ByteBusters

- Davide De Vittorio
- Riccardo Rota Mino
- Migs Matthew Acar
- Houssem Ben Turkia

UniBook

Progetto I24 - Programmazione a oggetti e ingegneria del software

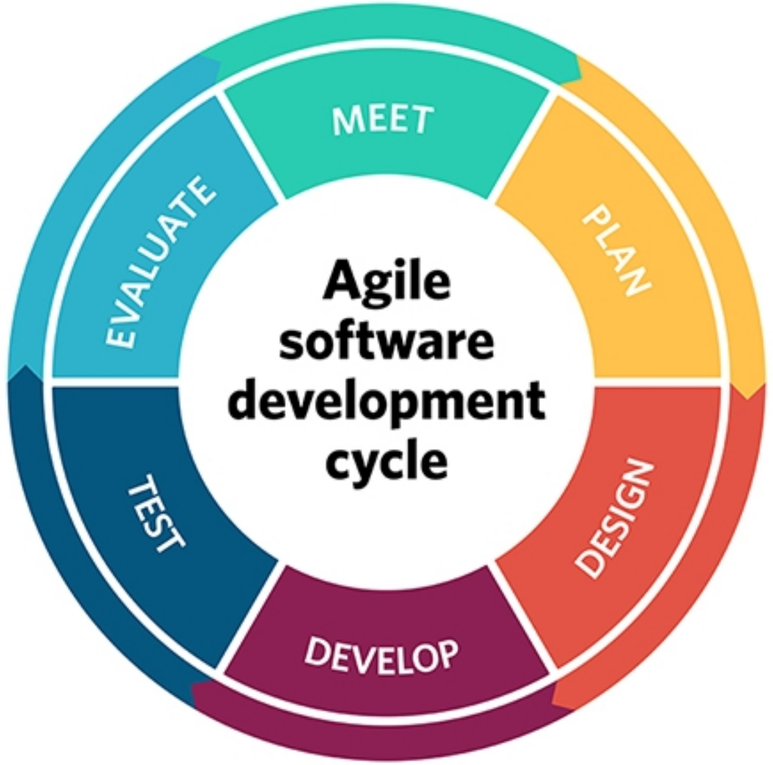
Università degli studi di Pavia - A.A. 2023/2024

Introduzione

Sistema informatico universitario che promuove la collaborazione, semplifica la gestione delle risorse accademiche e arricchisce l'esperienza universitaria degli utenti. In particolare permette di:

- Prenotare risorse
- Affittare Risorse
- Aggiungere risorse
- Scambiare messaggi
- Condividere file

Processo Software: Agile



Meet	Raccolta dei requisiti principali
Plan	Analisi dei requisiti principali e studio di fattibilità
Design	Analisi della maggior parte dei casi d'uso, produzione documentazione
Develop	Sviluppo dei componenti e implementazione delle funzionalità richieste
Test	Test e collaudo del sistema
Evalutate	Rilascio e raccolta feedback

Requisiti funzionali

- Registrazione
- Login/Logout
- Prenotazione risorse
- Cancellazione prenotazioni
- Affitto risorse
- Estensione periodo affitto
- Aggiunta risorse
- Rimozione risorse
- Scambio di messaggi
- Upload file
- Download file

Requisiti non funzionali

Esterni

- Utilizzo di dati personali dell'utente
- Utilizzo di database MySQL
- Interazione con sistema bancario esterno

Organizzativi

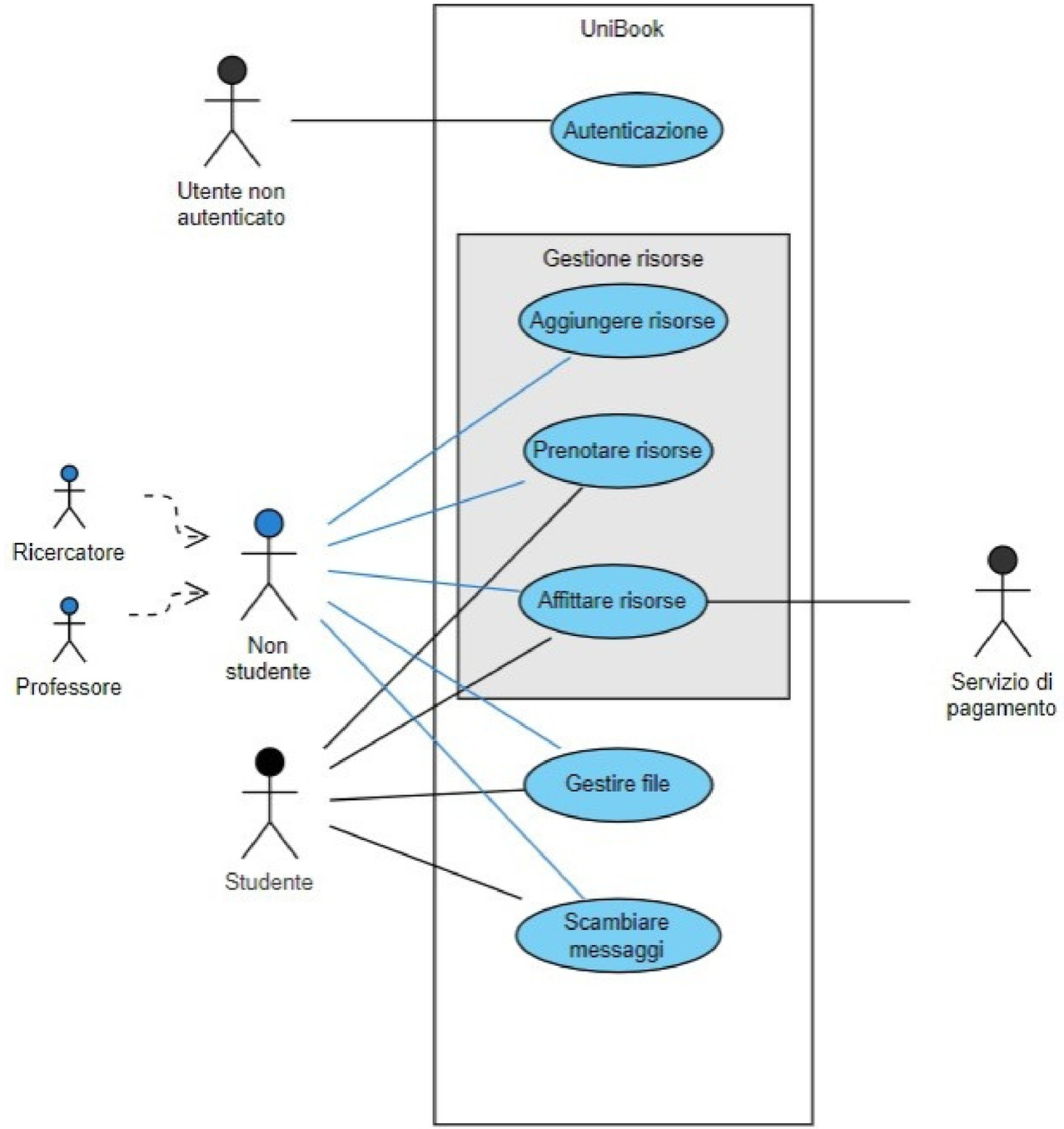
- Termini per la consegna del software: 01 marzo 2024
- Documentazione in lingua italiana

