

UNIVERSITÀ DEGLI STUDI DI FIRENZE  
SCUOLA DI INGEGNERIA - DIPARTIMENTO DI INGEGNERIA  
DELL'INFORMAZIONE

---

Ingegneria del software

## GESTIONALE DI TESI DI LAUREA

William Zhao, Chaohao Zheng  
Sessione in Aprile  
2022

# Indice

<b>Introduzione</b>	<b>i</b>
<b>1 Progettazione</b>	<b>1</b>
1.1 Use case diagram, templates e mockups . . . . .	1
1.2 Activity Diagram . . . . .	15
1.3 Class Diagram . . . . .	20
1.4 Packaging . . . . .	23
<b>2 Implementazione</b>	<b>25</b>
2.1 Domain model . . . . .	25
2.2 Dao . . . . .	29
2.3 Business logic . . . . .	30
2.4 User interface . . . . .	31
<b>3 Testing</b>	<b>33</b>
3.1 Integration Test . . . . .	33
3.1.1 Publication Cycle Test . . . . .	33
3.2 Unit Test . . . . .	36
3.2.1 Login Test . . . . .	36
<b>4 Demo</b>	<b>37</b>



# Introduzione

## Statement

Il nostro intento è quello di studiare e sviluppare il domain model di un'applicazione a supporto della organizzazione delle tesi di laurea di una scuola, e.g. la Scuola di Ingegneria. L'applicazione di gestione delle tesi di laurea ha lo scopo di facilitare la gestione tramite un'interfaccia molto user-friendly, aiutando così alle persone meno abili nell'utilizzo dei dispositivi elettronici a fare quello che vorrebbero fare, e.g. lo studente che studia filosofia vuole prenotarsi all'appello per la tesi. Questa applicazione visualizza una interfaccia dedicata per ogni utente con un diverso ruolo. Questa applicazione si rivolge agli studenti, ai responsabili amministrativi, ai presidenti dei corsi, ai docenti e al presidente della scuola. Ogni ruolo è fondamentale per completare il ciclo di un appello di tesi.

## Realizzazione

L'applicazione è stata realizzata con il linguaggio di programmazione Java usando l'IDE Eclipse e abbiamo utilizzato il servizio Database MySQL in xampp per la memorizzare dei dati. Per la connettività al database, ci siamo affidati alle librerie Java Database Connectivity (JDBC). I test sono realizzati

---

con JUnit. La grafica è stata sviluppata con le librerie SWT [1]. Infine i diagrams con lo standard UML in starUML.

# Capitolo 1

## Progettazione

### 1.1 Use case diagram, templates e mockups

L'use case diagram contiene molti attori:

1. Presidente del corso di laurea
2. Presidente della scuola
3. Responsabile amministrativo delle tesi
4. Studenti laureandi
5. Relatori
6. Membri della commissione
7. Presidente della commissione

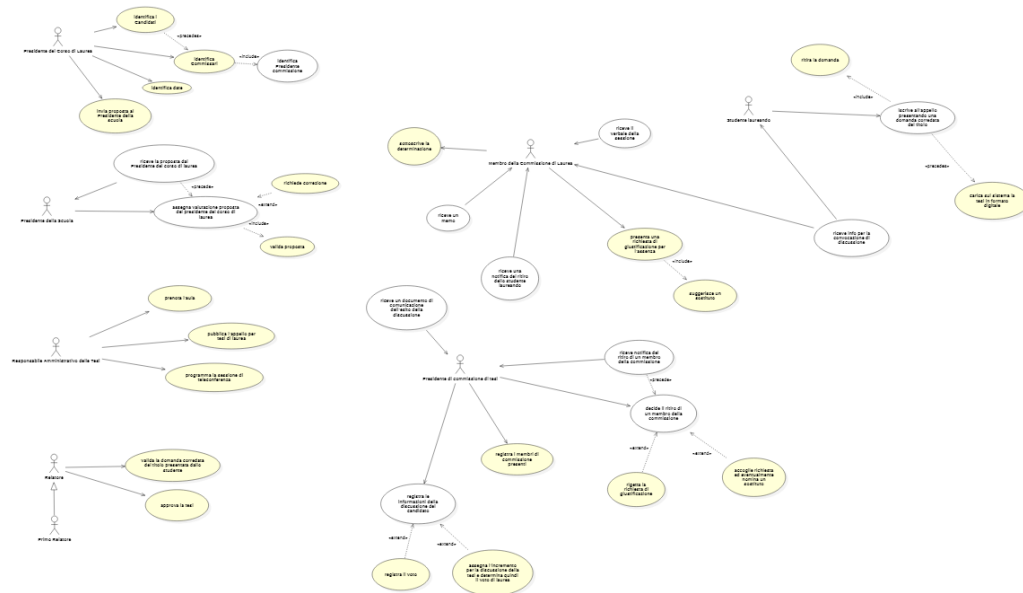


Figura 1.1: Visione completa dell'use case diagram

Partiamo dal responsabile amministrativo delle tesi, perché è lui che da inizio al ciclo di completamente dell'appello. Il responsabile pubblica l'appello per tesi di laurea. Prenota l'aula in cui avverrà la discussione e programma la sessione di teleconferenza che accompagna la discussione in presenza.

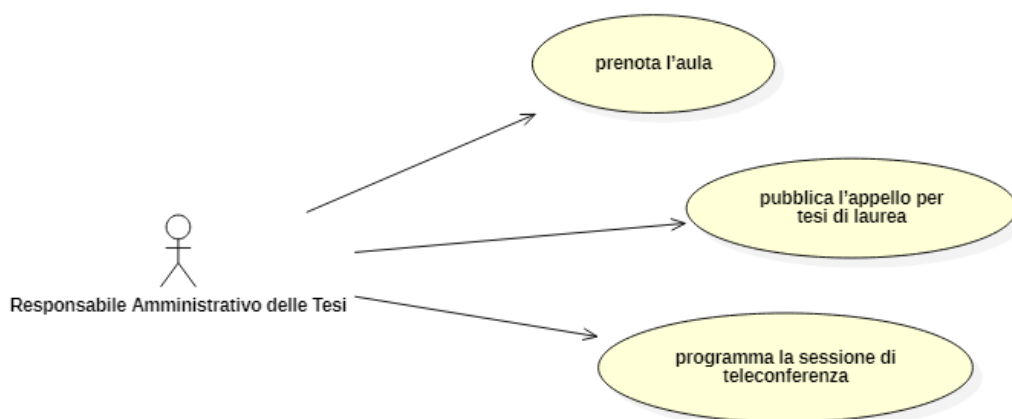


Figura 1.2: Case Diagram: parte del responsabile amministrativo delle tesi

Gli studenti interessati possono allora decidere di iscriversi all'appello senza però sapere ancora la data e ora. Lo studente ha facoltà di ritirare la domanda entro un anticipo definito.

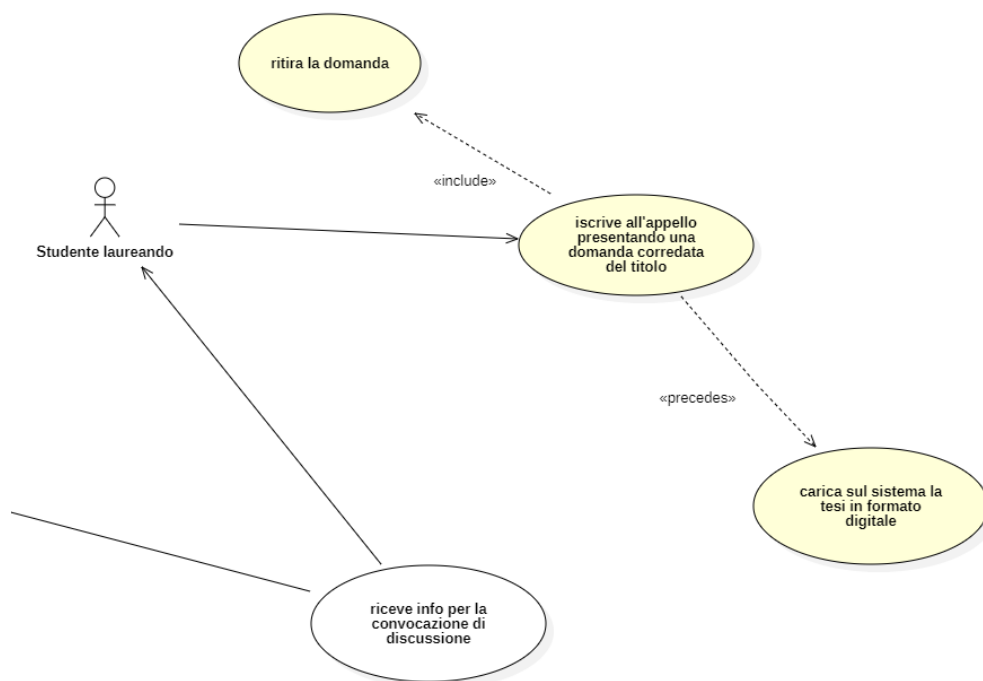


Figura 1.3: Case Diagram: parte dello studente

Il relatore valida la domanda e approva la tesi.

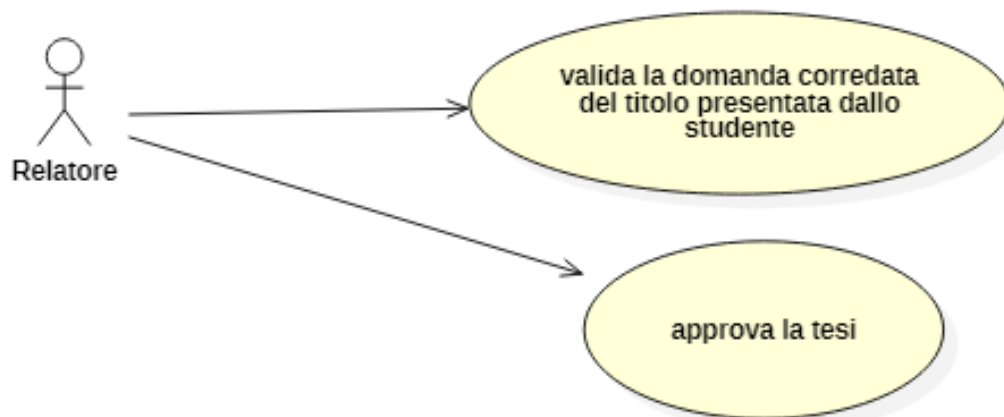




Figura 1.4: Case Diagram: parte del relatore

A seguire c'è il Presidente del corso di laurea. Il Presidente del corso di laurea identifica un insieme di date in cui organizzare la discussione delle tesi degli iscritti, identifica i candidati e una commissione. E fra i membri della commissione identifica un Presidente di commissione di tesi.

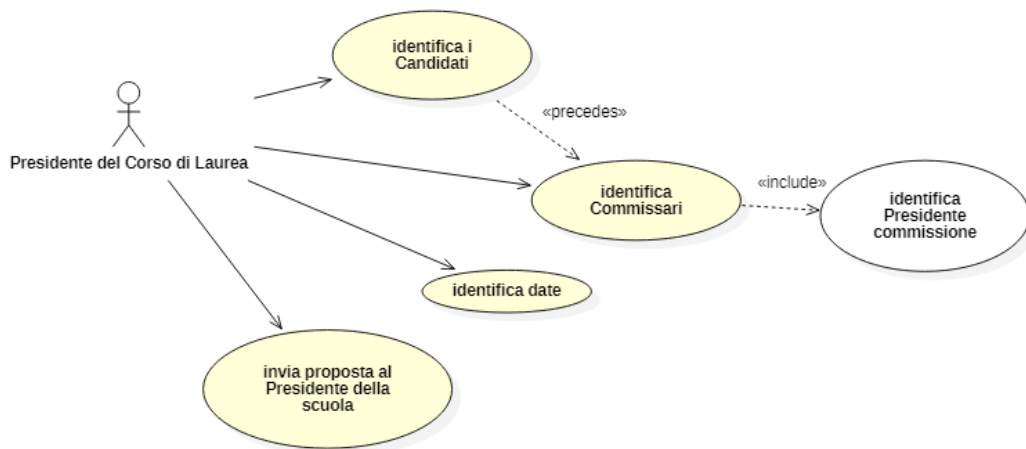


Figura 1.5: Case Diagram: parte del presidente del corso

La proposta dell'appello viene elaborato dal Presidente della scuola. Il Presidente della scuola valida la composizione della commissione o ne richiede correzione.

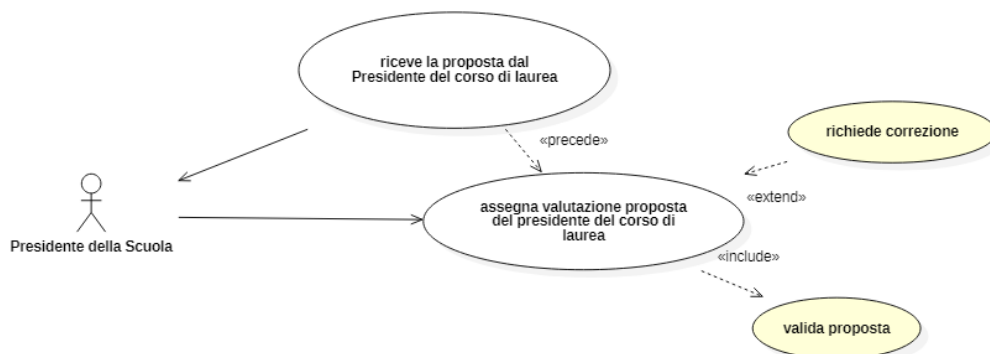


Figura 1.6: Case Diagram: parte del presidente della scuola

Infine arriviamo alla fase di pre-discussione di tesi e post-discussione di tesi, dove i Membri della commissione e Il Presidente della commissione sono ben identificati.

i membri della commissione hanno facoltà di presentare una richiesta di giustificazione per l'assenza accompagnata dall'eventuale suggerimento di un sostituto. Sottoscrivono la determinazione attraverso l'applicazione.

