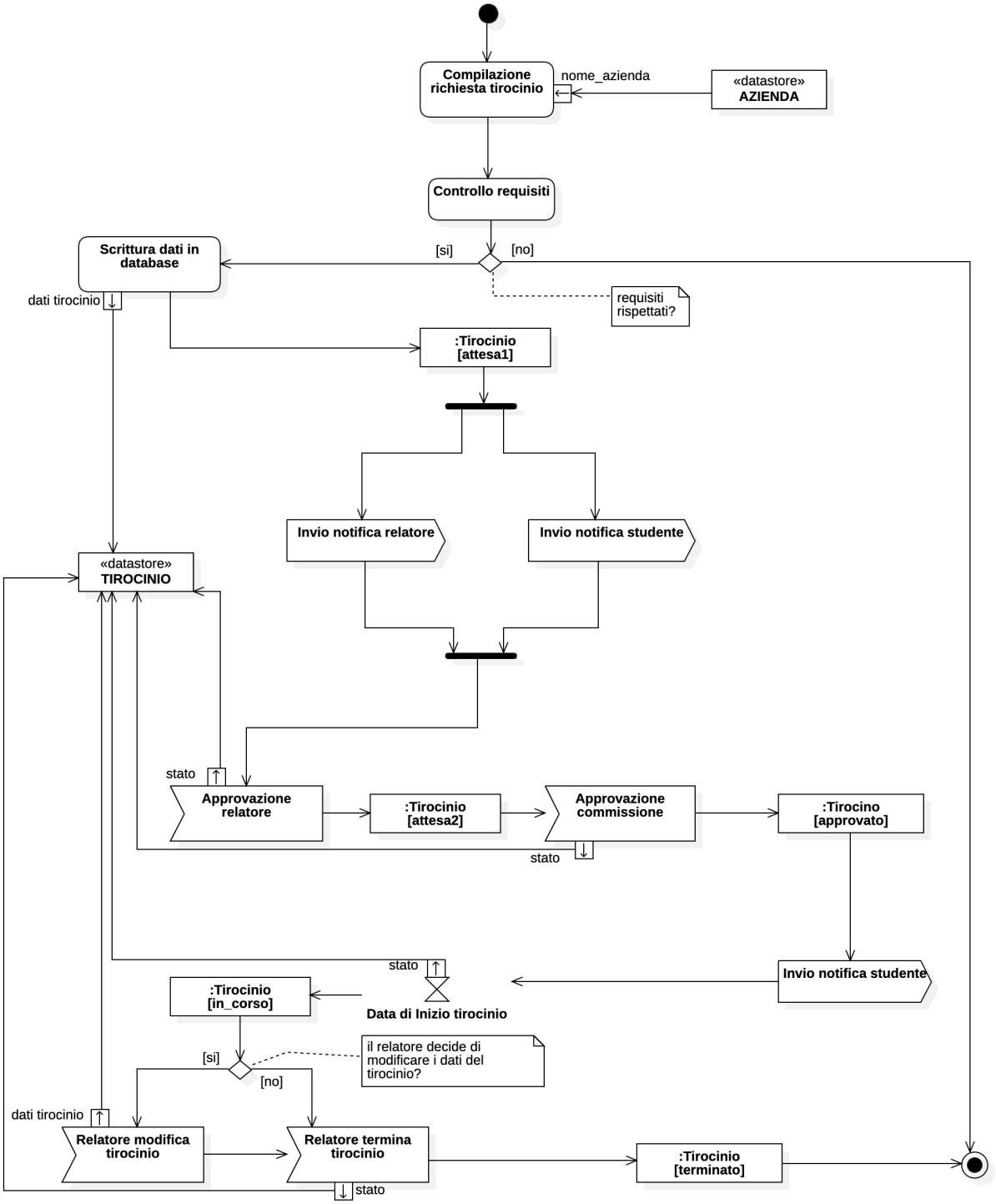



Diagramma di attività

modifica lo stato del tirocinio. Se il relatore o la commissione tirocini rifiutano la proposta di tirocinio, l'oggetto Tirocinio viene distrutto. Si riporta la macchina di stato UML:

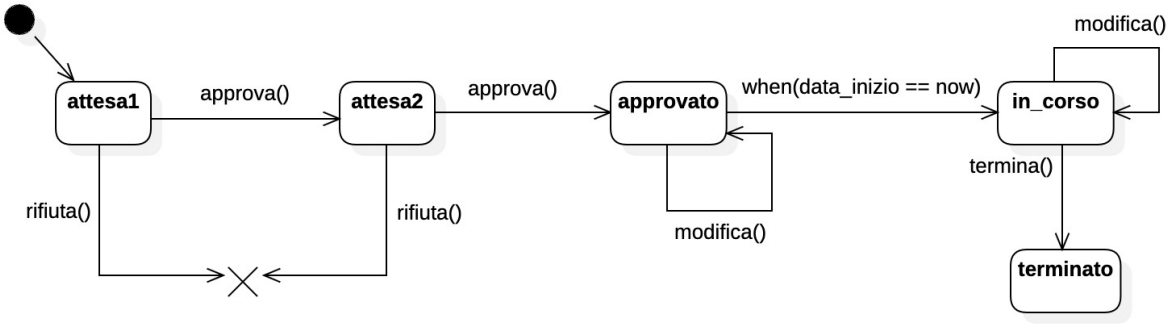



Diagramma della macchina di stato

# 3 Dettagli architetturali del Sistema

## 3.1 Diagramma delle classi di programma

Rispetto al diagramma delle classi di progetto (2.3), questo diagramma è realizzato con l'intento di essere il più possibile vicino al codice, quindi sono stati aggiunti i metodi *getter* e *setter* e le classi che implementano la logica di accesso al database. Sono state aggiunte le classi necessarie ad implementare i design pattern (vedere la sezione seguente) e tutte le classi presenti nel diagramma sono state divise in package.

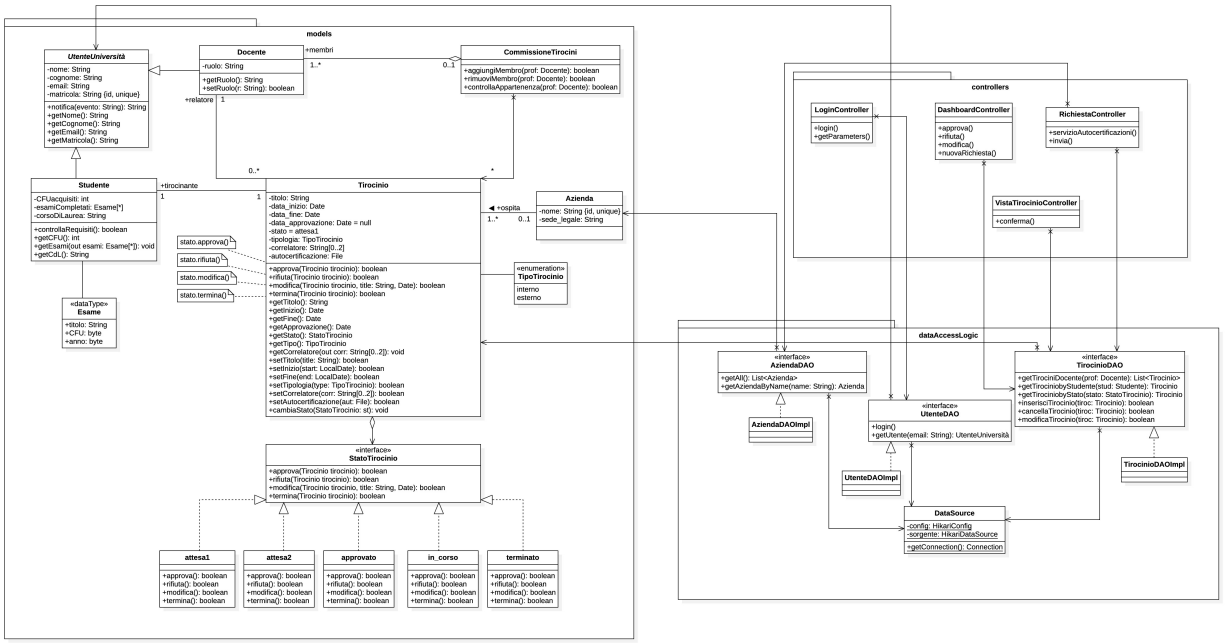



Diagramma delle classi di programma

### 3.1.1 Pattern architetturali

- Model View Controller: l'intero progetto è stato pensato seguendo il pattern MVC, infatti il codice è organizzato in tre packages[3], uno per i modelli, uno per i controllers e uno per le viste, che sono file FXML. Questo pattern consente di dividere le classi che rappresentano il dominio, ovvero i modelli, dall'interfaccia grafica e dalle classi che servono a manipolarla.

- Data Access Object: l'accesso al database è gestito attraverso il pattern DAO, che prevede l'utilizzo di un'interfaccia e una classe che implementa tale interfaccia per creare un oggetto che consente di trasferire i dati da e verso il database.

- State: la classe Tirocinio è stateful e il suo stato è gestito con il pattern State, quindi la classe è in relazione con un'interfaccia che consente di gestire ogni possibile input, poi ogni stato implementa l'interfaccia e realizza i cambiamenti di stato in base all'input ricevuto. Rispetto al diagramma delle classi di progetto, in cui lo stato era gestito con un tipo enumerativo, per implementare il pattern ogni stato è una classe.