Tecnici

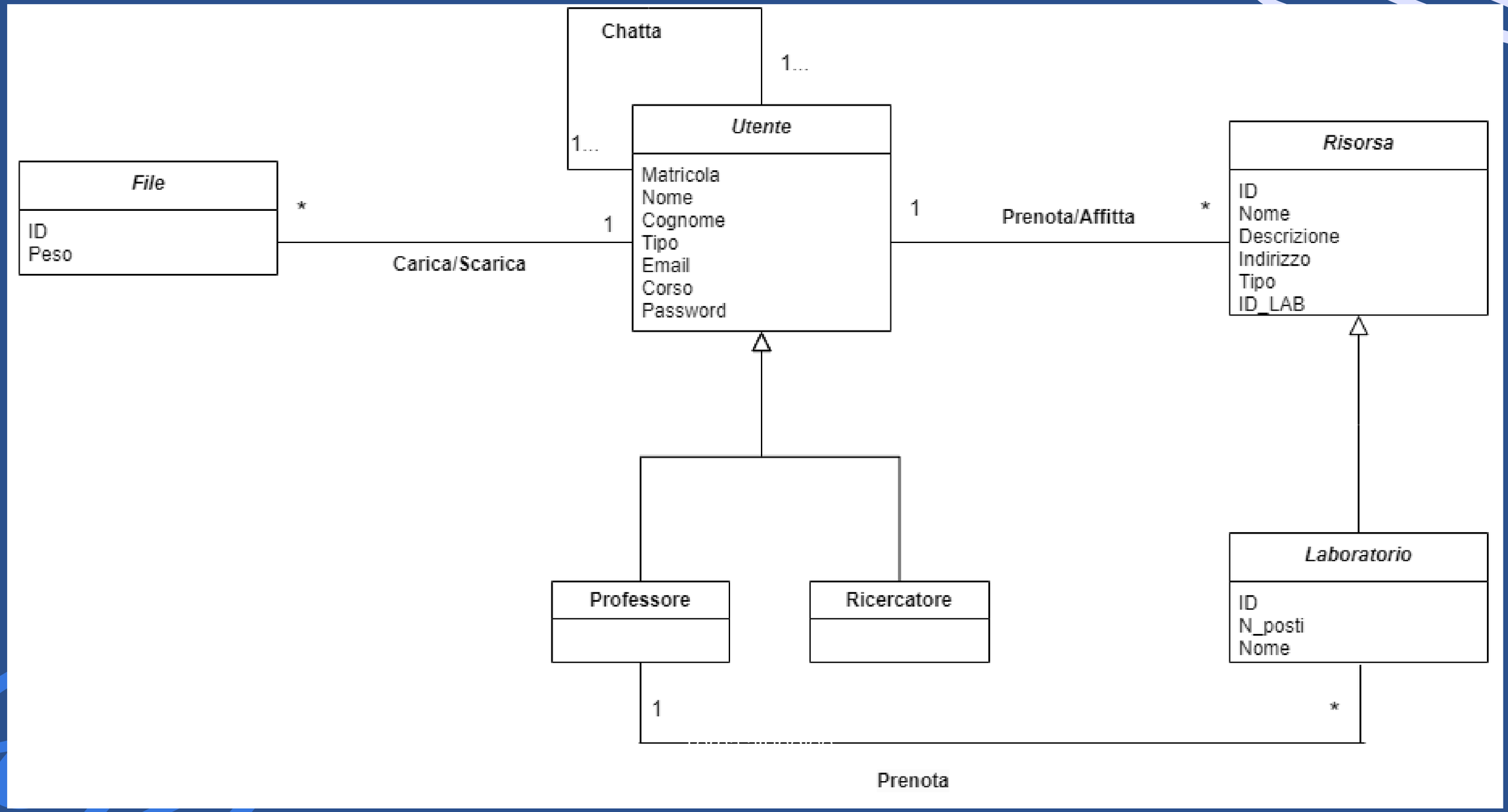
- Linguaggi: Java 8, SQL
- GUI: Swing

Casi d'uso

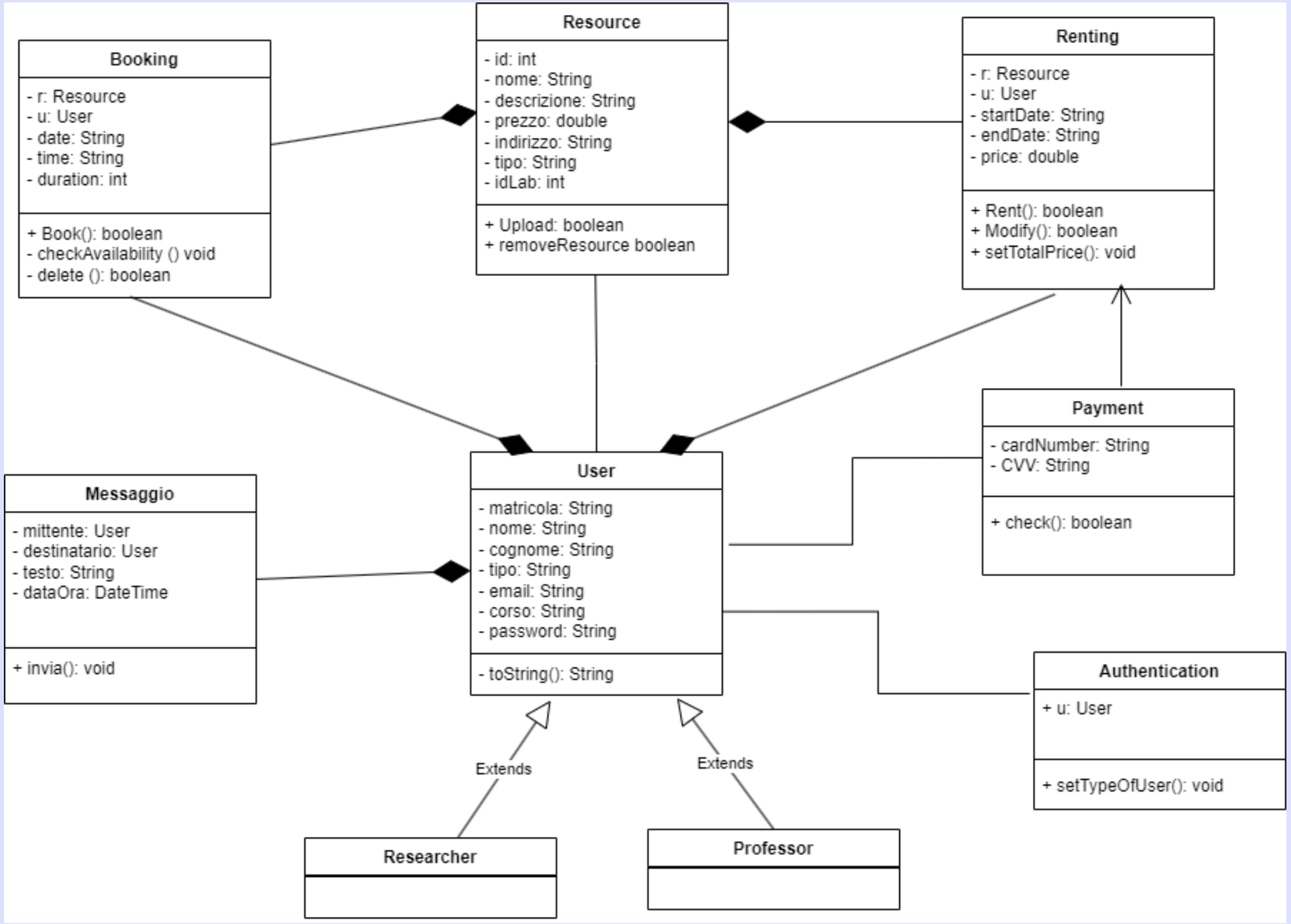
- **UC1: Prenotazione Risorse**
- **UC2: Affitto Risorse**
- **UC3: Gestione Risorse**
- **UC4: Servizio di messaggistica**
- **UC5: Condivisione file**



