Documentazione Ingegneria del Software

Leonardo Confente, Manuel Pilato, Giorgio Tosi

Febbraio 2024

Contents

1 Introduzione

1.1 Consegna

Si vuole progettare un sistema informatico per gestire il servizio di prenotazione per il rilascio del passaporto per una Questura, che dispone di più sedi sul territorio. Il sistema di prenotazione è gestito dal personale della Questura, che può inserire il tipo di richiesta di volta in volta considerata (ritiro passaporto, rilascio passaporto per la prima volta, rilascio passaporto per scadenza del precedente, per furto, per smarrimento, per deterioramento, e così via). Il personale attiva di volta in volta le disponibilità per le varie richieste, indicando per ogni specifica richiesta, i giorni, gli orari e le sedi disponibili per i cittadini. Le disponibilità sono indicate rispetto al personale disponibile in Questura. I cittadini che necessitano di un servizio legato al rilascio del passaporto possono accedere al sistema di prenotazione dopo essersi registrati. In fase di registrazione, essi devono indicare nome, cognome, data e luogo di nascita, e codice fiscale. Se il cittadino non rientra nell'anagrafica a disposizione del sistema, il sistema stesso provvederà a indicare tale anomalia al cittadino e ad indicare l'email a cui far pervenire eventuali domande di chiarimento. Una volta registrati, i cittadini accedono al sistema e trovano i servizi per i passaporti a cui possono aderire. Per ogni servizio il cittadino può vedere gli orari e le sedi disponibili giorno per giorno e selezionare il momento desiderato presso la sede opportuna. Il sistema permette ovviamente più prenotazioni e registrazioni contemporaneamente da differenti cittadini. Il sistema verifica che la prenotazione dei servizi sia consistente: ad esempio, non è possibile prenotare il ritiro di un passaporto mai richiesto. Per il ritiro, il sistema permette di prenotare una data, almeno un mese dopo quella della richiesta di rilascio. Il sistema, inoltre, visualizza, dopo l'avvenuta prenotazione, i documenti e le ricevute che i cittadini devono portare con sé al momento della presentazione della richiesta: modulo di richiesta compilato, marca da bollo, ricevuta del versamento su C/C postale, due fototessera su sfondo bianco, passaporto precedente, e così via. Il sistema mostra al cittadino anche la finestra temporale nella quale sono state identificate le disponibilità, ed evidenzia opportunamente i periodi per i quali non sono ancora state inserite le disponibilità da parte della Questura. In questo modo il cittadino è in grado di distinguere slot occupati, slot liberi e slot non ancora gestiti. Il sistema avvisa il cittadino, che ne abbia fatto richiesta attraverso il sistema, rispetto al momento in cui un certo periodo di tempo sarà disponibile per le prenotazioni di un certo servizio. Il sistema memorizza i dati demografici essenziali per ogni cittadino: codice fiscale, numero di tessera sanitaria, cognome, nome, luogo e data di nascita, specifiche categorie di appartenenza (cittadini con figli minori, cittadini con passaporto diplomatico, e così via), e verifica che i dati delle registrazioni dei singoli cittadini siano corrette rispetto all'anagrafica a disposizione.