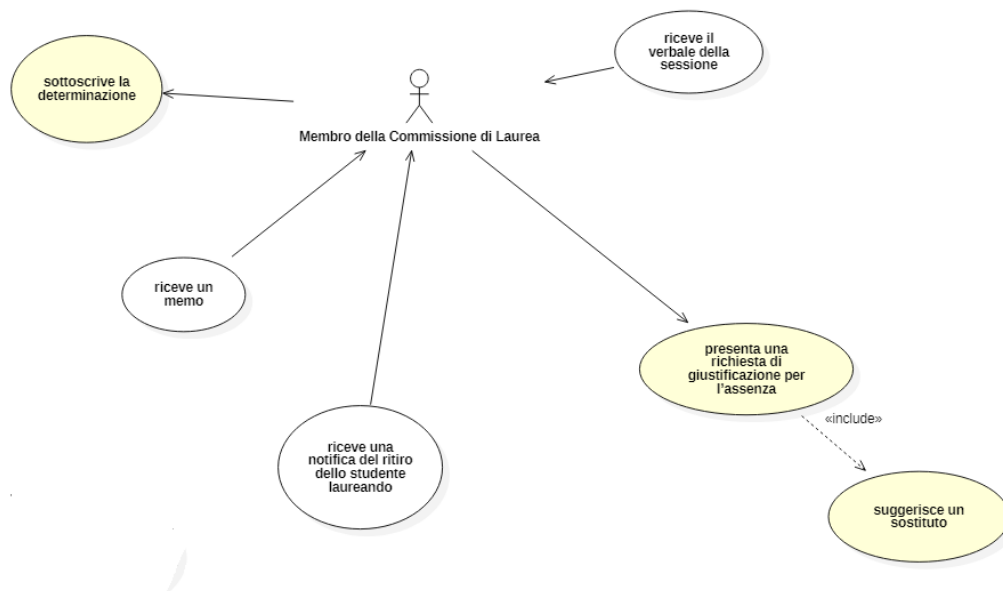


Figura 1.7: Case Diagram: parte del membro di commissione

Con conclusione all'use case diagram abbiamo il nostro Presidente della commissione che rigetta la richiesta di giustificazione di un membro di commissione oppure accoglie e può eventualmente nominare un sostituto. Prima dell'inizio della discussione di tesi registra i membri di commissione presenti e poi al termine della discussione registra il voto o l'incremento assegnato per la discussione della tesi e determina quindi il voto di laurea.

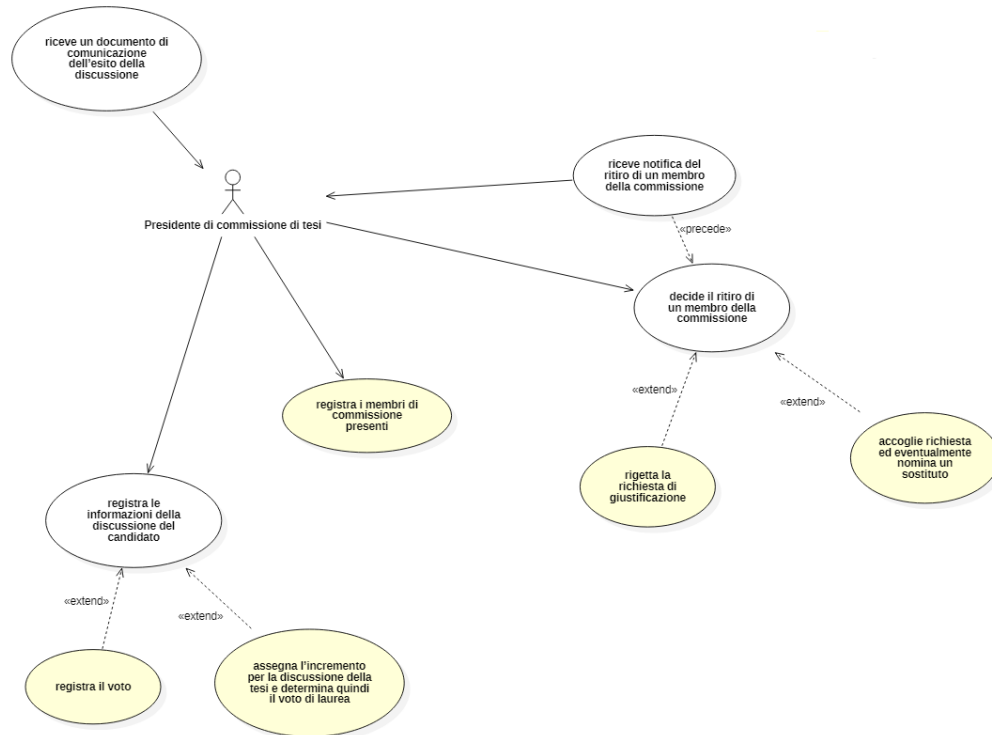


Figura 1.8: Case Diagram: parte del membro di commissione

UC	Pubblica Appello
Level	User Goal
Actor	Responsabile amministrativo
Basic Course	1-il responsabile amministrativo clicca il bottone pubblica appello. vedi figura 1.9 2-il responsabile sceglie il corso. vedi figura 1.10 3-il responsabile clicca conferma 4-il sistema registra l'appello e visualizza un messaggio di completamento. vedi figura 1.11
Alternative Course	3a-il responsabile clicca indietro

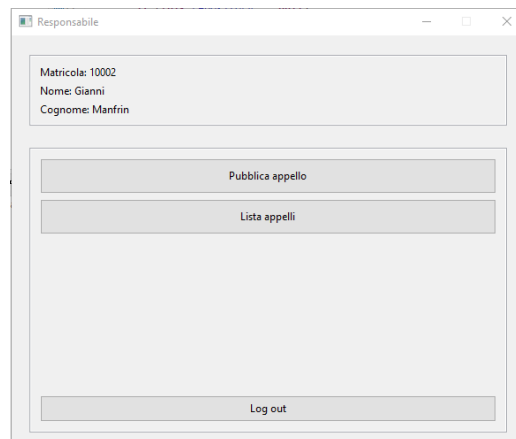


Figura 1.9: Mockup: Responsabile clicca il bottone pubblica appello

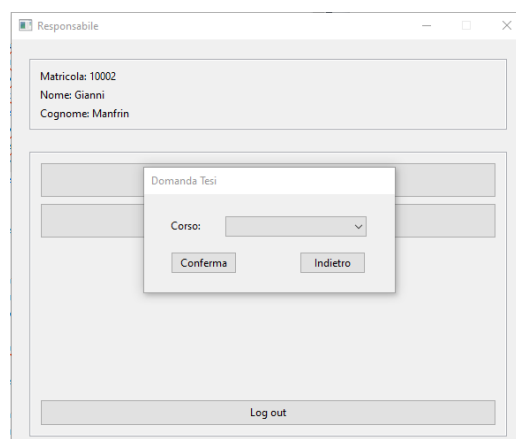


Figura 1.10: Mockup: Responsabile sceglie il corso

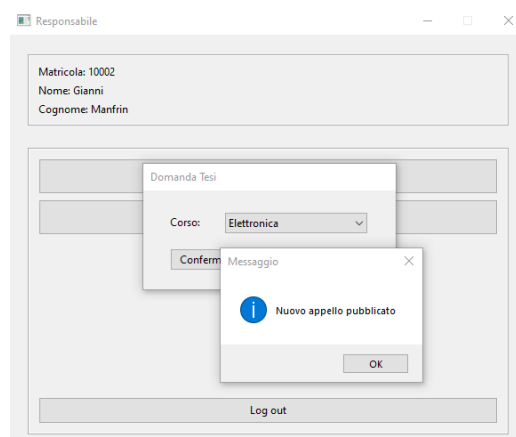


Figura 1.11: Mockup: Applicazione registra l'appello

UC	Prenota Aula e Inserimento Orario
Level	User Goal
Actor	Responsabile amministrativo
Basic Course	<p>1-il responsabile amministrativo clicca il bottone lista appelli.</p> <p>2-il responsabile sceglie l'appello e clicca il bottone dettagli. vedi figura 1.12</p> <p>3-il responsabile clicca sul bottone prenota aula e inserisci orario. vedi figura 1.13</p> <p>4-il responsabile seleziona l'aula e inserisce un'orario vedi figura 1.14</p> <p>5-il sistema registra l'aula e l'orario</p>
Alternative Course	<p>2a-il responsabile clicca indietro</p> <p>3a-il responsabile esce dalla finestra</p> <p>4a-il responsabile clicca indietro</p>

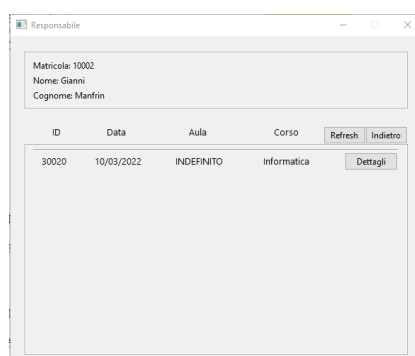


Figura 1.12: Mockup: finestra appelli

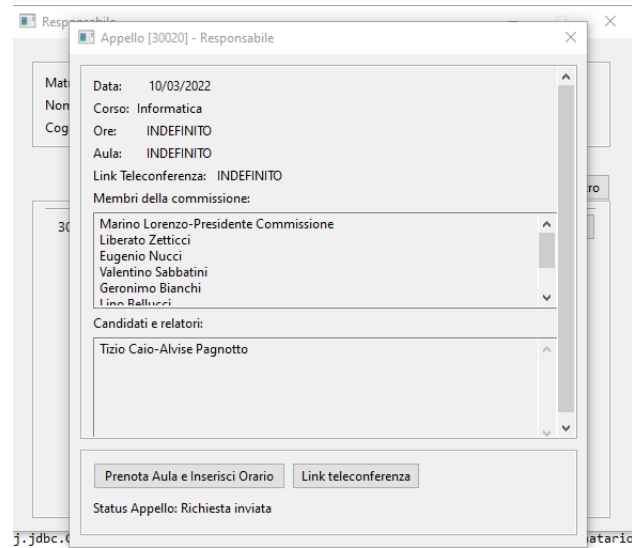


Figura 1.13: Mockup: finestra dettagli

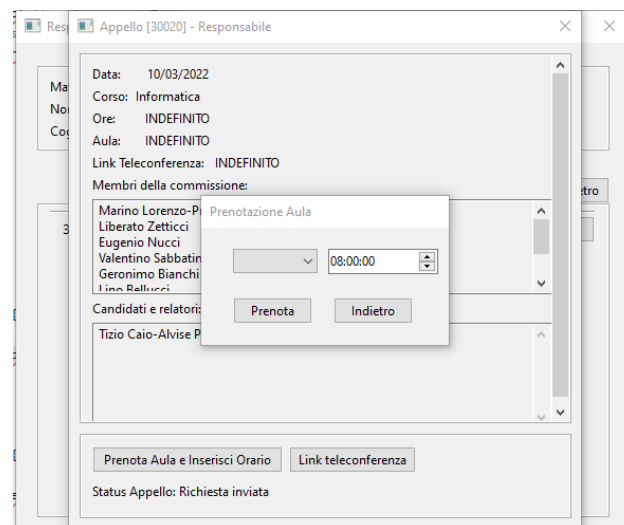


Figura 1.14: Mockup: finestra prenota aula e inserisci orario

UC	Iscrive all'appello
Level	User Goal
Actor	Studente laureando
Basic Course	1-lo studente laureando clicca sul bottone iscrizione tesi. vedi figura 1.15 2-lo studente sceglie il corso di appartenenza, sceglie il relatore e clicca conferma. vedi figura 1.16 3-lo studente riceve conferma. vedi figura 1.17 4-il sistema registra la sua domanda
Alternative Course	2a-lo studente clicca indietro

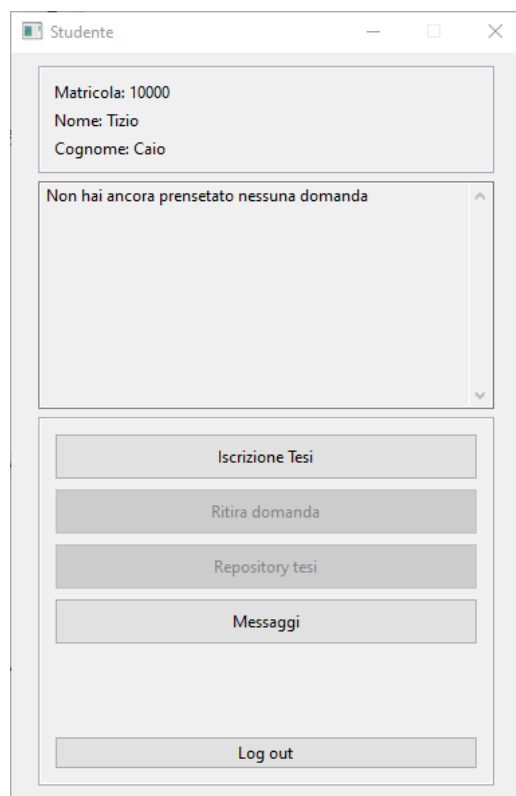


Figura 1.15: Mockup: interfaccia utente dello studente

The mockup shows a window titled "Studente" with a standard OS title bar (minimize, maximize, close buttons). Inside the window, there is a header section with the following text:

Matricola: 10000  
Nome: Tizio  
Cognome: Caio

Below the header, a message states: "Non hai ancora prestatato nessuna domanda".

A modal dialog box titled "Domanda Tesi" is open in the center. It contains two dropdown menus:

Corso: Informatica (with a downward arrow)  
Relatore: Alvise Pagnotto (with a downward arrow)

At the bottom of the dialog are two buttons: "Conferma" and "Indietro".

Below the dialog, there are three buttons in the main window: "Repository tesi", "Messaggi", and "Log out".