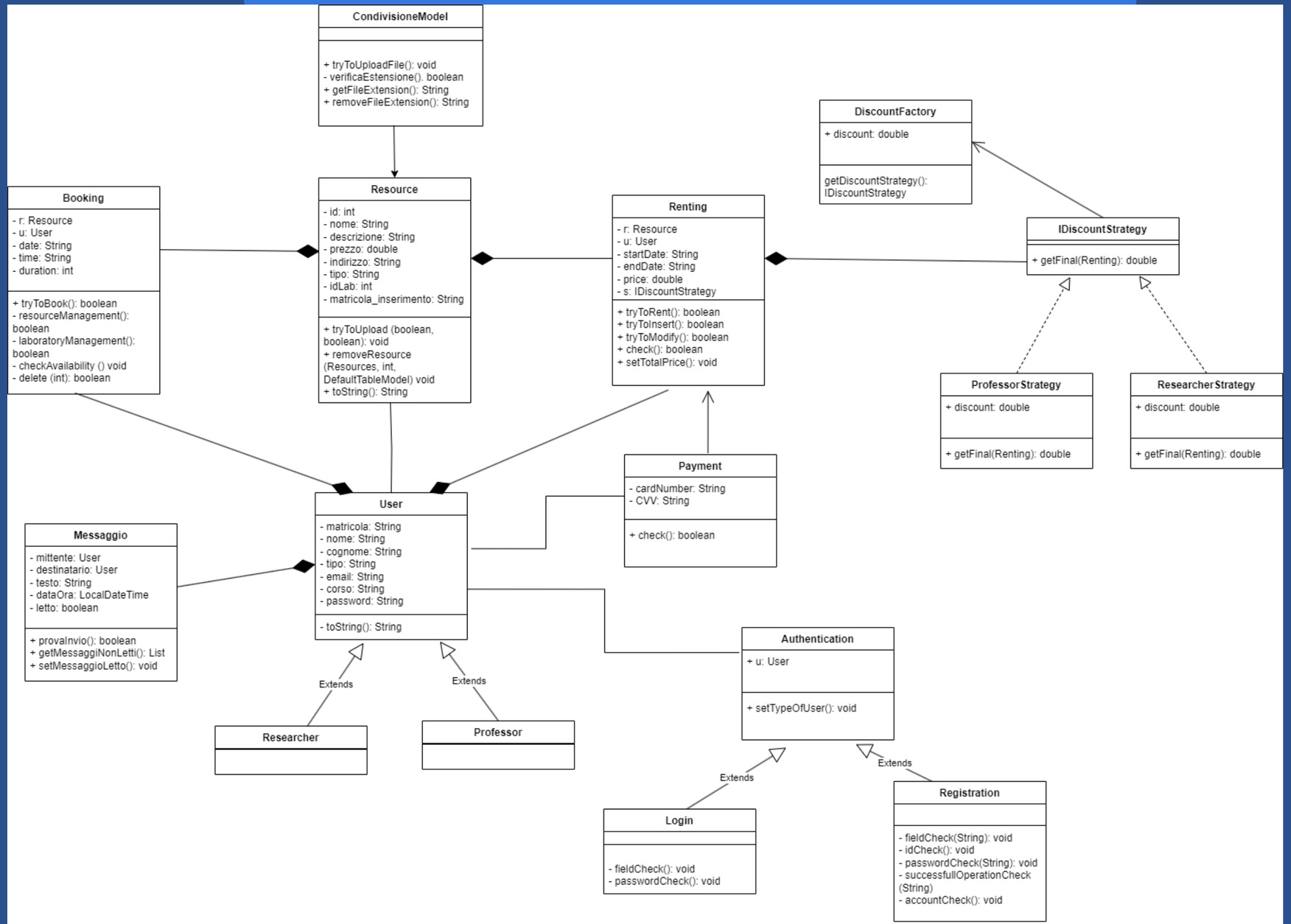
Modello di dominio



UML classi iniziale

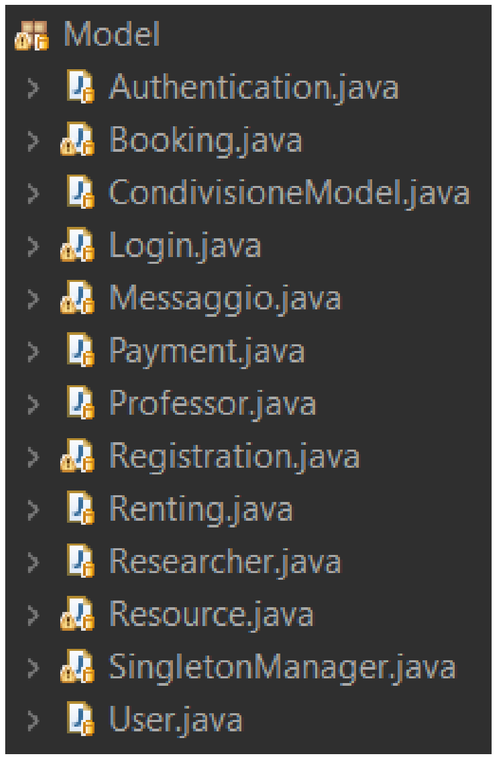


UML classi finale

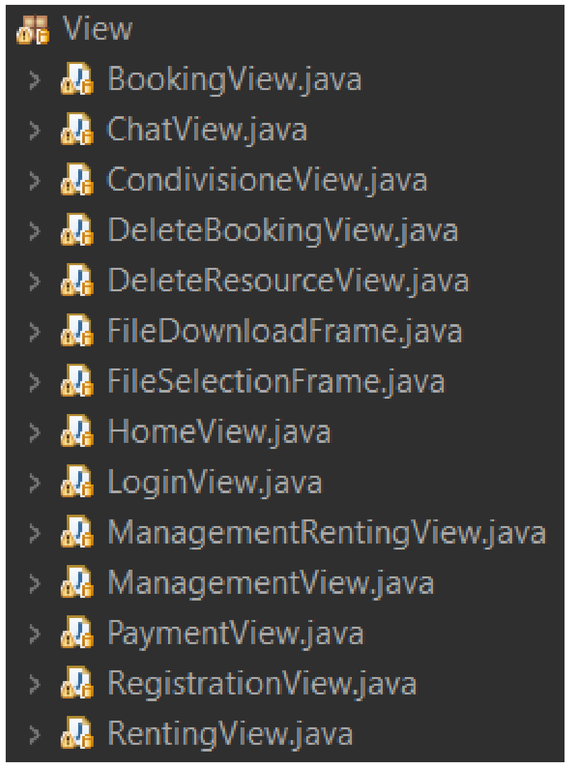


Struttura progetto

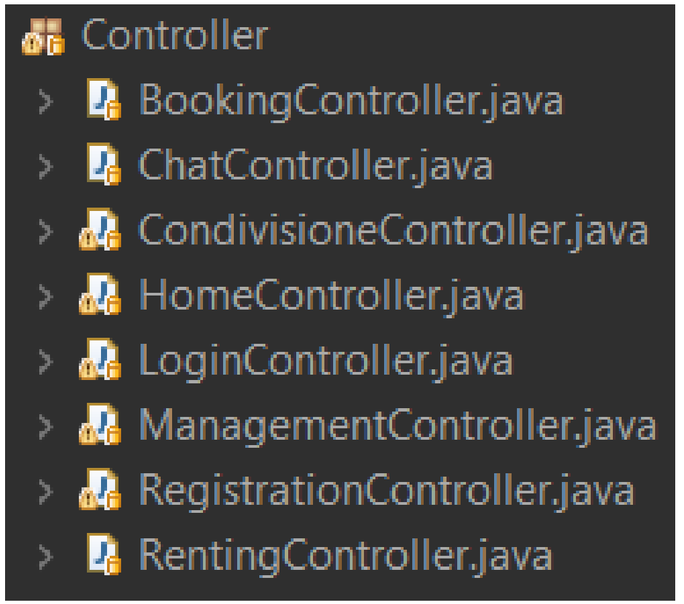
Model



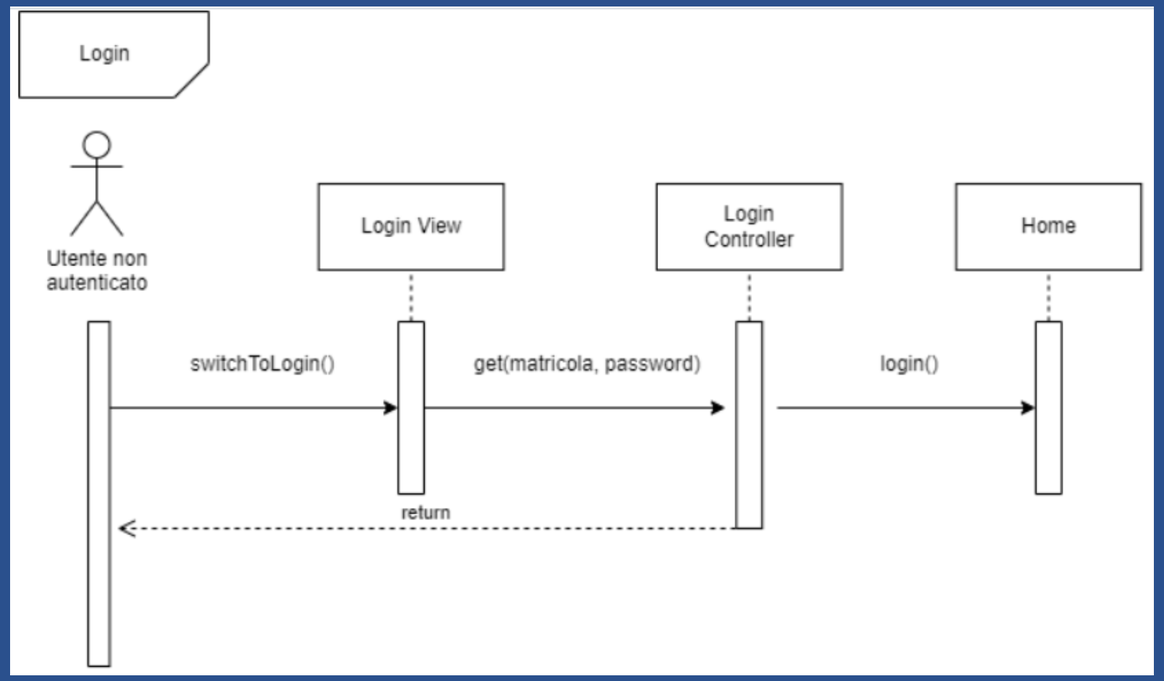
View

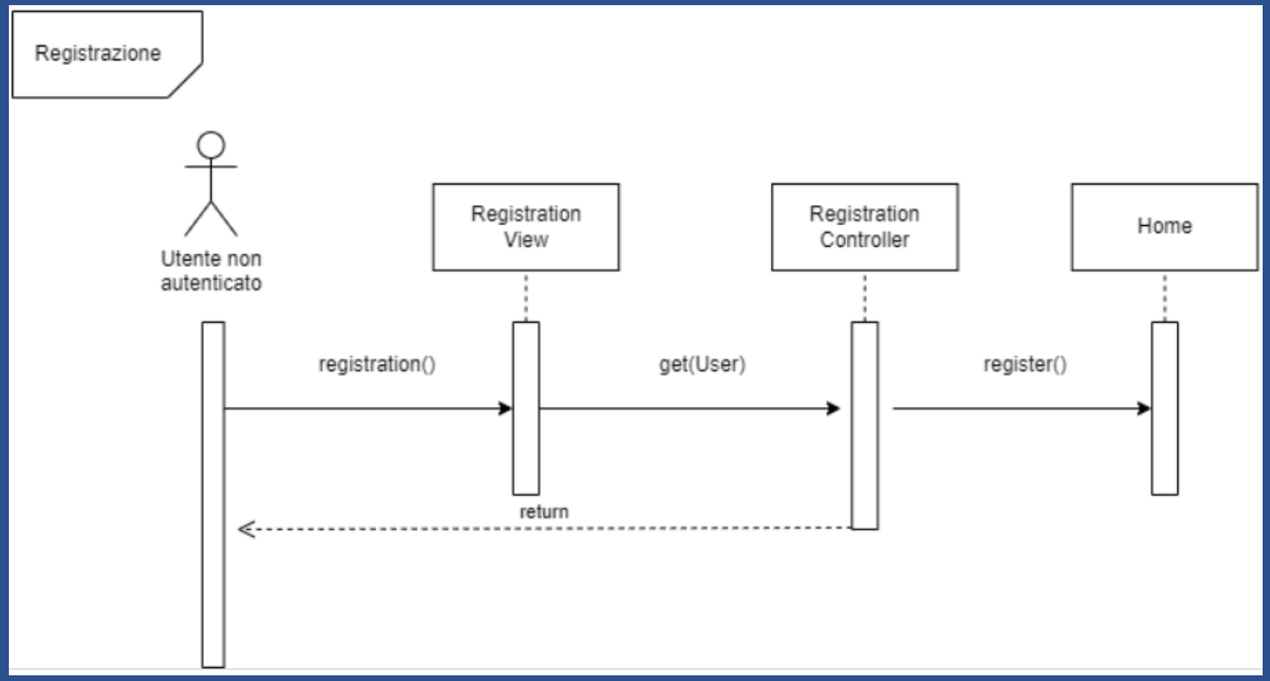


Controller

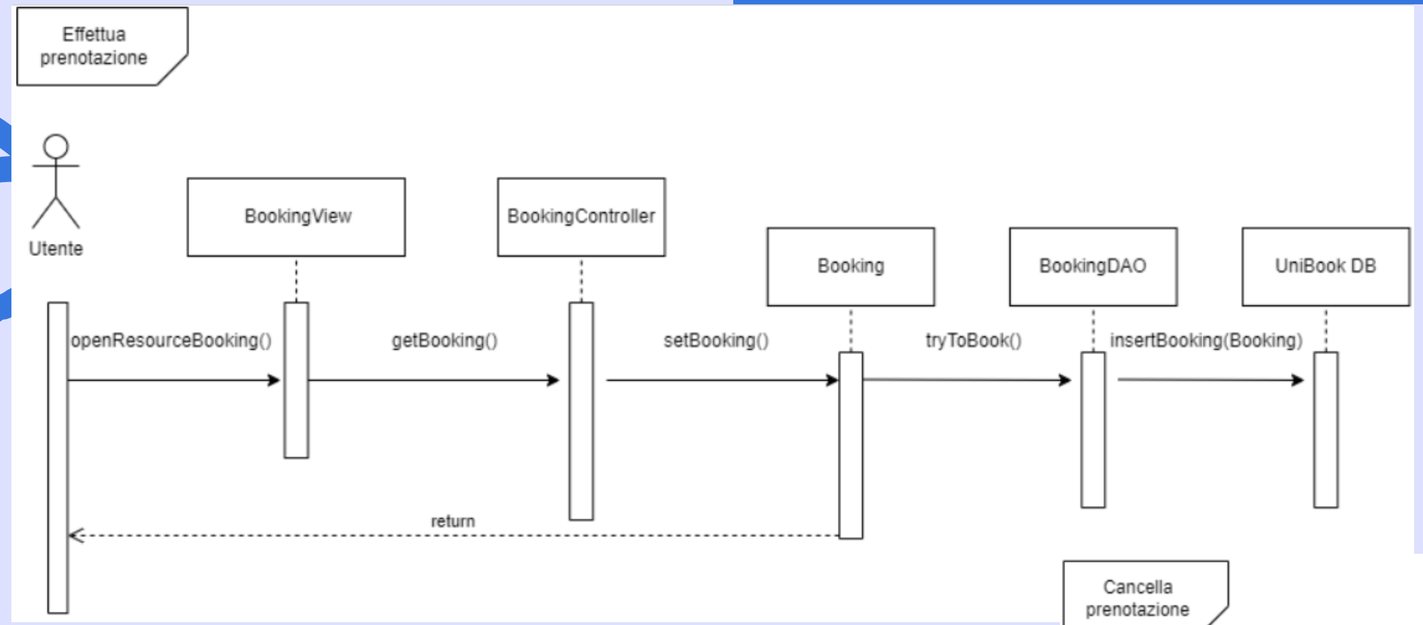


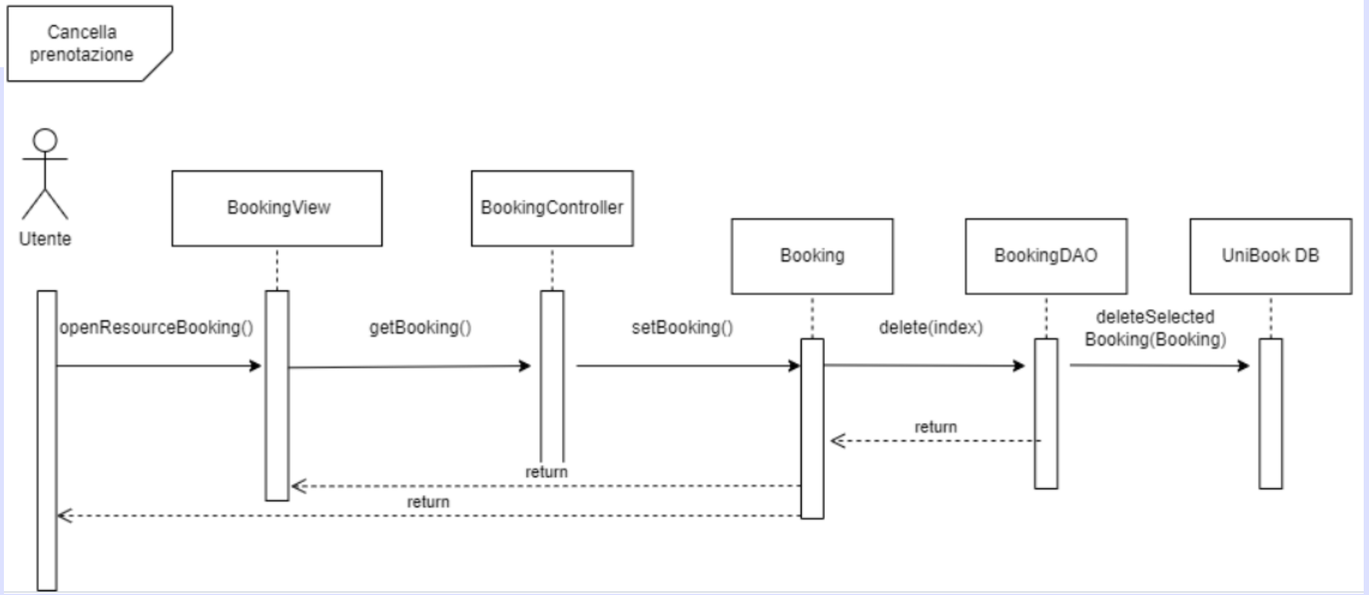
Diagrammi di sequenza autenticazione



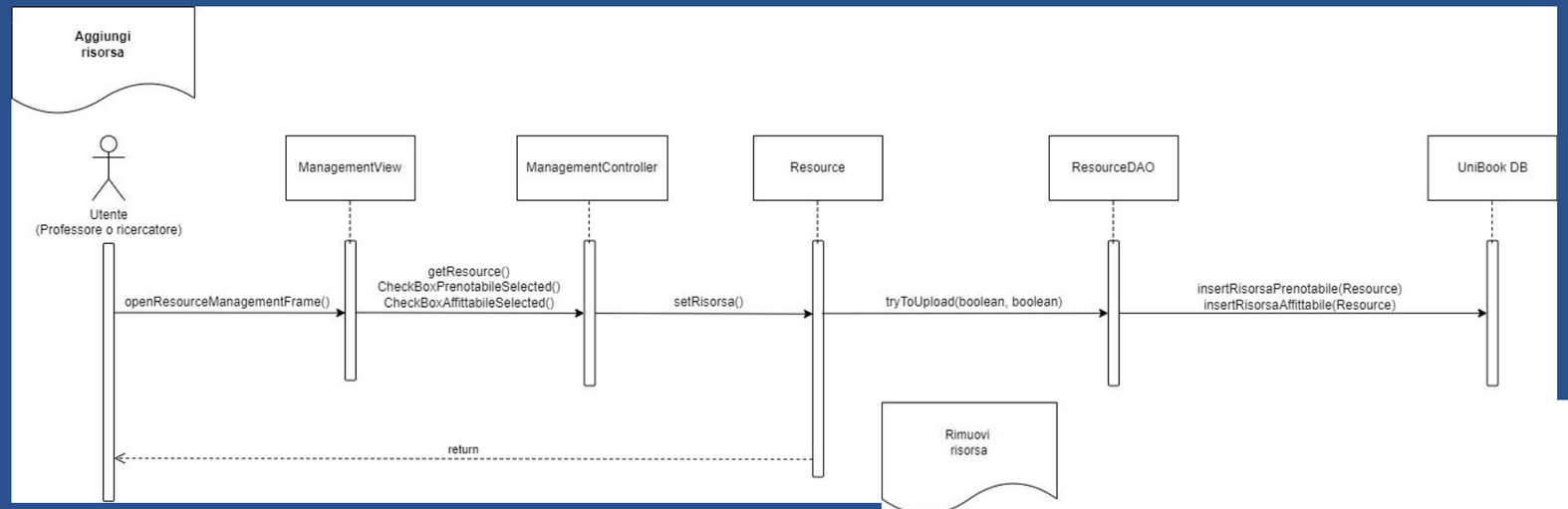


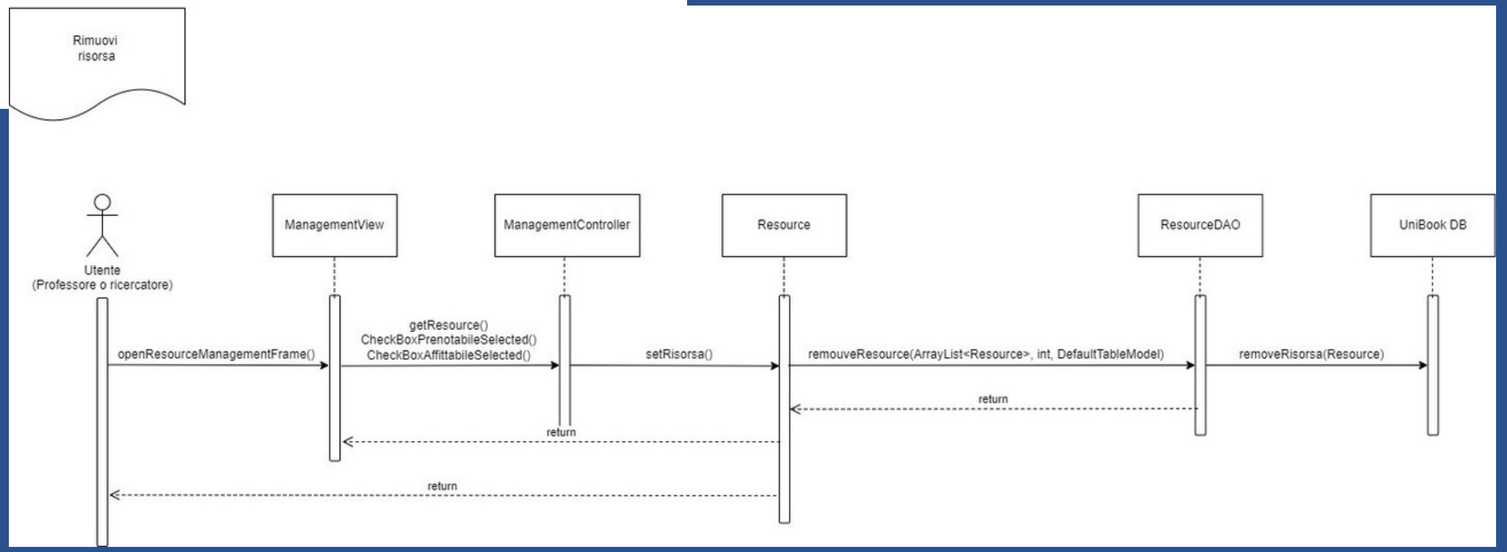
Diagrammi di sequenza prenotazione



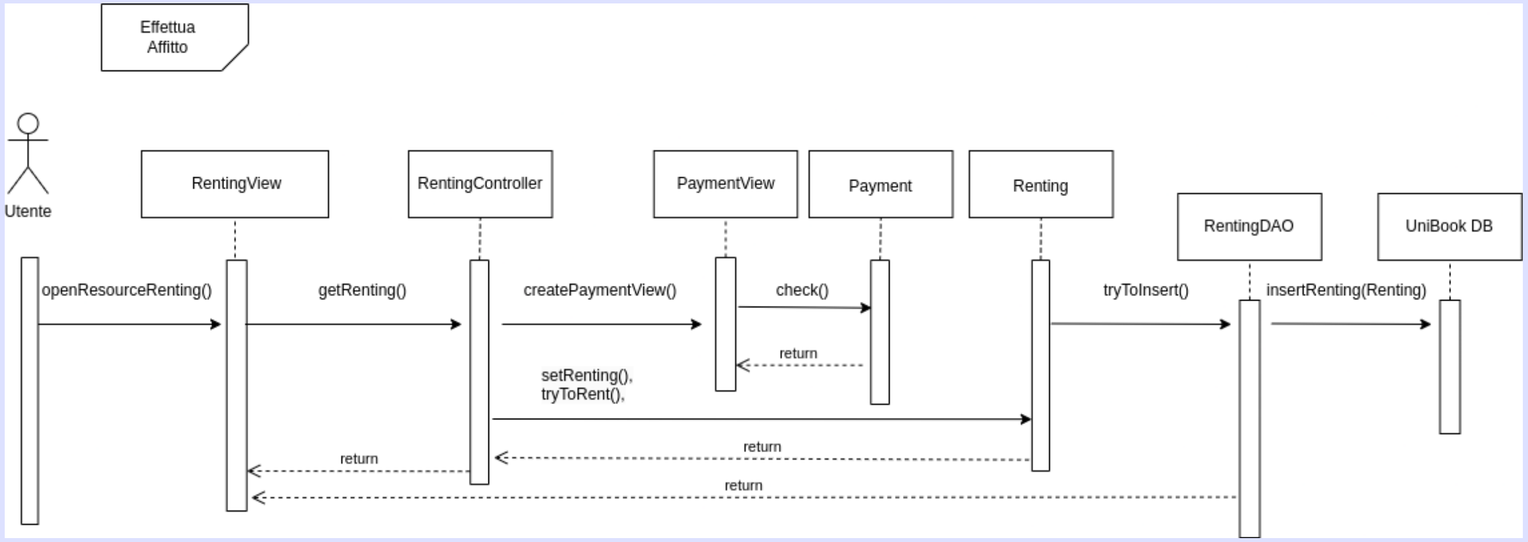