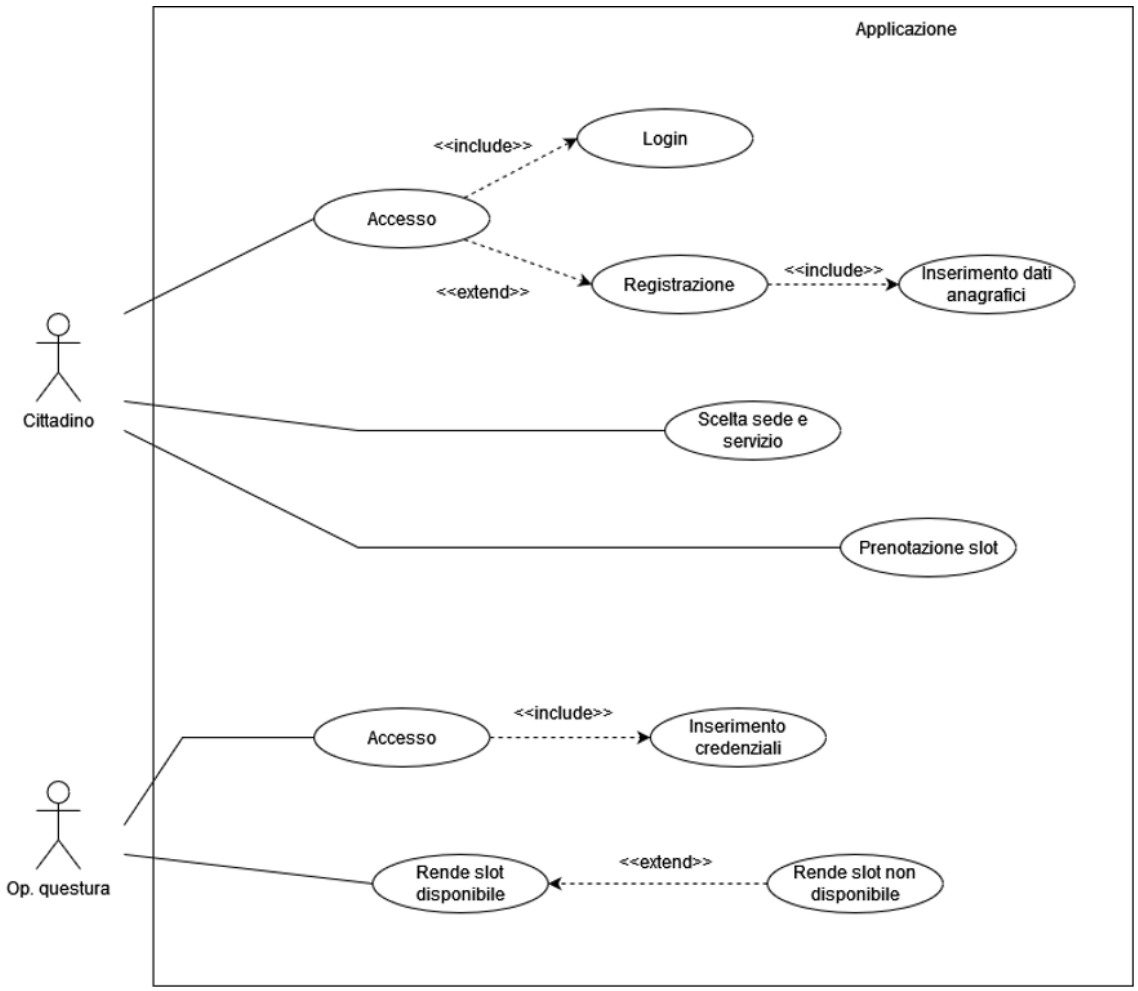
1.2 Metodologia di lavoro adottata

La metodologia di lavoro da noi adottata si avvicina molto al modello **plan driven**, infatti tutte le attività del processo sono state pianificate in anticipo secondo un modello a cascata che prevedeva l'analisi iniziale dei requisiti, la progettazione del software da un punto di vista ad alto livello, la realizzazione dello stesso tramite il linguaggio di programmazione Java e una fase finale di test per verificare la corretta funzionalità del prodotto finale. Questa metodologia di sviluppo è stata usata maggiormente nella parte di sviluppo dell'interfaccia grafica e di tutta la parte relativa ai controlli gestiti da quest'ultima, la parte del progetto riguardante la gestione dei dati (memorizzazione lettura ecc.) è stata ideata seguendo una strategia più vicina al **metodo agile** con implementazione **incrementale**: man mano che venivano aggiunte nuove funzionalità, si procedeva con una fase di test e validazione.
Il lavoro è stato gestito con una metodologia ispirata alla **tecnica di scrum**: ogni task è stata valutata rispetto alla sua difficoltà di implementazione, in base a questo dato poi sono state divise in modo equo. Le parti che sono risultate più difficili sono state fatte in Pair Progamming. Inoltre, è stato fatto il **Refactory** di alcuni pezzi del codice, in modo da diminuire il tempo di implementazione. Il lavoro e la condivisione del progetto sono stati gestiti grazie al software di controllo di versione git e GitHub che permette di lavorare in gruppo in maniera efficiente anche modificando il progetto in maniera asincrona (su branch differenti) per poi combinare le diverse modifiche (merge) fatte in un secondo momento. Ciò ha permesso maggiore accessibilità alle ultime versioni progettuali e una migliore comprensione delle modifiche.

2 Struttura del progetto

2.1 Casi d'uso principali

Un **diagramma dei casi d'uso** è un tipo di diagramma UML che descrive le funzioni o i servizi offerti da un sistema. Si compone di quattro elementi principali: sistema, attori, casi d'uso e relazioni. Il **sistema** è il contesto in cui si svolgono le funzioni, gli **attori** sono le entità che svolgono un ruolo nel sistema, **i casi d'uso** sono le funzioni o le azioni che il sistema offre agli attori, e le **relazioni** sono i collegamenti tra gli elementi del diagramma.
Il diagramma dei casi d'uso raffigurato mostra i modi principali per utilizzare questa applicazione, di seguito sono riportate le singole schede specifiche per i casi d'uso più importanti.