Figura 1.16: Mockup: finestra domanda tesi



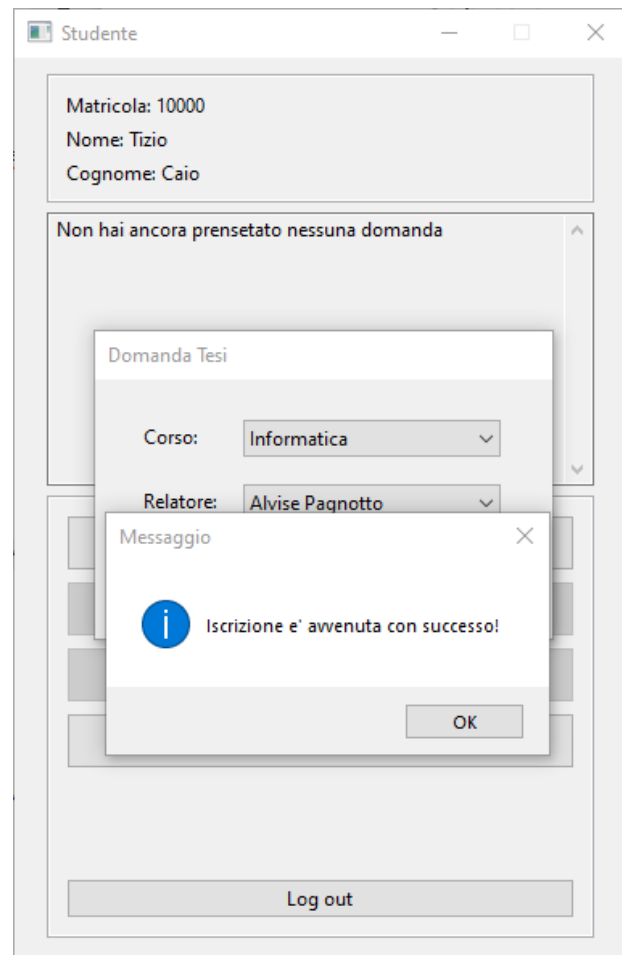


Figura 1.17: Mockup: l'applicazione dà conferma

UC	Identifica i candidati e commisari
Level	User Goal
Actor	Presidente del corso di laurea
Basic Course	<p>1-il presidente del corso di laurea clicca visualizza gli appelli di tesi. vedi figura 1.18</p> <p>2-il presidente clicca dettagli sull'appello che gli interessa vedi figura 1.19</p> <p>3-il presidente clicca sul bottone identifica membri vedi figura 1.20</p> <p>4-il presidente seleziona gli studenti-relatori e li aggiunge vedi figura 1.21</p> <p>5-il presidente seleziona i docenti e li aggiunge</p> <p>6-il presidente preme conferma e riceve conferma 1.22</p> <p>7-il sistema lo registra</p>
Alternative Course	<p>2a-il presidente clicca indietro</p> <p>3a-il presidente chiude la finestra</p> <p>6a-il presidente rimuove tutti i membri</p> <p>6a-chiude la finestra</p>

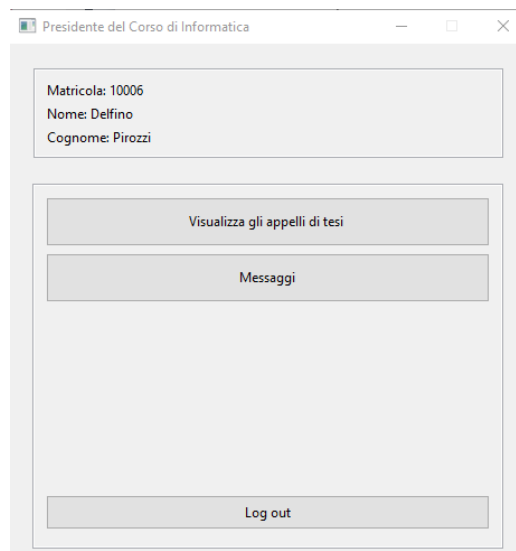
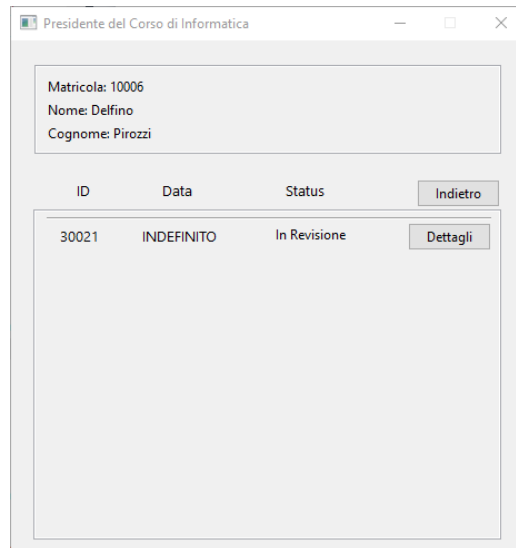


Figura 1.18: Mockup: interfaccia utente del presidente del corso

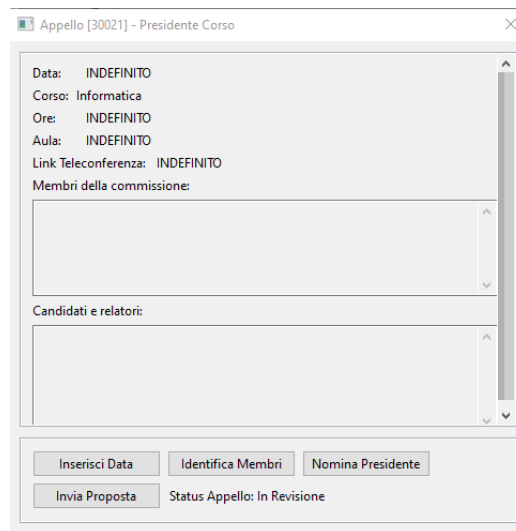


Presidente del Corso di Informatica

Matricola: 10006  
Nome: Delfino  
Cognome: Pirozzi

ID	Data	Status	Indietro
30021	INDEFINITO	In Revisione	Dettagli

Figura 1.19: Mockup: appelli disponibili



Appello [30021] - Presidente Corso

Data: INDEFINITO  
Corso: Informatica  
Ore: INDEFINITO  
Aula: INDEFINITO  
Link Teleconferenza: INDEFINITO  
Membri della commissione:

Candidati e relatori:

Inserisci Data Identifica Membri Nomina Presidente  
Invia Proposta Status Appello: In Revisione

Figura 1.20: Mockup: interfaccia utente appello del presidente del corso

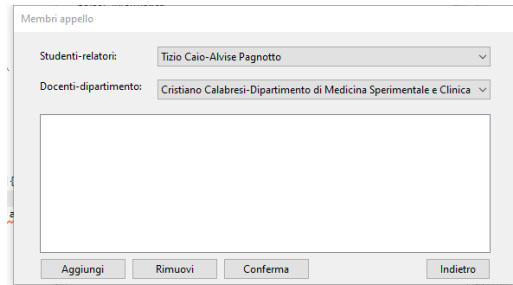


Figura 1.21: Mockup: interfaccia utente per l'identificazione dei membri

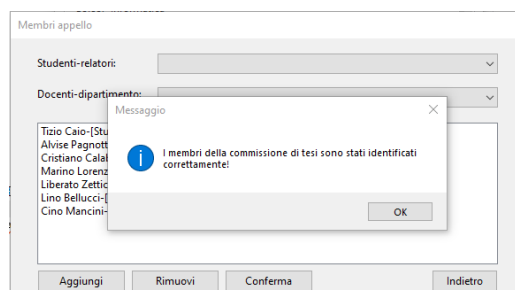


Figura 1.22: Mockup: applicazione visualizza messaggio di conferma d'identificazione

## 1.2 Activity Diagram

L'activity diagram è composto da 3 fasi. La prima fase è quella di identificazione dei candidati, la seconda fase è di organizzazione della discussione, l'ultima fase è quella di discussione.