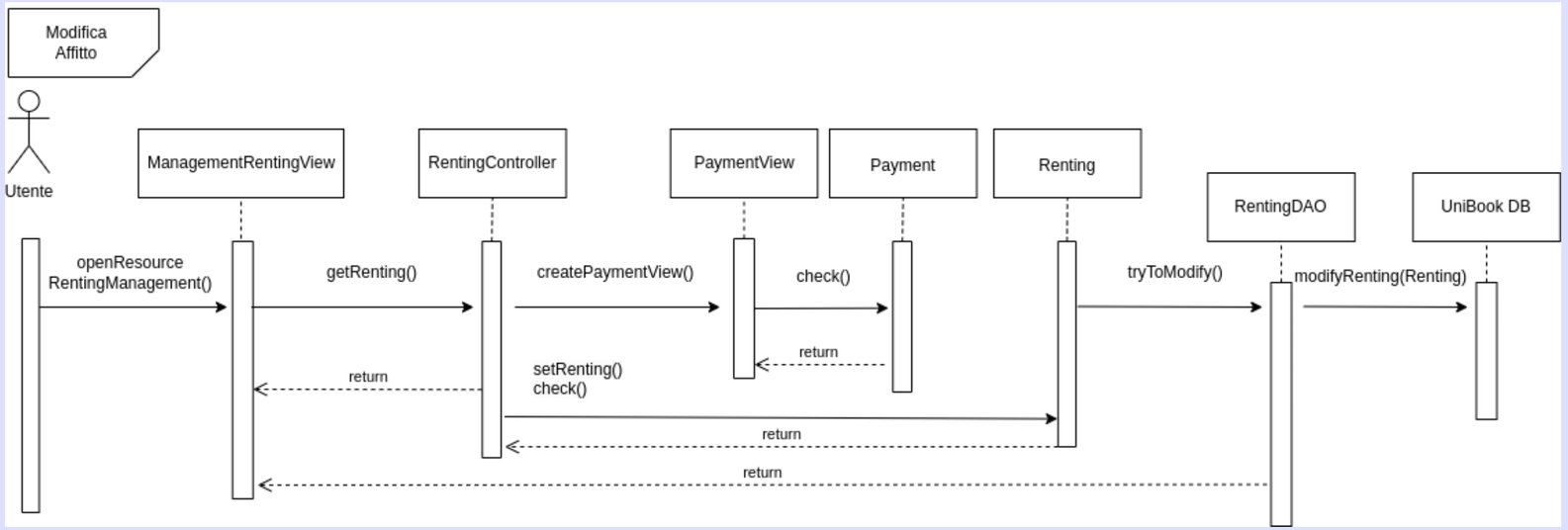
Diagrammi di sequenza gestione risorse



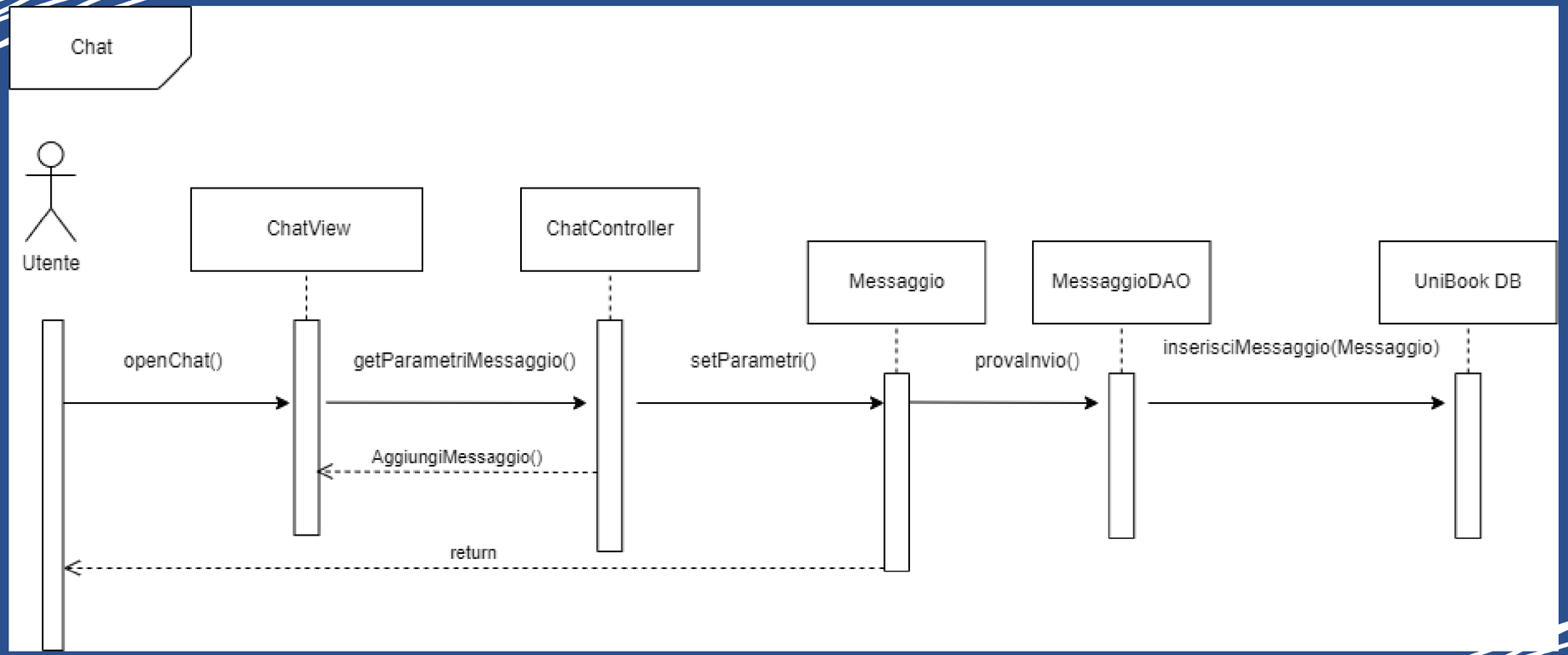


Diagrammi di sequenza affitto risorse

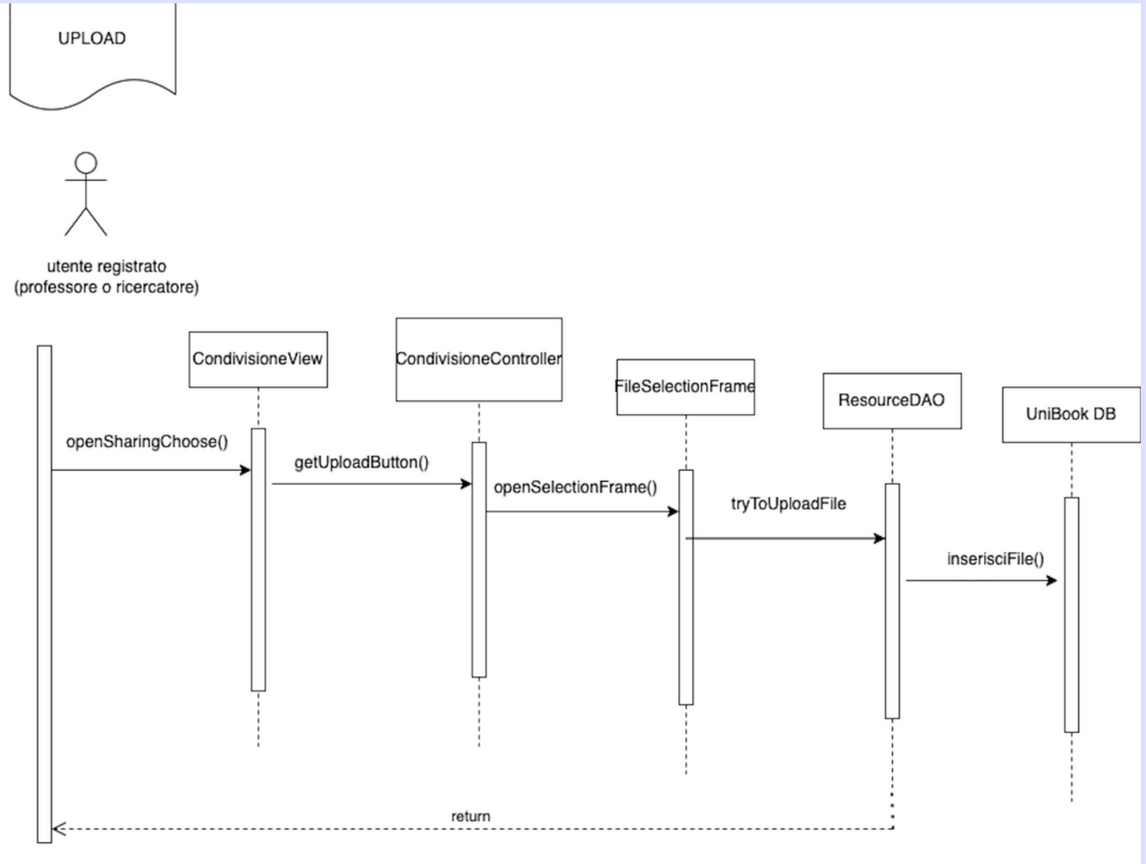


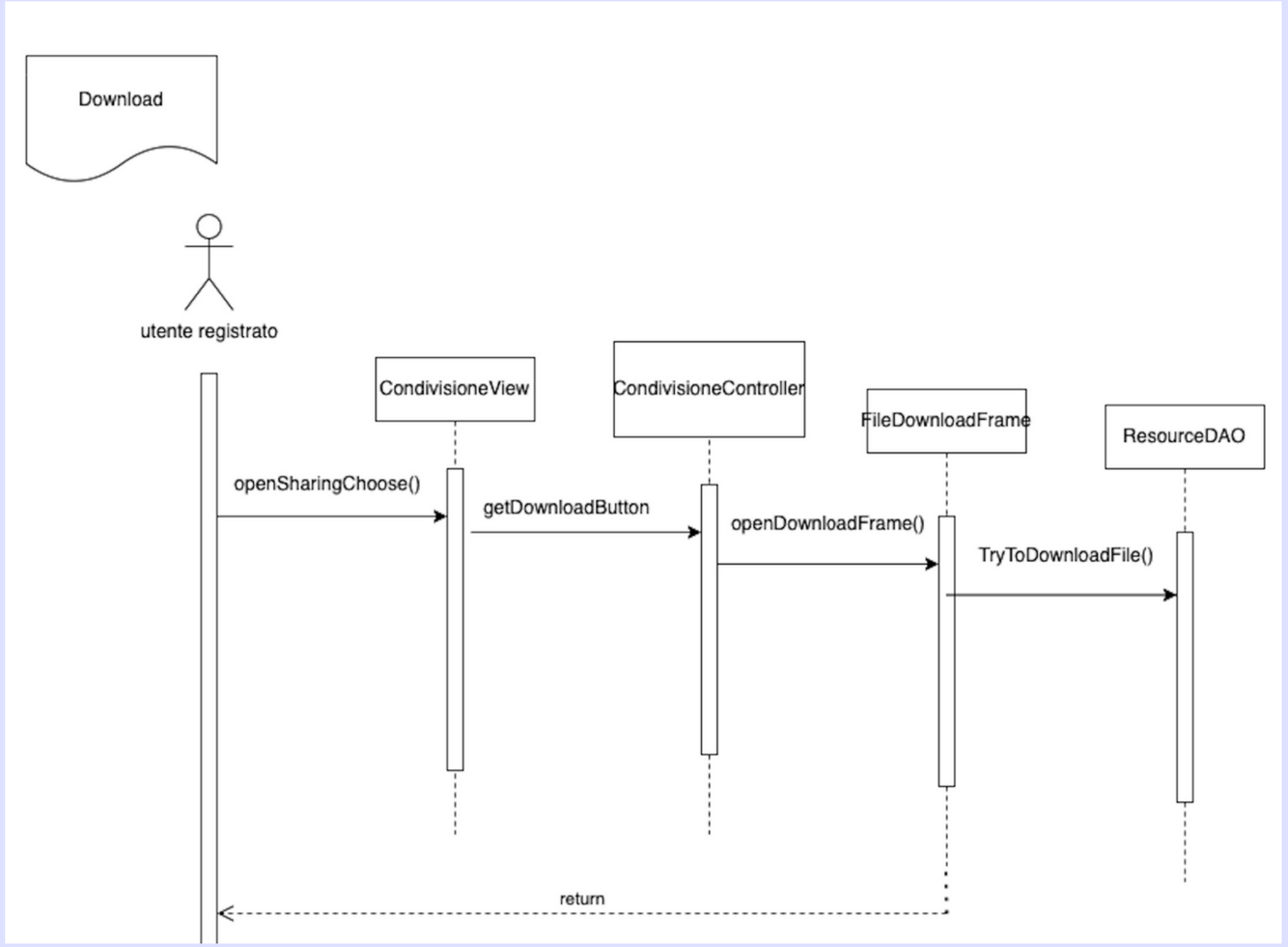


Diagrammi di sequenza chat



Diagrammi di sequenza condivisione file

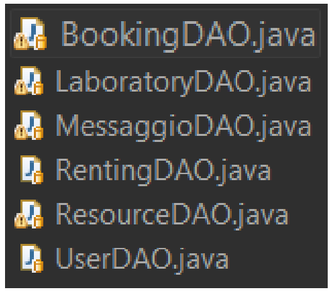




Pattern seguiti

DAO

Per separare la logica di accesso ai dati dal resto dell'applicazione



Singleton

Per avere una sola istanza di alcune classi

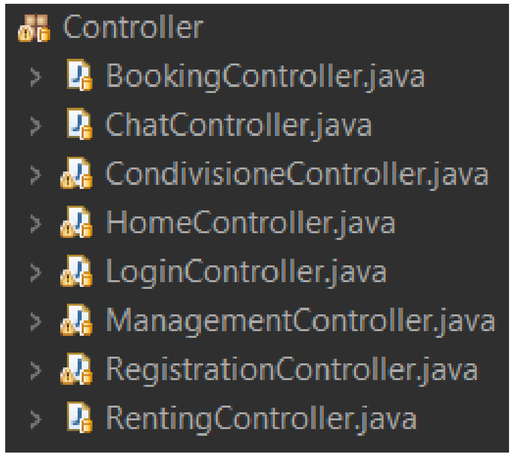
```java
// Costruttore privato
private SingletonManager() {
