Software Biblioteche Trentine

PROGETTO GRUPPO t00

Giulio Migani

2022

Indice

[Diagramma delle classi 2](#_Toc127474422)

[Sistema per mandare le Email 2](#_Toc127474423)

[Autenticazione nei due applicativi 3](#_Toc127474424)

[Operazioni degli utenti 3](#_Toc127474425)

[Codice in Object Constraint Language 4](#_Toc127474426)

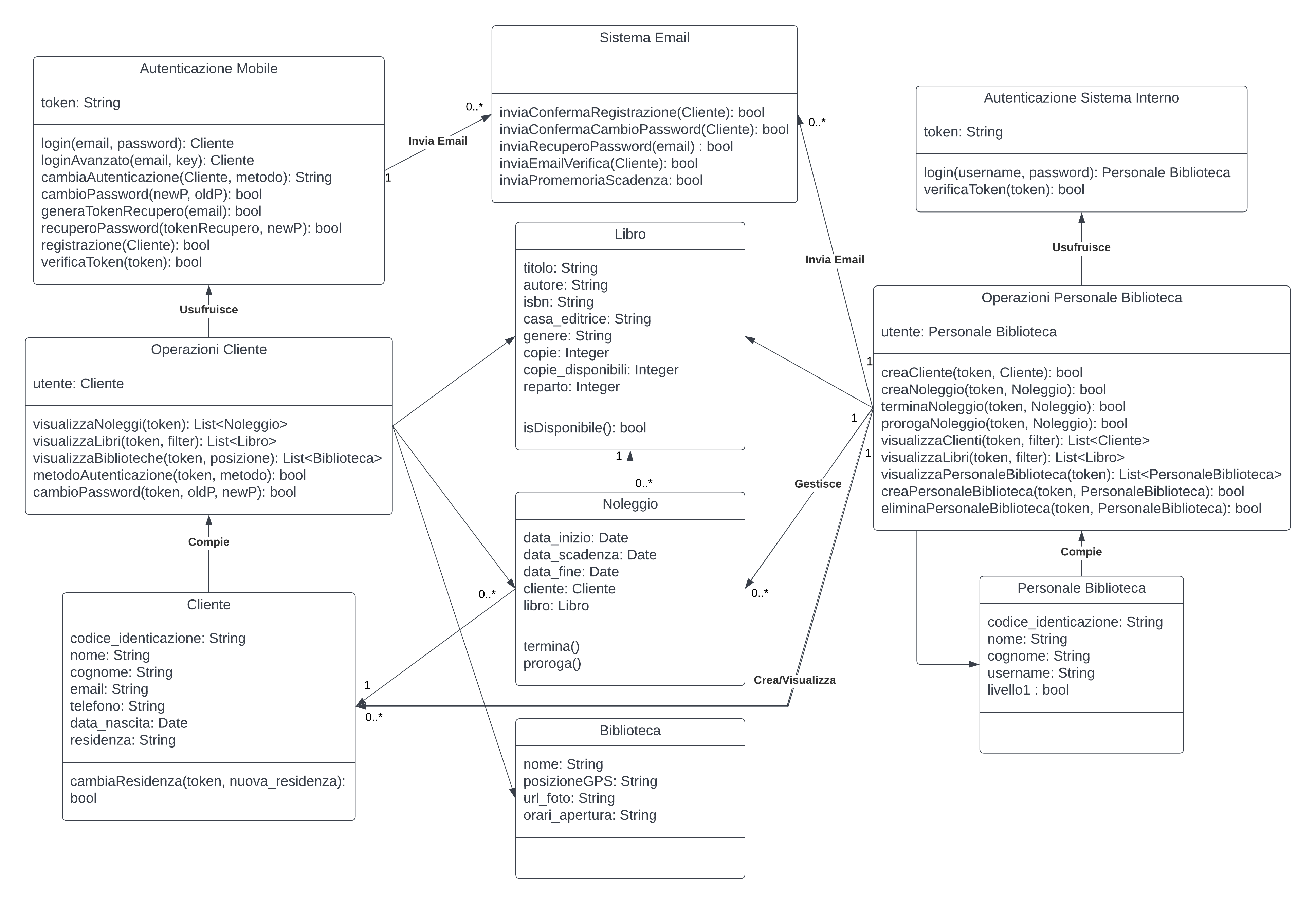
[Creazione del noleggio 4](#_Toc127474427)

[Copie disponibili dei libri 5](#_Toc127474428)