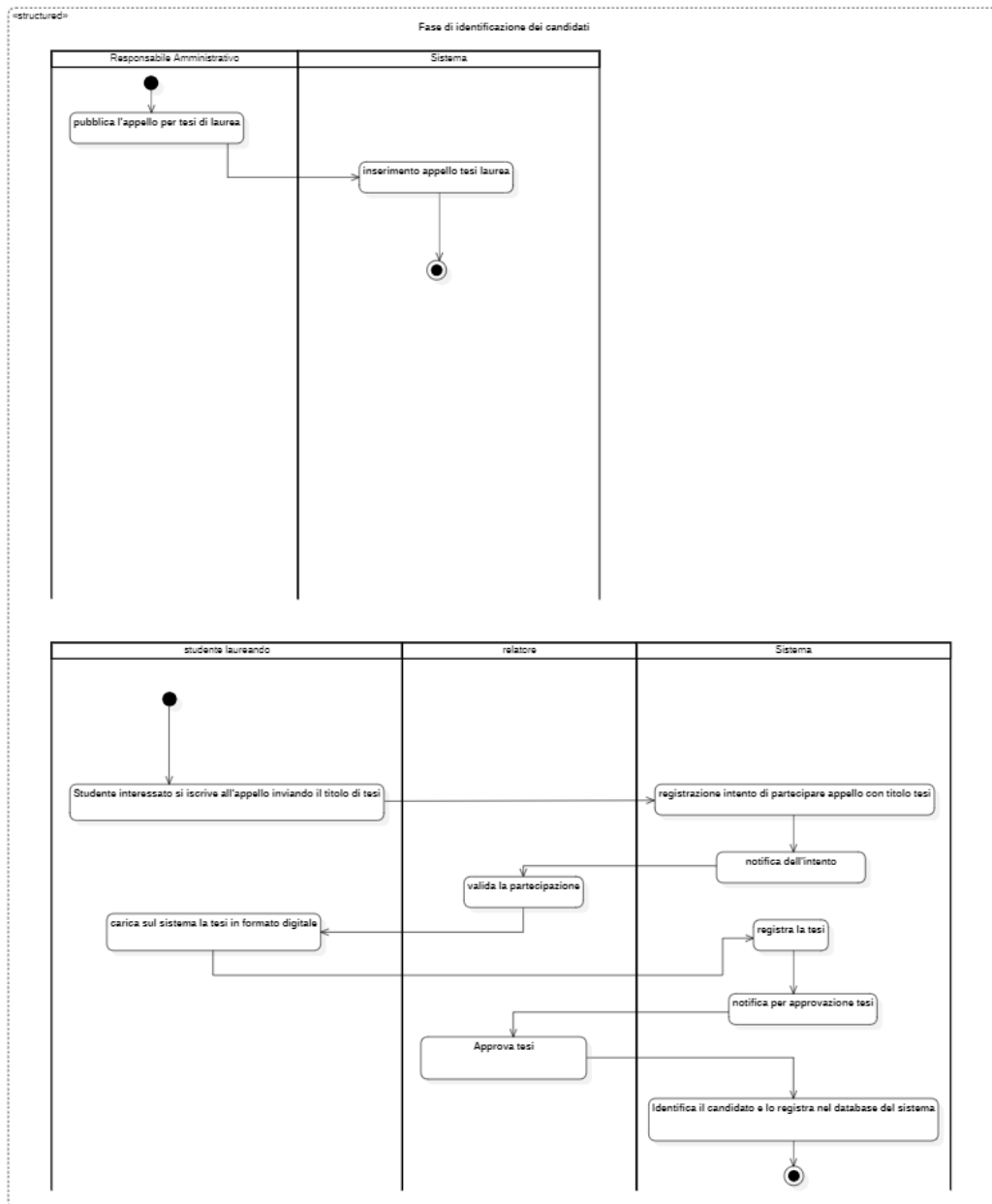


Figura 1.23: Prima fase dell'activity diagram

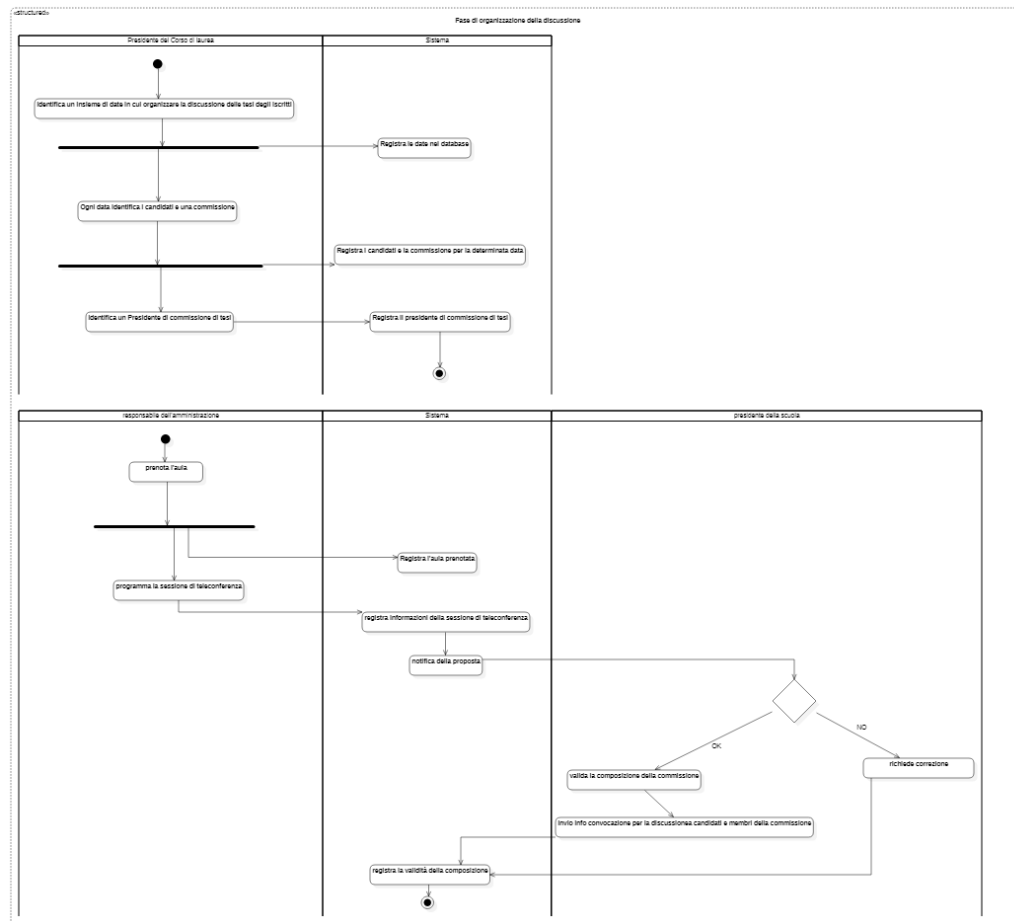


Figura 1.24: Seconda fase dell'activity diagram

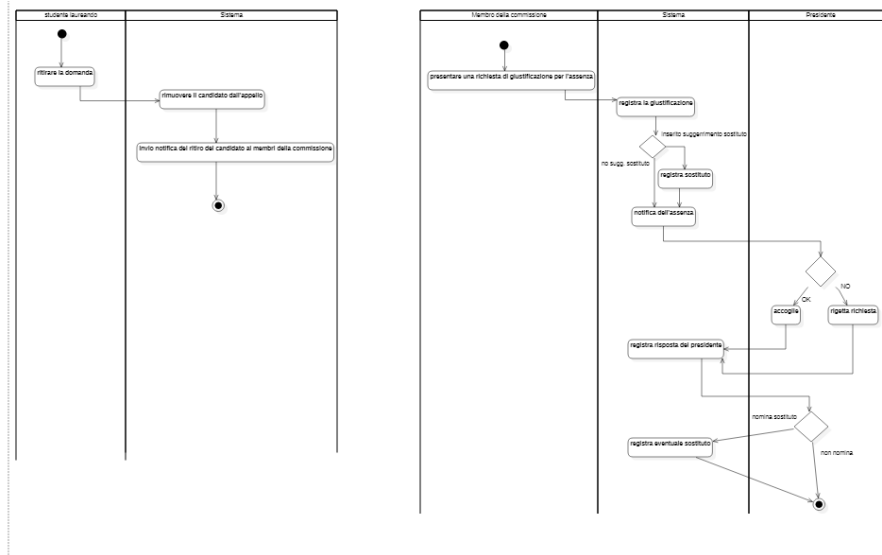


Figura 1.25: Seconda fase dell'activity diagram

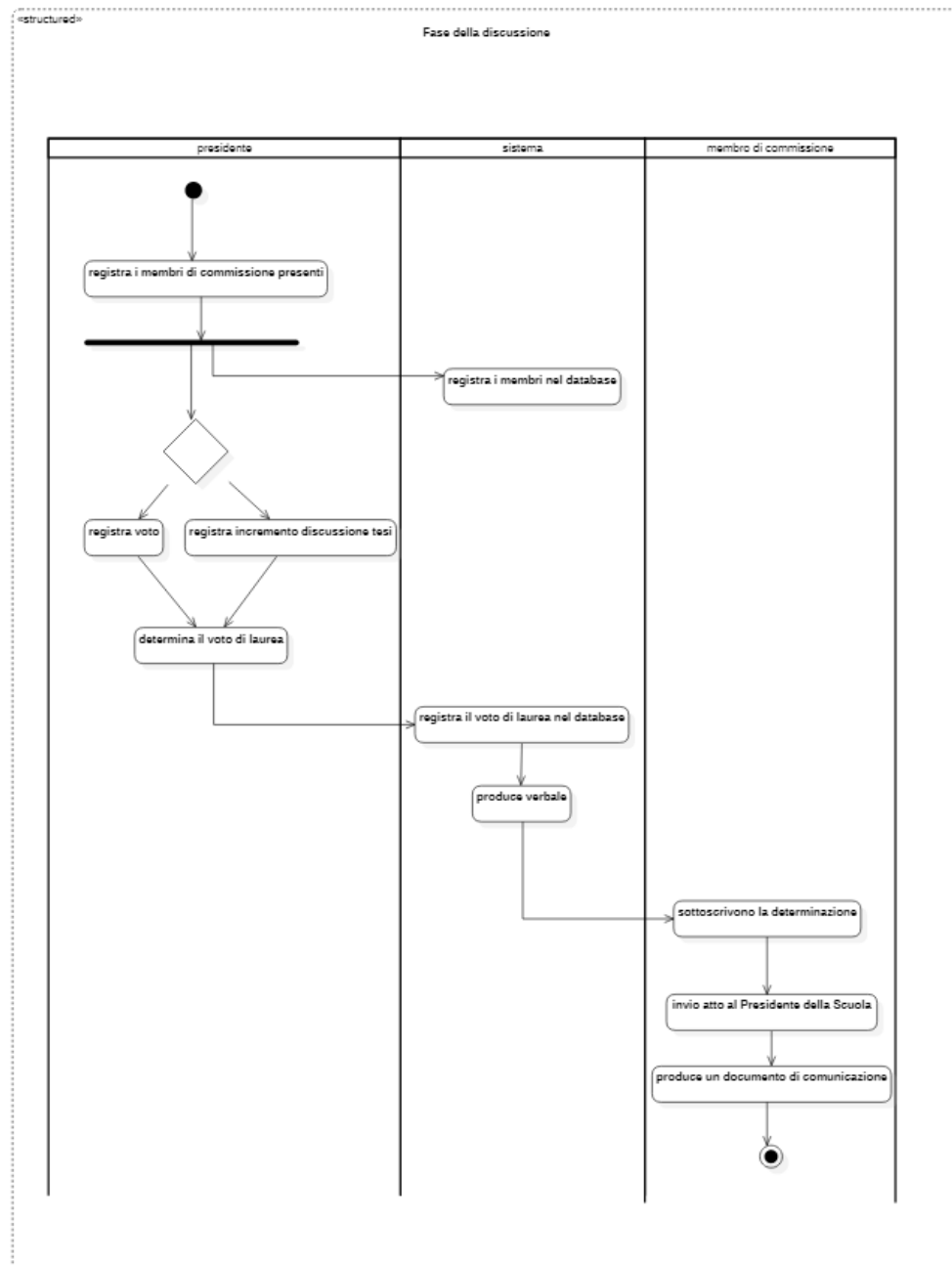


Figura 1.26: Terza fase dell'activity diagram



## 1.3 Class Diagram

Il class diagram è composto da tre packages:

1. domainModel: descrive le varie entità che fanno parte nel sistema (models)
2. userInterface: contiene le interfacce grafiche che l'utente andrà a interagire (views)
3. businessLogic: si riferisce a tutta quella logica di elaborazione che rende operativa un'applicazione (controllers).