    // Inizializzazione oggetti
    this.bookingDAO = new BookingDAO();
    this.rentingDAO = new RentingDAO();
    this.resourceDAO = new ResourceDAO();
    this.userDAO = new UserDAO();
    this.laboratoryDAO = new LaboratoryDAO();
    this.messaggioDAO = new MessaggioDAO();
    loggedUser = new User();
}

// Metodo pubblico per ottenere l'istanza Singleton
public static SingletonManager getInstance() {
    if (instance == null) {
        instance = new SingletonManager();
    }
    return instance;
}
```

Pattern seguiti

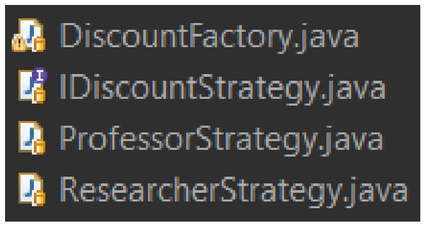
Controller

Controllore per ogni caso d'uso per collegare view e model



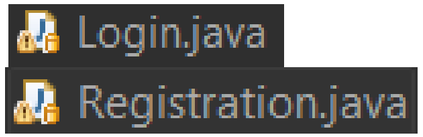
Strategy e factory

Per gestire politiche di sconto variabili



Pure Fabrication

Creazione di oggetti artificiali per login e registrazione



Test autenticazione

Individuazione delle partizioni di equivalenza che sono state poi sottoposte ai test

```java
public class AuthenticationTest {

    private User u[];

    @Before
    public void initTest() {
        u = new User[3];
    }

    @Test
    public void testLoginOk() {

        u[0] = new User("S500816", "000");
        u[1] = new User("P501934", "234");
        u[2] = new User("R509822", "345");

        Login l;

        for (User user : u) {