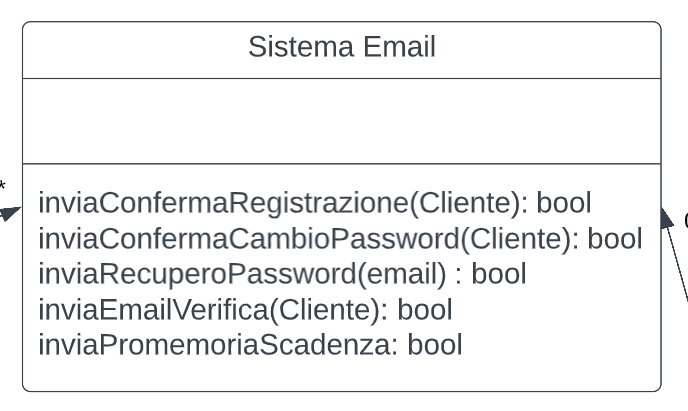
[Date del noleggio 6](#_Toc127474429)

[Gestire Personale Biblioteca 6](#_Toc127474430)

# Diagramma delle classi



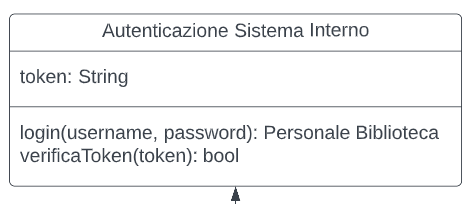
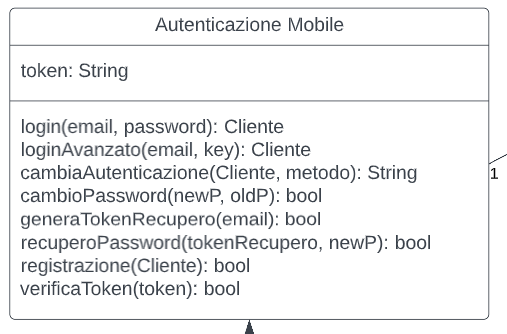
## Sistema per mandare le Email



Per interfacciarsi con il sistema esterno delle mail è prevista una classe di appoggio già predisposta per le varie funzioni presenti nel diagramma dei casi d’uso e nel diagramma delle componenti.

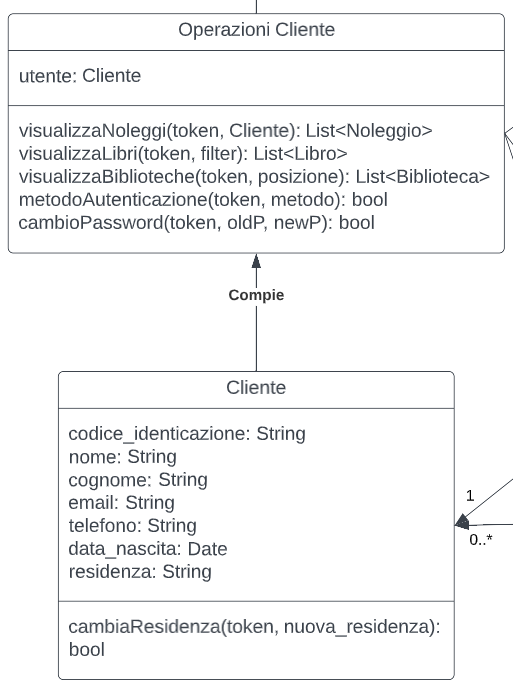
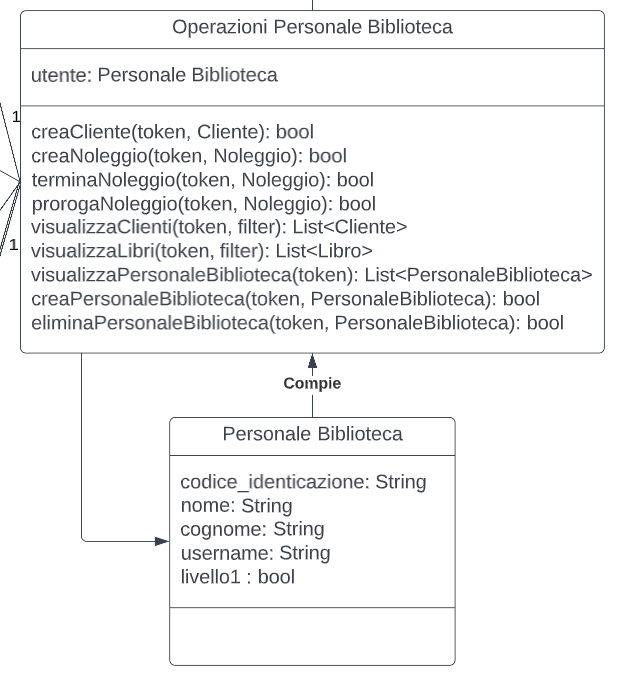
## Autenticazione nei due applicativi

Sono presenti due classi per gestire l’autenticazione perché fanno riferimento a due applicativi distinti. Inoltre, anche le credenziali sono gestite separatamente nel database. Al momento dell’autenticazione viene verificata la corrispondenza con il codice hash associato alla password dell’utente e poi viene generato un token per gestire la sessione dell’utente. Se il login ha successo il metodo login restituisce i dati associati all’utente che si è loggato, altrimenti una variabile nulla. Le password non vengono gestite in nessuna classe del sistema, ma viene verificata direttamente dal database, in questo modo si raggiunge un livello di sicurezza maggiore. Ed è presente un metodo “verificaToken” per gestire la sessione dell’utente.