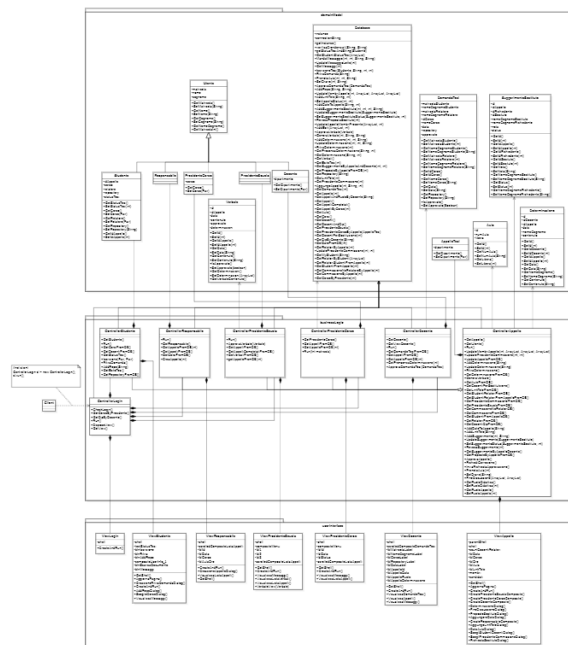


Figura 1.27: Visione completa del class diagram

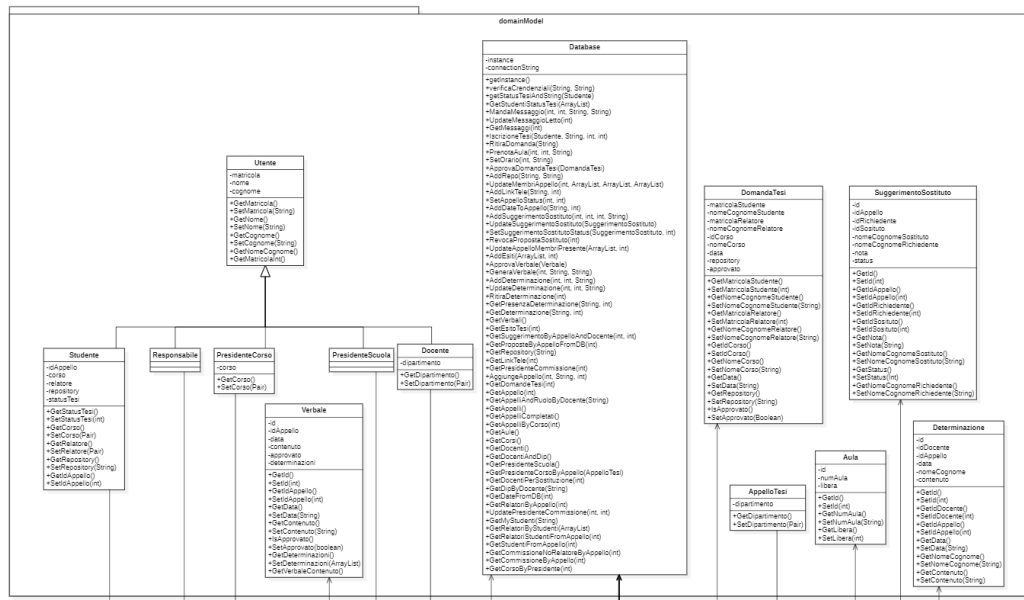


Figura 1.28: Class Diagram: DomainModel

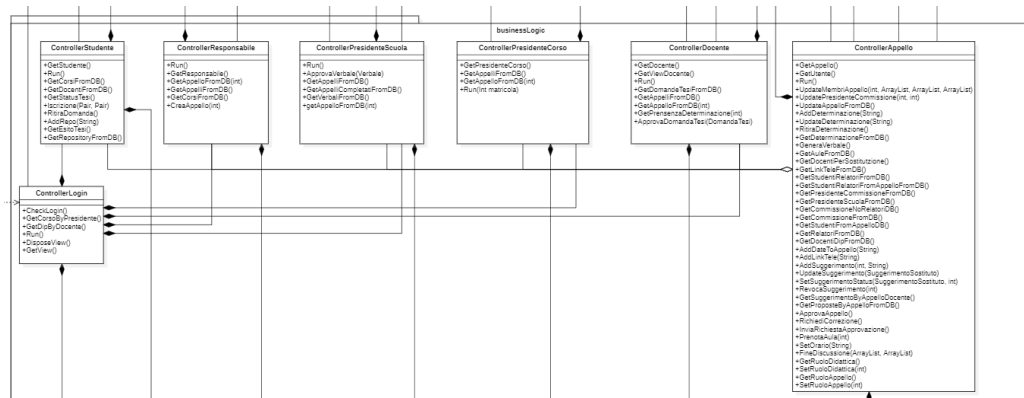


Figura 1.29: Class Diagram: BusinessLogic

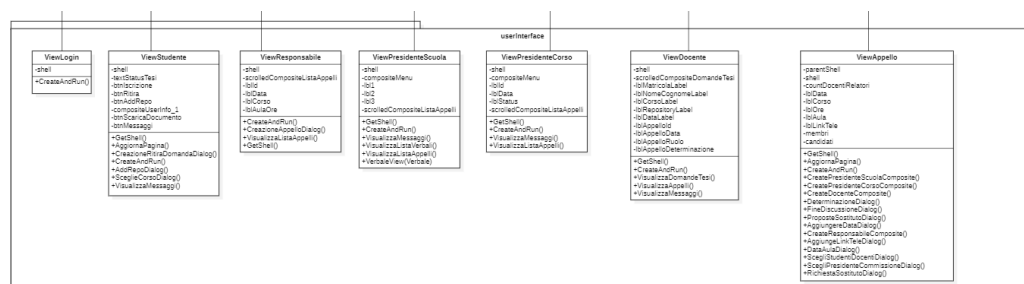


Figura 1.30: Class Diagram: UserInterface

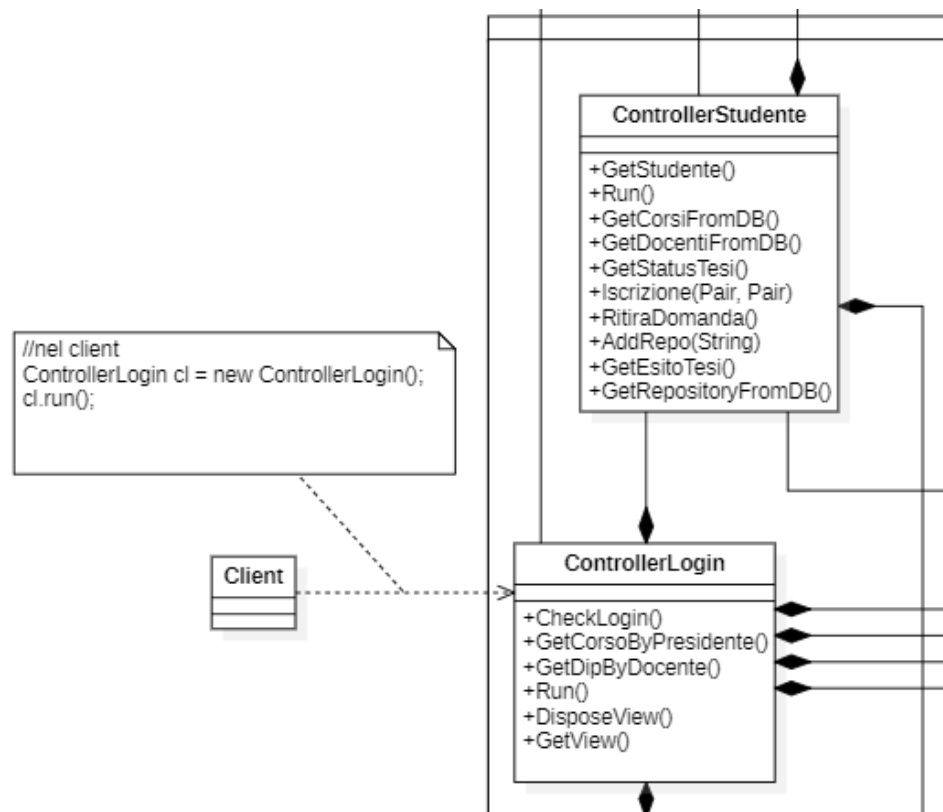


Figura 1.31: Class Diagram: Client

## 1.4 Packaging

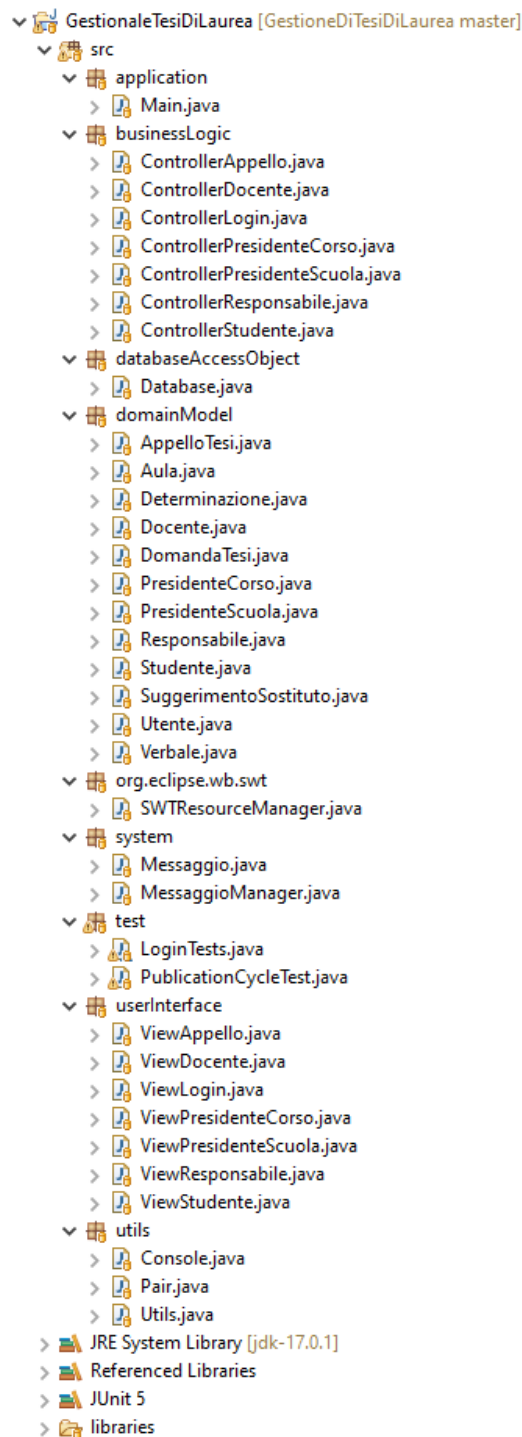


Figura 1.32: Java Packaging in Eclipse

# Capitolo 2

## Implementazione

Nella realizzazione del progetto è stato utilizzato il pattern architetturale MVC che separa la logica di presentazione dei dati dalla logica di business. I models stanno nel package domain model, i controllers nella business logic e i view nell'userInfo.

### 2.1 Domain model

La parte del domain model descrive le varie entità che fanno parte o hanno rilevanza nel sistema e le loro relazioni. I models forniranno dei metodi che i controllers andranno a utilizzare per accedere ai dati utili per l'applicazione. Nel nostro caso i models sono:

1. Studente
2. Responsabile
3. PresidenteCorso
4. PresidenteScuola

- 5. Utente
- 6. Docente
- 7. Aula
- 8. AppelloTesi
- 9. DomandaTesi
- 10. SuggerimentoSostituto
- 11. Verbale
- 12. Determinazione

Le classi `Studente`, `Responsabile`, `PresidenteCorso`, `PresidenteScuola` e `Docente` sono ereditate dalla classe padre `Utente`.

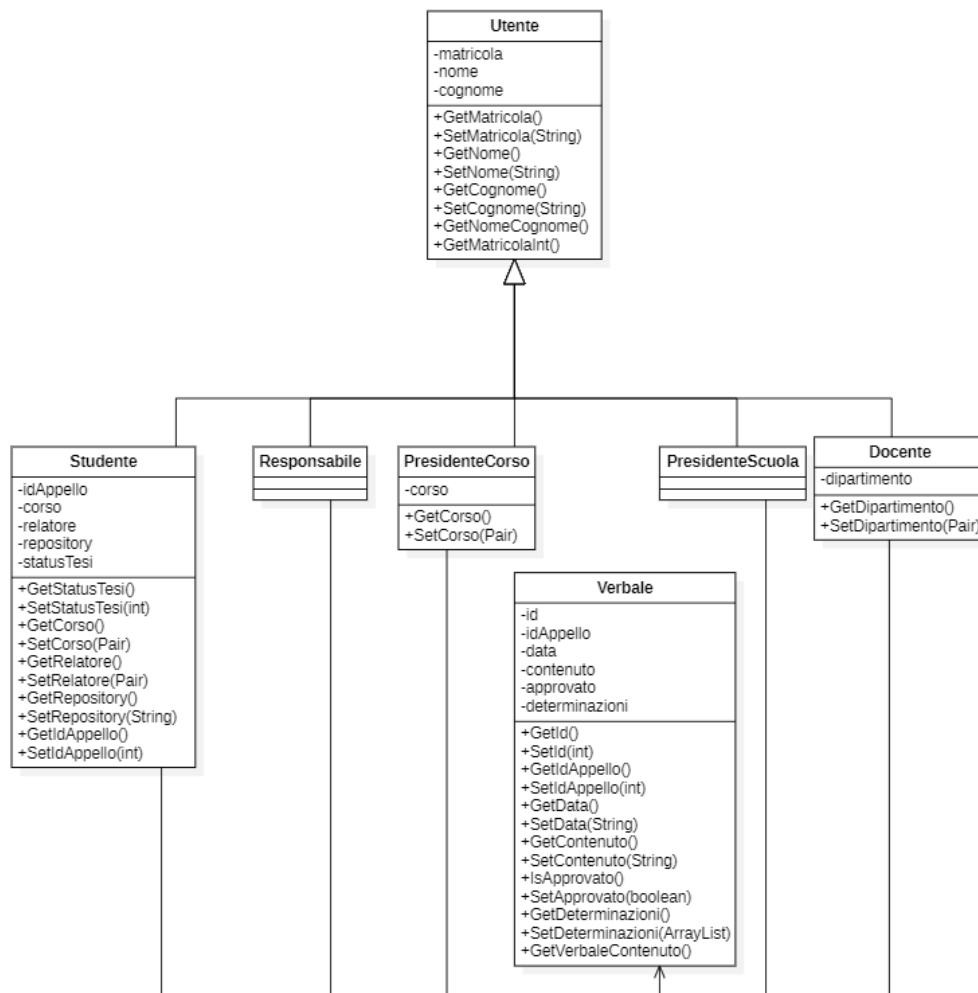


Figura 2.1: Classi ereditate da utente

Ogni utente darà informazioni riguardanti ai propri dati personali, quindi matricola, il nome, il cognome, il corso associato.

Le restanti viste nel class diagram:



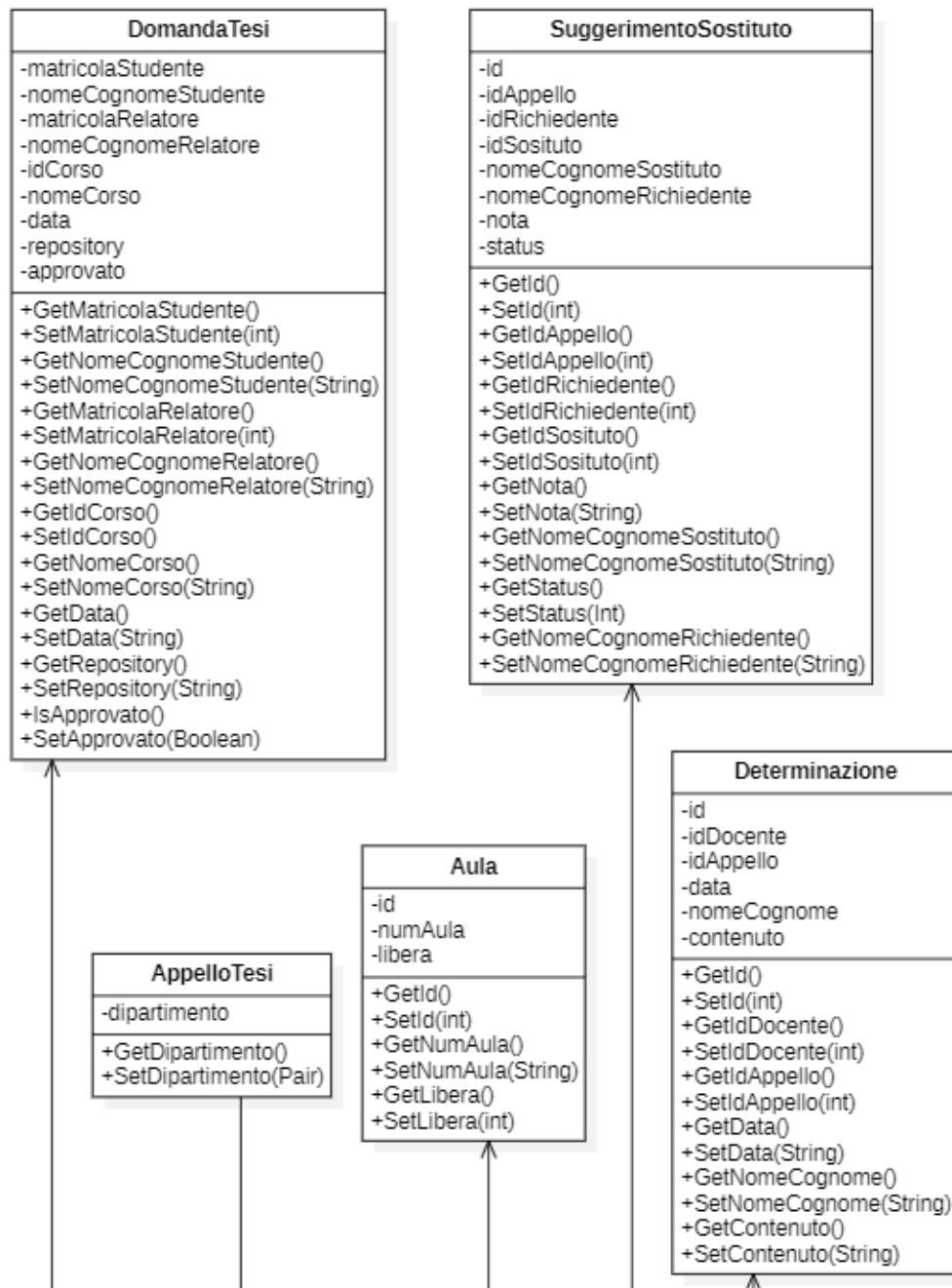


Figura 2.2: Classi in models

## 2.2 Dao

Il Dao (Database access object) è un pattern architetturale che gestisce la parte della persistenza dei dati. Fornisce molti metodi nei quali i controllers li utilizzeranno per la lettura o scrittura dei dati nel database tramite query. Le classi controllers accederanno al database tramite il pattern singleton.



Figura 2.3: Database

Il database si occuperà di registrare gli appelli, estrarre informazioni dell'appello, registrare studenti laureandi e docenti in un appello, estrarre

informazioni dei corsi ed ecc...

## 2.3 Business logic

La business logic è la parte che fa partire diciamo il motore di avviamento. Perché è essa che crea le istanze di views e models per poi coordinarli in modo tale che l'applicazione funzioni nella maniera giusta. I controllers sono:

1. ControllerAppello
2. ControllerDocente
3. ControllerLogin
4. ControllerPresidenteCorso
5. ControllerPresidenteScuola
6. ControllerResponsabile
7. ControllerStudente

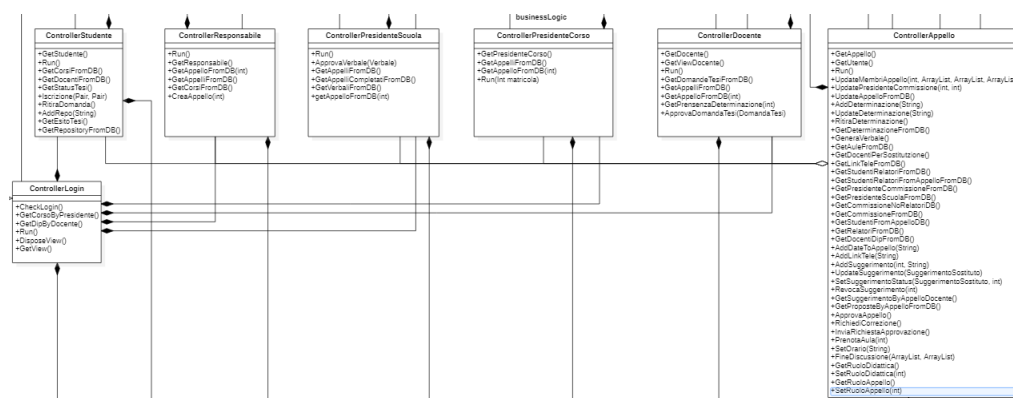


Figura 2.4: Classi Controllers

Questi controllers si occuperanno rispettivamente sulla gestione degli appelli, gestione degli appelli docente, sul login, gestione interfaccia per il presidente del corso, gestione interfaccia per presidente della scuola, gestione interfaccia per il responsabile e per lo studente.