**Nota:** la connessione al database *non* è stata implementata utilizzando il pattern singleton, ma sfruttando un meccanismo di connection pooling, implementato con Hikari. La scelta si deve alle performance migliori offerte dal pooling e dal fatto che il singleton può non essere thread safe.

[3]La logica di accesso al database è in un package separato

## 3.2 Diagramma dei componenti

Il programma è rappresentato da un unico componente centrale che contiene i componenti models, views, controllers e dataAccessLogic, e utilizza i servizi offerti da componenti esterni per connettersi al database, per inviare le email di notifica e per il logging. Il sistema di logging è implementato con il framework Apache Log4j, scelto in quanto open source e largamente adottato.

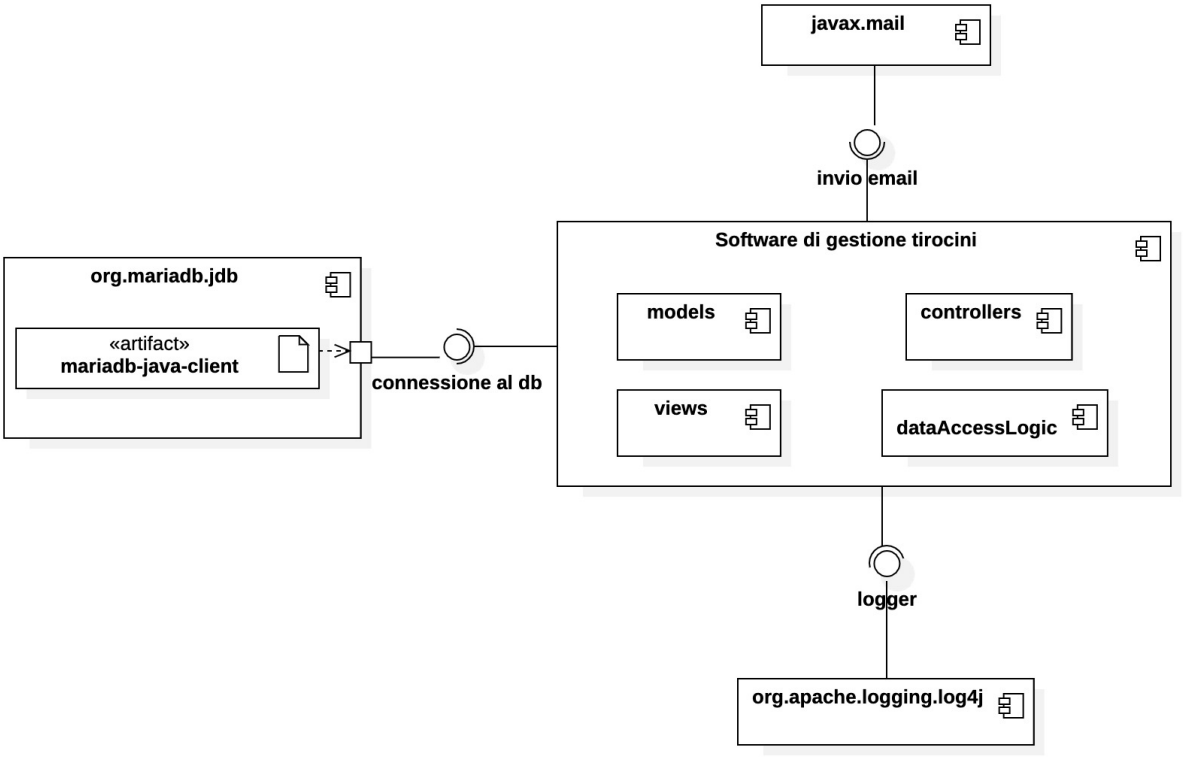



Diagramma dei componenti

## 3.3 Diagramma di deployment

Tutto l'applicativo software viene eseguito sulla macchina dell'utente, anche il database è gestito localmente. Per inviare le email e generare le autocertificazioni, il pc utente si interfaccia con i server dedicati di unimi. La comunicazione con il database avviene utilizzando JDBC, con il server mail è tramite il protocollo SMTP, con il servizio autocertificazioni è tramite il protocollo HTTPS.