La classe “Autenticazione Mobile” oltre al metodo login ha anche metodi per cambiare metodo di autenticazione o cambiare/recuperare la password del cliente, è presente il metodo per permettere ad un nuovo utente di registrarsi, il quale genererà la sua password temporanea prima di inserire i dati nel database.

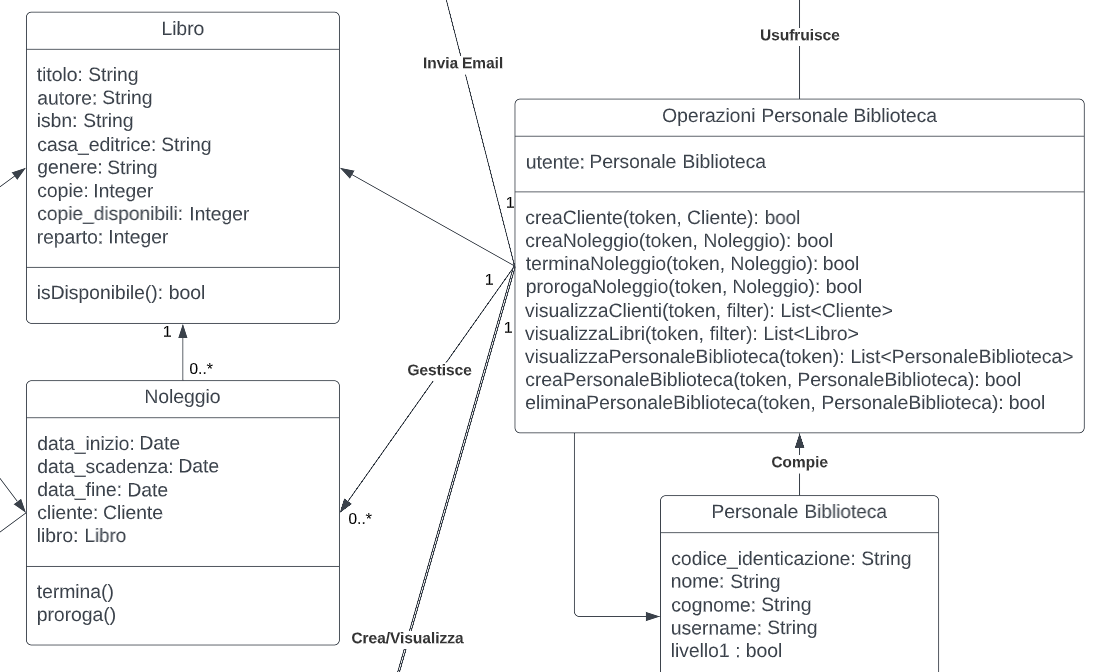
## Operazioni degli utenti

Si è deciso di progettare delle classi apposite per le operazioni dei due tipi di utenti per separare le funzioni delle classi. Il motivo più rilevante è che i metodi non vanno a modificare i dati dell’utente, ma semplicemente vengono compiuti da essi, ad eccezione di “cambiaResidenza” che infatti è stato messo nella classe “Cliente”. Per ognuna di queste operazioni è prevista la convalida del token per poter verificare la sessione dell’utente. Questo è necessario perchè questi metodi saranno usati dal frontend e quindi sarà necessario verificare che la sessione sia valida e non scaduta.

# Codice in Object Constraint Language

## Creazione del noleggio



context Operazioni Personale Biblioteca :: creaNoleggio()

pre : Noleggio.Libro.copie\_disponibili > 0

## Copie disponibili dei libri

Immagine che contiene testo

Descrizione generata automaticamente

context Libro inv :

self.copie >= self.copie\_disponibili

## Date del noleggio

Immagine che contiene testo

Descrizione generata automaticamente

context Noleggio inv :

self.data\_scadenza > self.data\_inizio

## Gestire Personale Biblioteca

Immagine che contiene testo

Descrizione generata automaticamente

context Operazioni Personale Biblioteca :: visualizzaPersonaleBiblioteca()

pre : self.utente.livello1 == True

context Operazioni Personale Biblioteca :: creaPersonaleBiblioteca()

pre : self.utente.livello1 == True

context Operazioni Personale Biblioteca :: eliminaPersonaleBiblioteca()

pre : self.utente.livello1 == True