## 2.4 User interface

L'utente interface è la parte nella quale l'utente finale interagisce con l'applicazione. L'utente, cliccando i bottoni sull'interfaccia grafica riesce a completare l'operazione desiderata. Quando l'utente cliccherà un bottone, l'applicazione produrrà un evento dove il controller lo riceverà e lo gestirà. Il controller, traducendo l'intenzione dell'utente, invierà dei comandi al model e view per raggiungere l'obiettivo che l'utente vorrebbe ottenere. I views sono:

1. ViewAppello
2. ViewDocente
3. ViewLogin
4. ViewPresidenteCorso
5. ViewPresidenteScuola
6. ViewResponsabile
7. ViewStudente



Figura 2.5: `UIInterface`

che sono rispettivamente una interfaccia per l'appello, interfaccia per il docente, interfaccia per il login, interfaccia per il presidente del corso, interfaccia per il presidente della scuola, interfaccia per il responsabile e infine interfaccia per lo studente.

# Capitolo 3

## Testing

Per verificare il corretto funzionamento dell'applicazione abbiamo creato dei tests con JUnit 5.

### 3.1 Integration Test

#### 3.1.1 Publication Cycle Test

Questo test è di tipo Gray-Box, ha lo scopo di verifica che il ciclo pubblicazione-approvazione appello funzioni correttamente. Quindi il responsabile pubblica l'appello, gli studenti si iscrivono, i relatori confermano la domanda e lo approvano, il presidente del corso identifica gli studenti laureandi-relatori e docenti, quindi una commissione, il responsabile prenota l'aula con il link di teleconferenza e infine il presidente della scuola approva. E questi tests (@test) sono eseguiti sequenzialmente grazie al @Order di JUnit 5.

```
1 public class PublicationCycleTest {  
2  
3  
4     @Test  
5     @DisplayName("Test pubblica appello")
```

```

6      @Order(1)
7      public void pubblicaAppelloTest() {
8          ControllerResponsabile controller = new
9              ↳ ControllerResponsabile("10002", "Gianni", "Manfrin");
10         assertTrue(controller.creaAppello(20002), () -> "Ok");
11     }
12
13     @Test
14     @DisplayName("Test iscrizione appello")
15     @Order(2)
16     public void iscrizioneStudenteTest() {
17         ControllerStudente controller = new ControllerStudente("10000", "Tizio", "Caio");
18         assertTrue(controller.iscrizione(Pair.of(20002, "Informatica"), Pair.of(10017,
19             ↳ "Alvise Pagnotto")), () -> "Ok");
20         assertTrue(controller.addRepo("test"), () -> "Ok");
21     }
22
23     @Test
24     @DisplayName("Test approvazione appello")
25     @Order(3)
26     public void approvazioneAppelloTest() {
27         ControllerDocente controller = new
28             ↳ ControllerDocente("10017", "Alvise", "Pagnotto", Pair.of(40002, "Dipartimento di
29             ↳ Ingegneria dell'Informazione"));
30         ArrayList<DomandaTesi> domande = controller.getDomandeTesiFromDB();
31         DomandaTesi d = domande.get(domande.size()-1);
32         assertTrue(controller.approvaDomandaTesi(d, true), () -> "Ok");
33     }
34
35     @Test
36     @DisplayName("Test identificazione membri, assegnazione presidente e set orario")
37     @Order(4)
38     public void identificazioneMembriTest() {
39         ControllerPresidenteCorso controller = new
40             ↳ ControllerPresidenteCorso("10006", "Delfino", "Pirozzi", Pair.of(20002,
41             ↳ "Informatica"));
42
43         ArrayList<AppelloTesi> appelli = controller.getAppelliFromDB();
44         AppelloTesi appello = appelli.get(appelli.size()-1);
45
46         ControllerAppello ca = new ControllerAppello(appello, null,
47             ↳ Utils.getRuolo(controller.getPresidenteCorso()),
48             ↳ 6, controller.getPresidenteCorso());
49
50         assertTrue(ca.updateMembriAppello(appello.getId(), new
51             ↳ ArrayList<Integer>(Arrays.asList(10000)),
52             ↳ new ArrayList<Integer>(Arrays.asList(10017)), new
53             ↳ ArrayList<Integer>(Arrays.asList(10018, 10019, 10020, 10021))) , () -> "Ok");
54
55         assertTrue(ca.updatePresidenteCommissione(appello.getId(), 10017), () -> "Ok");

```

```
48     assertTrue(ca.addDataToAppello("10/03/2022"), () -> "Ok");
49 }
50
51
52 @Test
53 @DisplayName("Test settaggio orario, aula appello e link teleconferenza")
54 @Order(5)
55 public void setOrarioAppelloTest() {
56     ControllerResponsabile controller = new
57         ↪ ControllerResponsabile("10002", "Gianni", "Manfrin");
58
59     ArrayList<AppelloTesi> appelli = controller.getAppelliFromDB();
60     AppelloTesi appello = appelli.get(appelli.size()-1);
61
62     ControllerAppello ca = new ControllerAppello(appello, null,
63         ↪ Utils.getRuolo(controller.getResponsabile()),
64         4, controller.getResponsabile());
65
66     assertTrue(ca.prenotaAula(50001) , () -> "Ok");
67     assertTrue(ca.setOrario("13:00") , () -> "Ok");
68     assertTrue(ca.addLinkTele("test.com") , () -> "Ok");
69 }
70
71 @Test
72 @DisplayName("Test approvazione dell'appello")
73 @Order(6)
74 public void approvaAppelloTest() {
75     ControllerPresidenteScuola controller = new
76         ↪ ControllerPresidenteScuola("10003", "Raniero", "Calabrese");
77
78     ArrayList<AppelloTesi> appelli = controller.getAppelliFromDB();
79     AppelloTesi appello = appelli.get(appelli.size()-1);
80
81     ControllerAppello ca = new ControllerAppello(appello, null,
82         ↪ Utils.getRuolo(controller.getPresidenteScuola()),
83         5, controller.getPresidenteScuola());
84
85     assertTrue(ca.approvaAppello() , () -> "Ok");
86 }
87 }
```



## 3.2 Unit Test

### 3.2.1 Login Test

Anche il login test per noi è molto importante. Perché siamo molto attenti riguardo alla sicurezza degli utenti, quindi vogliamo verificare se i login con matricole sbagliate e password sbagliate non entrino nel sistema, invece quelle giuste riescano ad entrare nel sistema.

```
1  class LoginTests {
2      ControllerLogin login;
3
4      @BeforeEach
5      void setUp() {
6          login = new ControllerLogin();
7          login.disposeView();
8      }
9
10     @ParameterizedTest
11     @ValueSource(ints = {10000, 10001, 10002, 10003, 10004})
12     @DisplayName("Tests con credenziali corretti")
13     void testMultiply(int candidate) {
14         assertTrue(login.checkLogin(candidate + "", "123"), () -> "Ok");
15     }
16
17     @ParameterizedTest
18     @ValueSource(ints = {100000, 100010, 100020, 100030, 100040})
19     @DisplayName("Tests con credenziali errati")
20     void testMultiplyWithZero(int candidate) {
21         assertFalse(login.checkLogin(candidate + "", "123"), () -> "Ok");
22     }
23 }
```

# Capitolo 4

## Demo

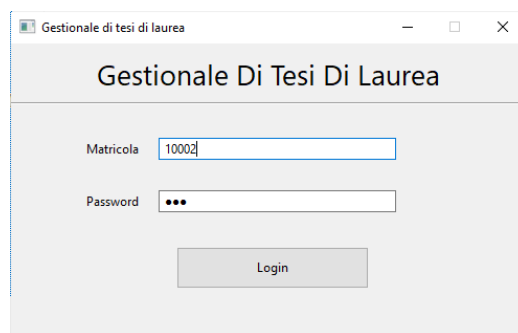


Figura 4.1: Primo passo: Responsabile fa il login

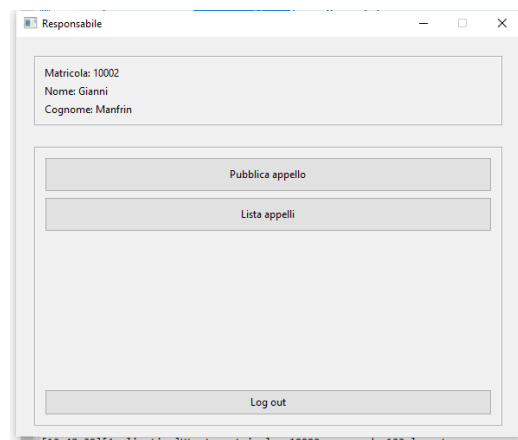


Figura 4.2: Primo passo: Responsabile entra nella sua interfaccia

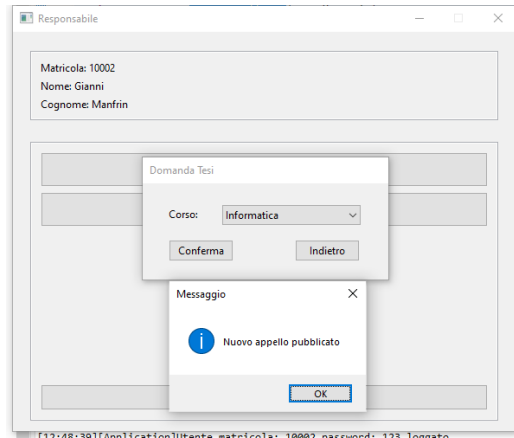


Figura 4.3: Primo passo: il responsabile clicca pubblica appello, sceglie il corso e clicca conferma

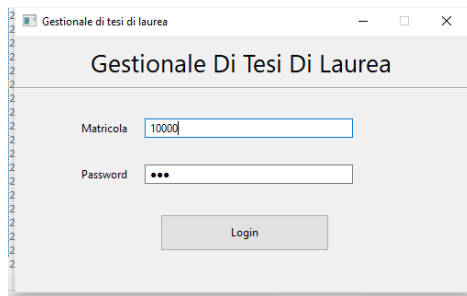


Figura 4.4: Secondo passo: Lo studente fa il login

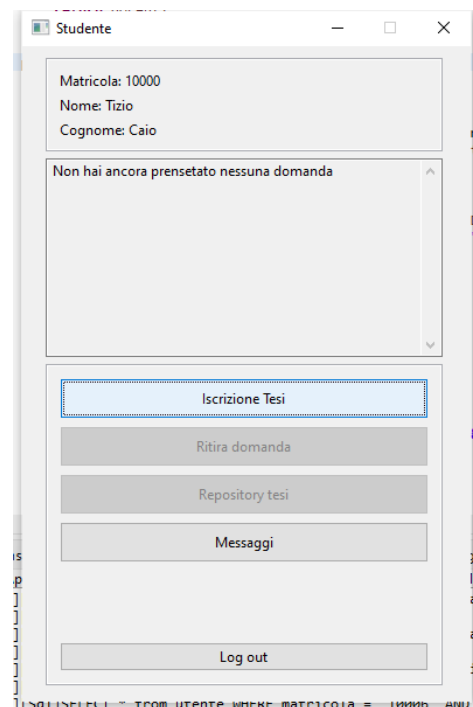


Figura 4.5: Secondo passo: Lo studente clicca iscrizione tesi

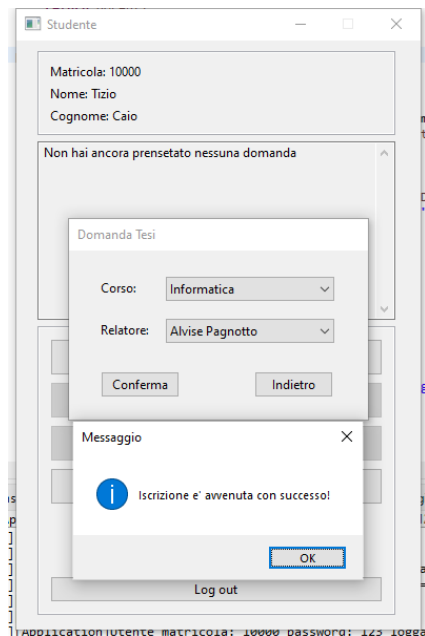


Figura 4.6: Secondo passo: Lo studente sceglie il corso, il proprio relatore e clicca conferma

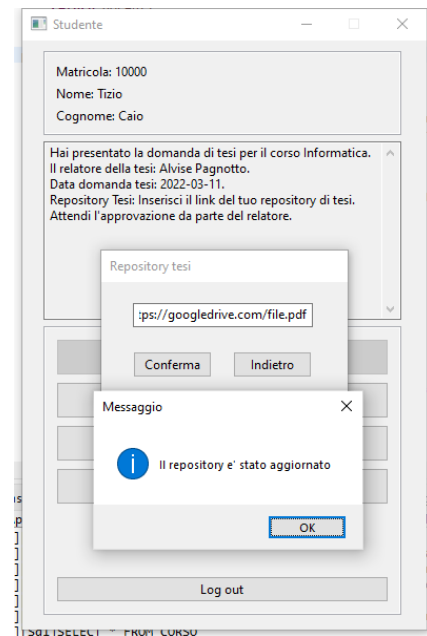


Figura 4.7: Secondo passo: Lo studente clicca sul bottone Repository Tesi e aggiunge il link per il materiale (tesi, progetto)

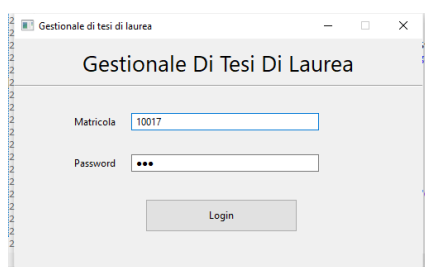


Figura 4.8: Terzo passo: Il relatore entra nella sua interfaccia

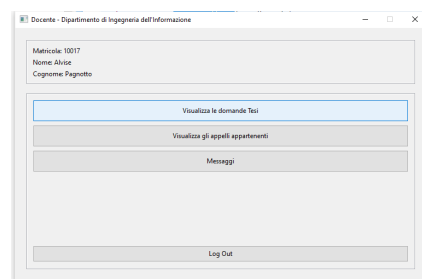


Figura 4.9: Terzo passo: Il relatore clicca sul bottone visualizza domande di tesi