            l = new Login(SingletonManager.getInstance().getUserDAO().selectUserByMatricola(user));
            assertTrue(l.login());
        }
    }
```

```java
@Test
public void testLoginNotOk() {

    u[0] = new User("S000000", "000");
    u[1] = new User("P5019343213QA", "999");
    u[2] = new User("R509822", "222");

    Login l;

    for (User user : u) {
        l = new Login(SingletonManager.getInstance().getUserDAO().selectUserByMatricola(user));
        assertFalse(l.login());
    }
}
```

Test prenotazione

Test prenotazione risorse disponibili e test prenotazione risorse non disponibili

```java
@Test
public void testBookingOk() {

    b[0] = new Booking(r1, u1, "06/05/2021", "08:00:00", 1);
    b[1] = new Booking(r2, u1, "07/05/2021", "12:00:00", 2);
    b[2] = new Booking(r3, u1, "07/05/2021", "15:00:00", 3);

    for (Booking booking : b) {
        assertTrue(booking.tryToBook());
    }

    clear();

}
```

```java
@Test
public void testBookingNotOk1() {

    b[0] = new Booking(r1, u1, "06/05/2024", "09:00:00", 3);
    b[1] = new Booking(r2, u1, "07/05/2024", "08:00:00", 1);
    b[2] = new Booking(r3, u1, "06/05/2024", "09:00:00", 3);

    for (Booking booking : b) {
        assertFalse(booking.tryToBook());
    }

}
```

FINE