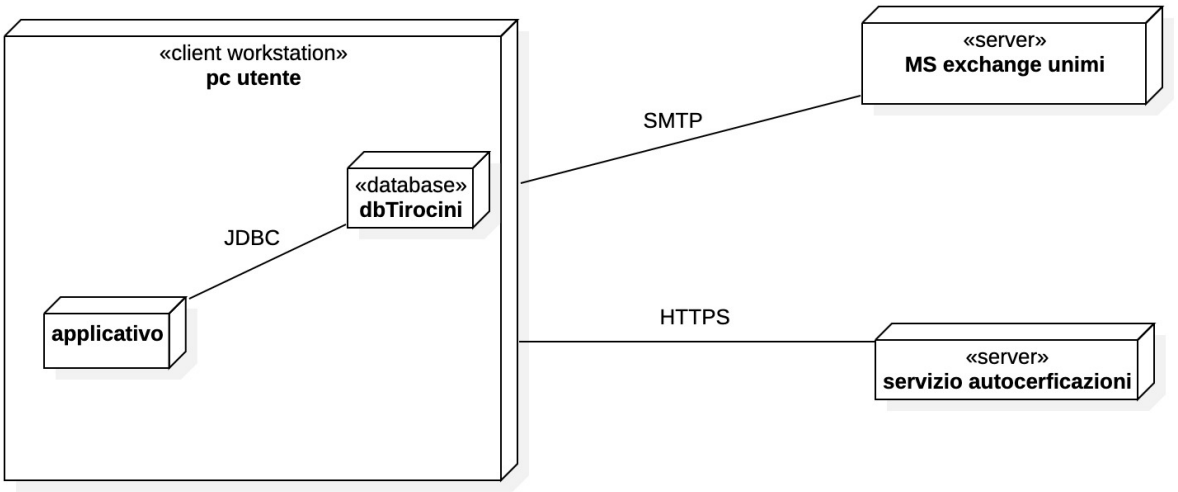



Diagramma di deployment

# 4 Dettagli implementativi

## 4.1 Java e JavaFX

La versione di Java utilizzata è Java SE 8 (SDK Oracle OpenJDK 21.0.1), JavaFX API versione 21.

## 4.2 Software manager

Il progetto è gestito con il tool Apache Maven[4], che consente di gestire le dipendenze delle librerie esterne e implementare i plugin necessari alla compilazione e alla distribuzione del software. La configurazione di Maven è basata sul project object model (POM), nello specifico tutte le informazioni di configurazione, le dipendenze e i plugin sono riportati in un file in eXtensible Markup Language.

## 4.3 Database

Il database relazionale è gestito con il DBMS MariaDB[5], il software si interfaccia al DBMS usando l'API JDBC MariaDB Connector/J, versione 3.3.2.

Il database è composto da 4 tabelle:

[4]https://maven.apache.org
[5]https://mariadb.com

1. *Utente* contiene email, password e tipo;
2. *Studente* contiene tutti gli attributi della classe Studente;
3. *Docente* contiene tutti gli attributi della classe Docente;
4. *Azienda* contiene tutti gli attributi della classe Azienda.

## 4.4 Gestione delle password

Le password sono memorizzate nel database come un hash che viene calcolato utilizzando l'algoritmo bcrypt implementato da BCryptPasswordEncoder di Spring security. Il matching della password è effettuato sempre tramite BCryptPasswordEncoder, è volutamente lento per rendere più difficili gli attacchi, quindi il login impiega circa 5 secondi.

## 4.5 Vincoli

I vincoli di seguito riportati in Object Constraint Language (OCL) sono scritti in Java Modeling Language (JML) direttamente nel codice.

Dal requisito non funzionale n°2: "lo studente deve indicare *obbligatoriamente* uno e un solo relatore [...]".

```
{ context Tirocinio
  inv: self.relatore <> null }
```

Dal requisito non funzionale n°6: "se il tirocinio è esterno, deve essere indicato un tutor esterno oltre al relatore accademico".

```
{ context Tirocinio
  inv: self.tipologia == #esterno implies self.correlatore.length >= 1 }
```

I vincoli di seguito servono per accertarsi che i metodi che cambiano lo stato eseguano i passaggi secondo la macchina di stato riportata nella sezione 2.6.

```
{ context Tirocinio::approva():boolean
  pre: self.oclInState(attesa1) or self.oclInState(attesa2)
  post: self@pre.oclInState(attesa1) implies self.oclInstate(attesa2)
        and self@pre.oclInState(attesa2) implies self.oclInstate(approvato) }

{ context Tirocinio::rifiuta():boolean
  pre: self.oclInState(attesa1) or self.oclInState(attesa2)
  post: self.stato == null }

{ context Tirocinio::modifica():boolean
  pre: self.oclInState(approvato) or self.oclInState(in_corso)
  post: self@pre.oclInState(approvato) implies self.oclInState(approvato)
        and self.oclInState(in_corso) implies self.oclInState(in_corso) }

{ context Tirocinio::termina():boolean
  pre: self.oclInState(in_corso)
  post: self.oclInState(terminato) }
```

```
{ context CommissioneTirocini::aggiungiMembro(Docente prof):boolean
  pre: prof <> null
  post: self.membri -> includes(prof) }

{ context CommissioneTirocini::rimuoviMembro(Docente prof):boolean
  pre: prof <> null
  post: self.membri -> excludes(prof) }
```