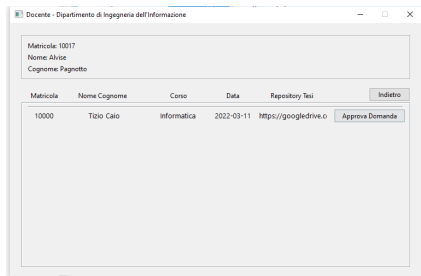


Figura 4.10: Terzo passo: Il relatore sceglie il proprio studente

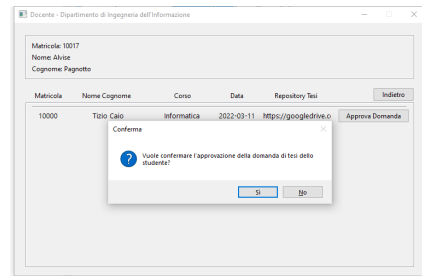


Figura 4.11: Terzo passo: Il relatore clicca approva domanda

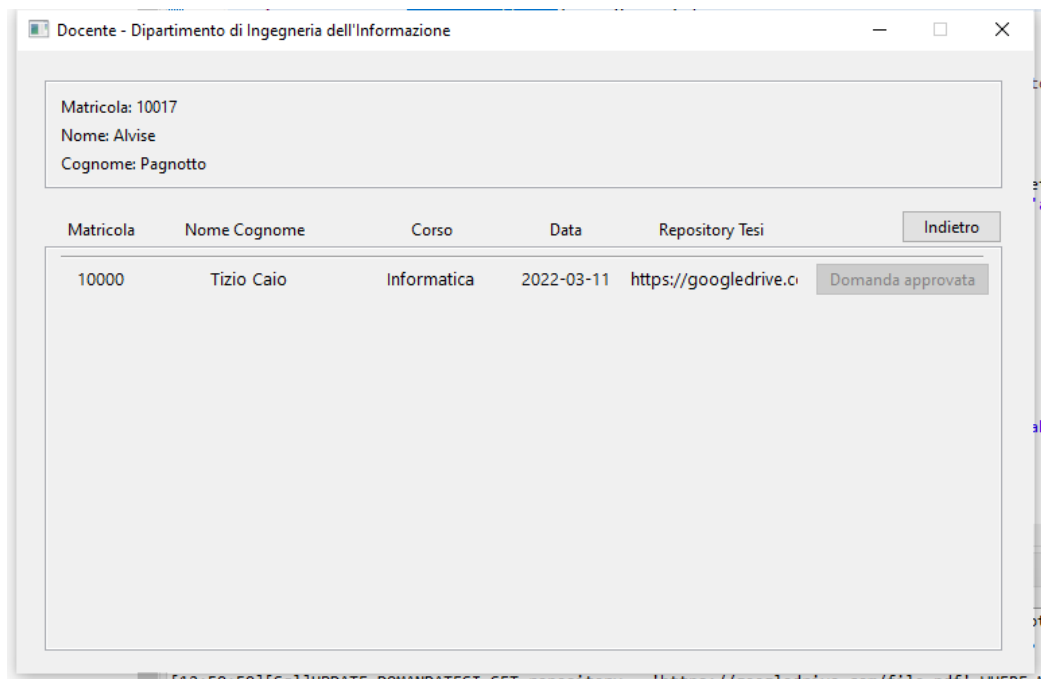


Figura 4.12: Terzo passo: a questo punto la domanda dello studente è stato approvato

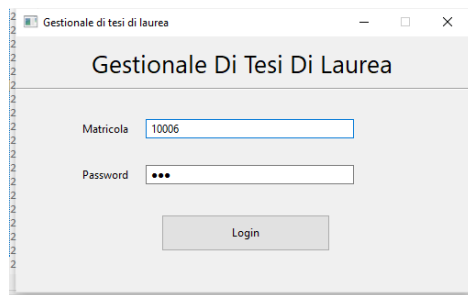


Figura 4.13: Quarto passo: Il presidente del corso fa il login con le proprie credenziali

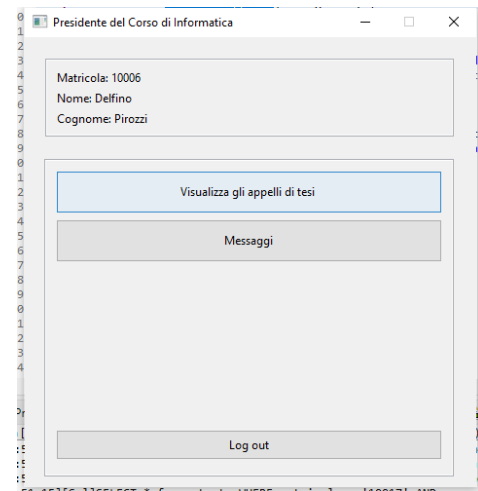


Figura 4.14: Quarto passo: Il presidente del corso clicca il bottone visualizza gli appelli di tesi

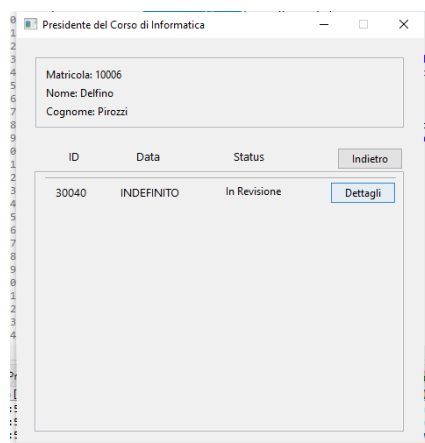


Figura 4.15: Quarto passo: Il presidente del corso sceglie l'appello e clicca il bottone su dettagli

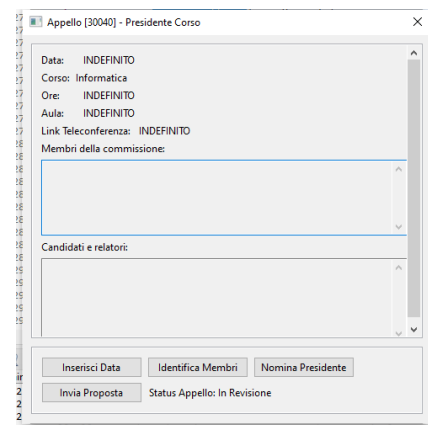


Figura 4.16: Quarto passo: Il presidente del corso a questo punto clicca sul bottone identifica membri

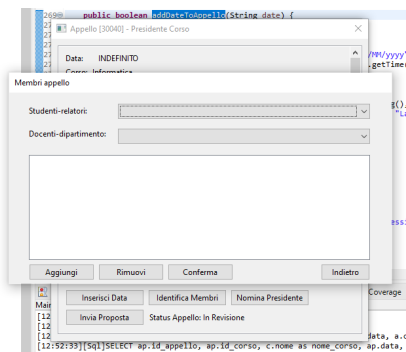


Figura 4.17: Quarto passo: Il presidente entra nell'interfaccia per aggiungere i membri

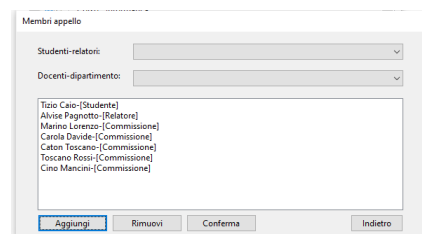


Figura 4.18: Quarto passo: Il presidente del corso aggiunge tutti gli studenti e docenti interessati

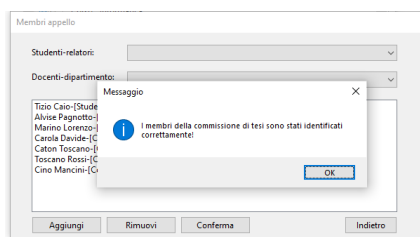


Figura 4.19: Quarto passo: Il presidente del corso clicca conferma

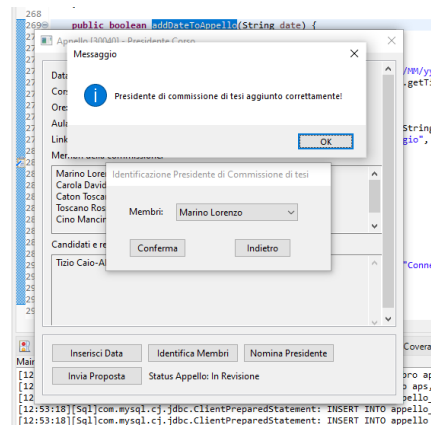


Figura 4.20: Quarto passo: Il presidente del corso clicca sul bottone nomina presidente, sceglie il membro a cui conferisce il ruolo e clicca conferma

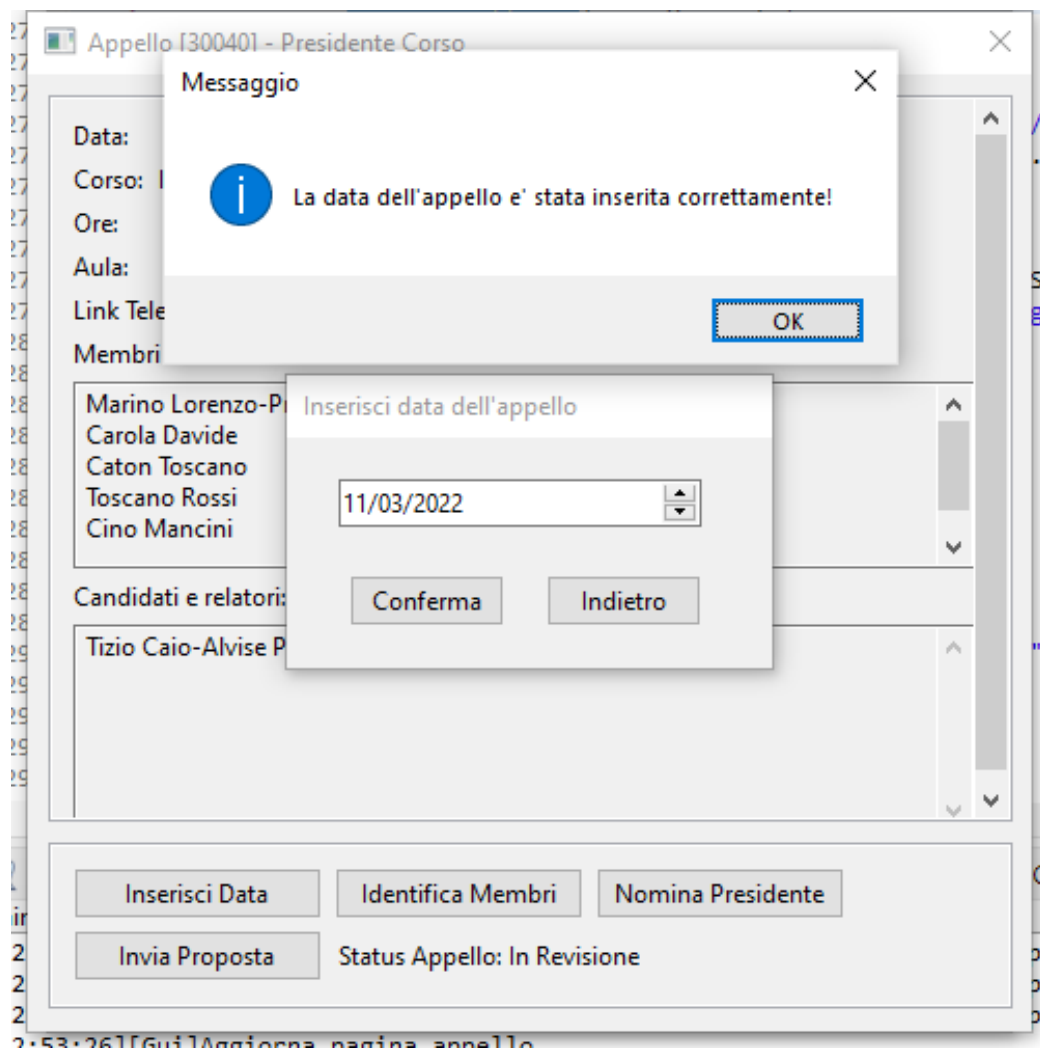


Figura 4.21: Quarto passo: Il presidente del corso sceglie la data dell'appello



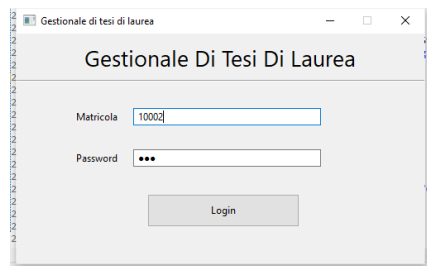


Figura 4.22: Quinto passo: Responsabile fa il login

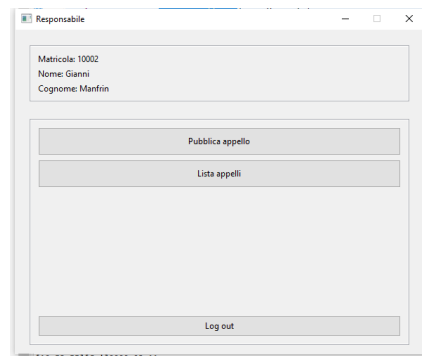


Figura 4.23: Quinto passo: Responsabile clicca lista appelli

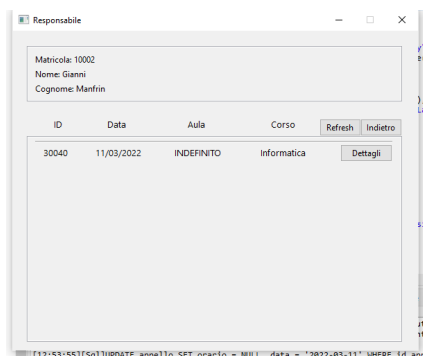


Figura 4.24: Quinto passo: Responsabile sceglie l'appello

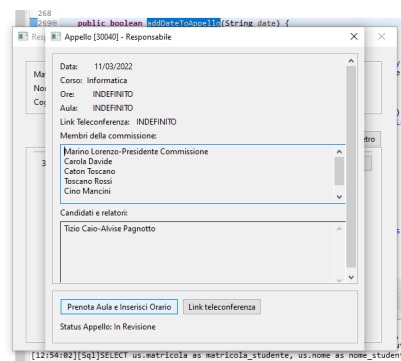


Figura 4.25: Quinto passo: Responsabile prenota l'aula e l'orario

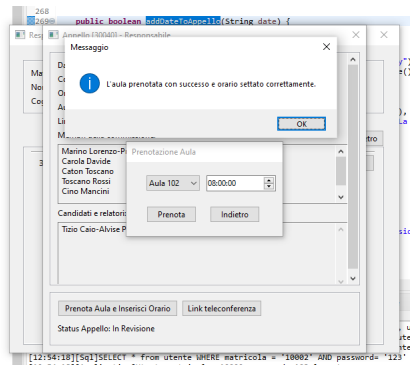


Figura 4.26: Quinto passo: Responsabile sceglie un'aula e un'orario e clicca conferma

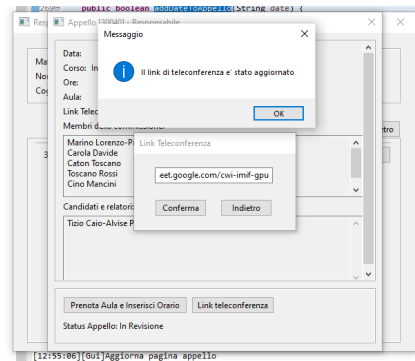


Figura 4.27: Quinto passo: Responsabile clicca sul bottone link teleconferenza, aggiunge un link e clicca conferma

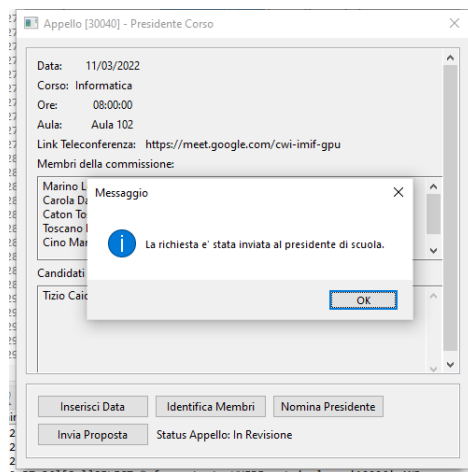


Figura 4.28: Sesto passo: Il presidente del corso clicca invia proposta

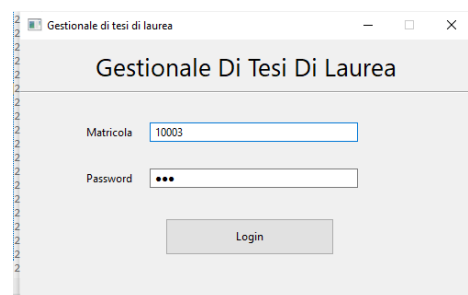


Figura 4.29: Settimo passo: Il presidente della scuola fa il login

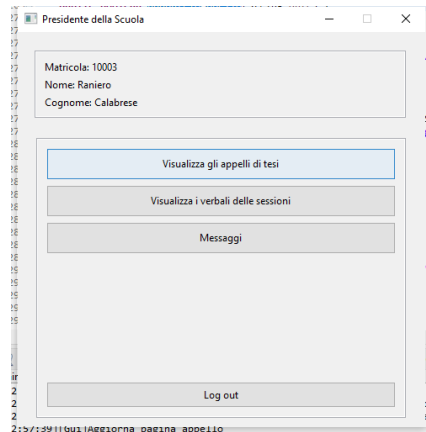


Figura 4.30: Settimo passo: Il presidente clicca visualizza gli appelli di tesi

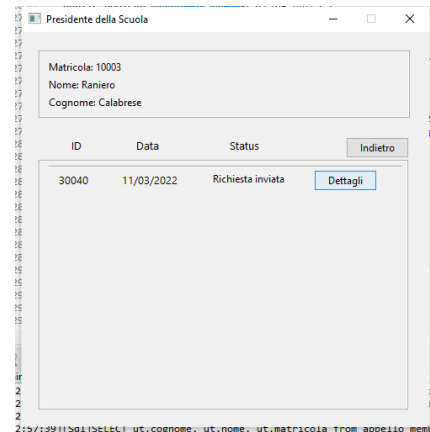


Figura 4.31: Settimo passo: Il presidente sceglie l'appello e clicca su dettagli

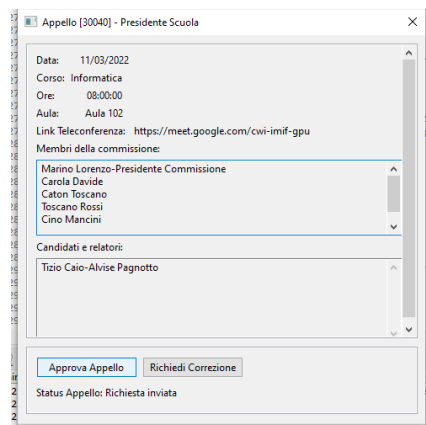


Figura 4.32: Settimo passo: Il presidente clicca su approva appello

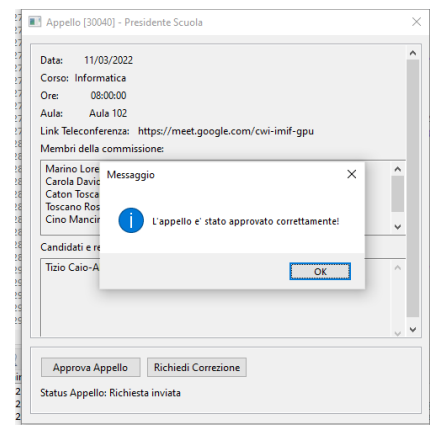


Figura 4.33: Settimo passo: Il presidente riceve conferma

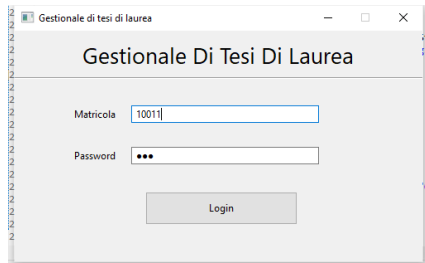


Figura 4.34: Ottavo passo(fase di discussione): Il presidente di commissione fa il login

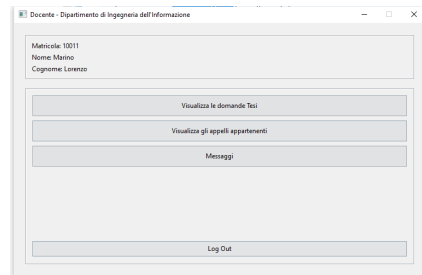


Figura 4.35: Ottavo passo: Il presidente clicca visualizza gli appelli appartenenti

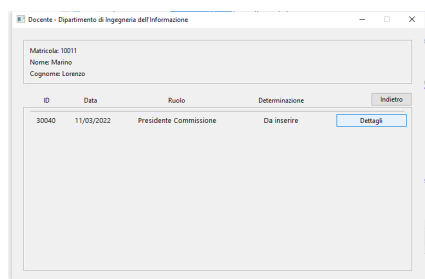


Figura 4.36: Ottavo passo: Il presidente sceglie l'appello e clicca su dettagli

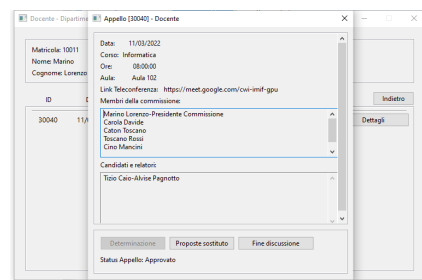


Figura 4.37: Ottavo passo: Il presidente clicca fine discussione

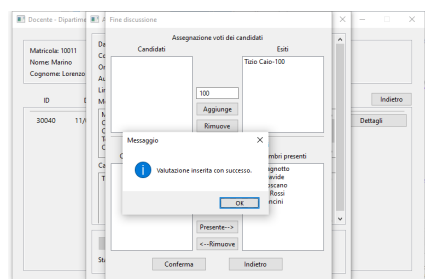


Figura 4.38: Ottavo passo: Il presidente aggiunge i docenti presenti e aggiunge i candidati assegnando un voto

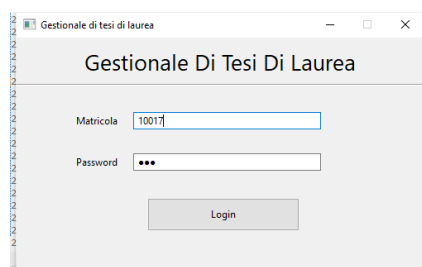


Figura 4.39: Nono passo: i docenti fanno il login

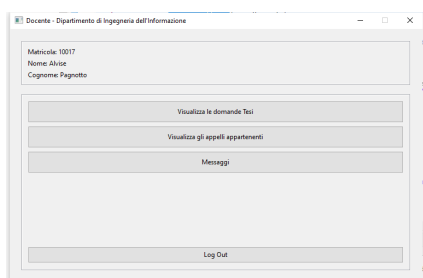


Figura 4.40: Nono passo: il docente clicca su visualizza gli appelli appartenenti

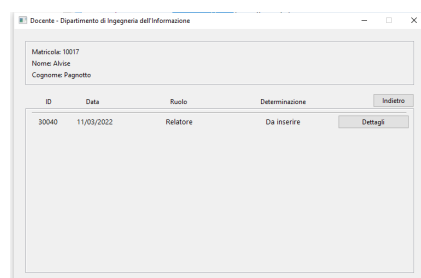


Figura 4.41: Nono passo: il docente sceglie l'appello e clicca su dettagli

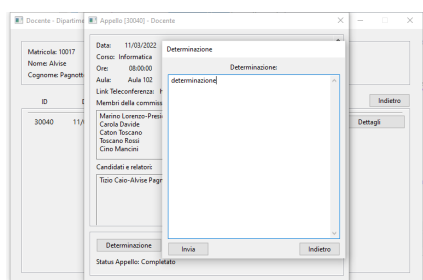


Figura 4.42: Nono passo: Il docente clicca sul bottone determinazione, scrive la sua determinazione e clicca invia

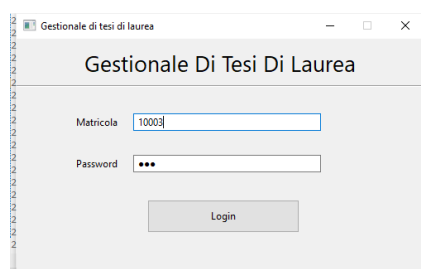


Figura 4.43: Decimo passo: Il presidente della scuola fa il login

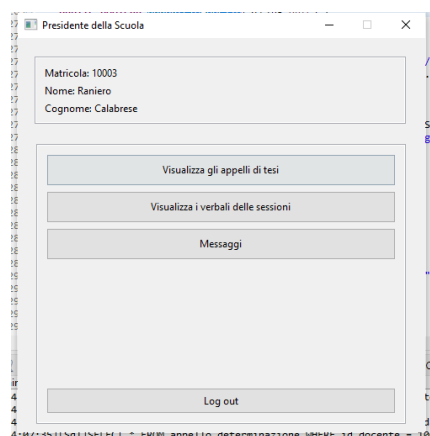


Figura 4.44: Decimo passo: Il presidente della scuola clicca sul bottone visualizza i verbali delle sessioni

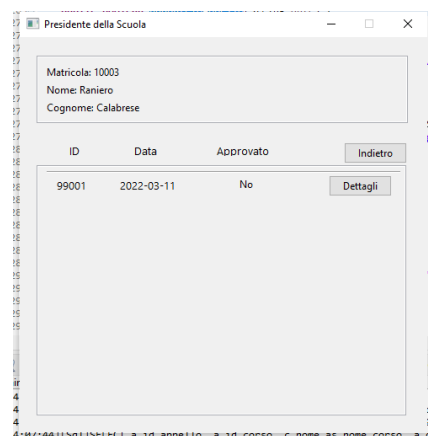


Figura 4.45: Decimo passo: Il presidente della scuola sceglie il verbale e clicca su dettagli

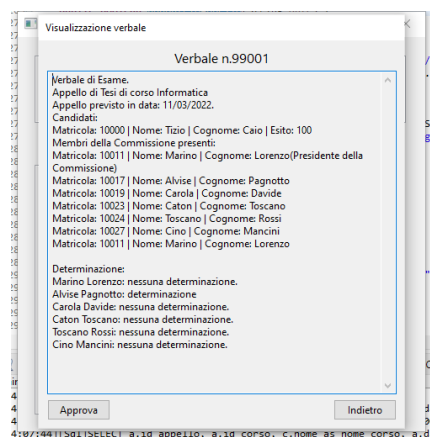


Figura 4.46: Decimo passo: Il presidente della scuola controlla il contenuto e poi clicca sul bottone approva

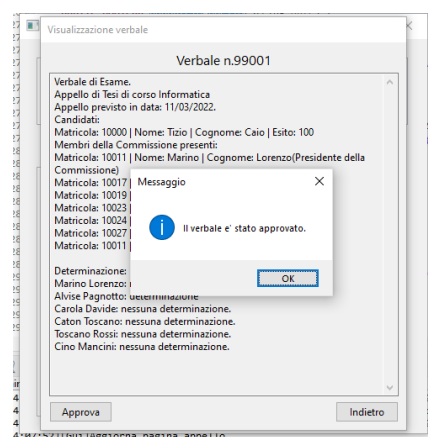


Figura 4.47: Decimo passo: Il presidente riceve conferma dell'approvazione

# Bibliografia

- [1] Eclipse. Swt: The standard widget toolkit. <https://www.eclipse.org/swt/>.