Use Case Diagram

2.2 Schede di specifica

2.1.1 Accesso utente

Precondizioni: nessuna

Attore: Cittadino

Passi:

1. Il cittadino entra nella pagina Accedi/Registrati

Postcondizioni: Il cittadino è pronto ad accedere se già registrato, altrimenti puo' registrarsi

Sequenza alternativa: In qualunque momento l'utente se accidentalmente è entrato nella sezione sbagliata può tornare indietro

Postcondizioni: L'utente è stato reindirizzato alla pagina di Benvenuto

Scheda di specifica dell'*Accesso utente*

2.1.2 Registrazione cittadino

Precondizioni: l'utente deve essere entrato nella pagina Accedi/Registrati e i suoi dati anagrafici devono essere presenti nell'anagrafe della questura

Attore: Cittadino

Passi:

1. Il cittadino inserisce correttamente tutti i dati anagrafici richiesti e fornisce una password

2. Il cittadino clicca il bottone "Registrati"

Postcondizioni: Se tutti i dati risultano corretti e il cittadino era presente nell'anagrafica della questura, un nuovo cittadino si è registrato nel sistema e ora può accedere

Sequenza alternativa: In qualunque momento il cittadino può abbandonare la pagina di registrazione e tornare indietro

Postcondizioni: L'utente è stato reindirizzato alla pagina Accedi/Registrati

Scheda di specifica della *Registrazione come cittadino*

2.1.3 Login cittadino

Precondizioni: l'utente deve essere registrato

Attore: Cittadino

Passi:

1. Il cittadino inserisce il suo codice fiscale

2. Il cittadino inserisce la sua password

3. Il cittadino clicca il bottone "Accedi"

Postcondizioni: un cittadino, se ha inserito dati corretti che effettivamente corrispondono con quelli con cui si è registrato, è finalmente entrato nel menu dove puo' selezionare il servizio di cui ha bisogno e può scegliere in quale sede si dovrà recare

Scheda di specifica dell'*Login come cittadino*

2.1.4 Scelta della sede e del servizio

Precondizioni: l'utente deve essere autenticato

Attore: Cittadino

Passi:

1. Cittadino sceglie il servizio

2. Cittadino sceglie la sede

Postcondizioni: Il cittadino è entrato nella pagina calendario dove può cercare gli slot che la questura della sede che ha selezionato ha messo a disposizione per il servizio selezionato

Sequenza alternativa 1: In qualunque momento il cittadino può abbandonare la pagina di menù principale e tornare indietro

Postcondizioni: L'utente è stato reindirizzato alla pagina Accedi/Registrati

Sequenza alternativa 2: In qualunque momento il cittadino può abbandonare la pagina di menu principale e tornare alla pagina iniziale

Postcondizioni: L'utente è stato reindirizzato alla pagina di Benvenuto

Scheda di specifica della *Scelta del servizio*

2.1.5 Prenotazione slot

Precondizioni: l'utente deve essere autenticato e aver selezionato servizio e sede

Attore: Cittadino

Passi:

1. Cittadino sceglie tra gli slot che sono disponibili (gli slot verdi)

2. Cittadino clicca bottona "Prenota"

Postcondizioni: Se la prenotazione è avvenuta correttamente la questura accetta la prenotazione del cittadino che risulta correttamente prenotato nello slot selezionato che ora risulta occupato (slot rosso)

Sequenza alternativa 1: In qualunque momento il cittadino può abbandonare la pagina di prenotazione e tornare indietro

Postcondizioni: L'utente è stato reindirizzato alla pagina di menù principale

Sequenza alternativa 2: In qualunque momento il cittadino può abbandonare la pagina di prenotazione e tornare alla pagina iniziale

Postcondizioni: L'utente è stato reindirizzato alla pagina di Benvenuto

.

Scheda di specifica della *Prenotazione slot*

2.1.6 Accesso Operatore questura

Precondizioni: L'operatore deve già risultare registrato

Attore: Operatore questura

Passi:

1. Operatore inserisce le sue credenziali

2. Operatore clicca sul bottone "Accedi"

Postcondizioni: Il cittadino è entrato nella pagina calendario lato questura dove può scegliere quali slot rendere disponibil

Sequenza alternativa 1: In qualunque momento l'operatore può abbandonare la pagina di inserimento delle credenziali e tornare indietro

Postcondizioni: L'operatore è stato reindirizzato alla pagina di Benvenuto

Scheda di specifica del *Login come operatore*

2.1.4 Disponibilità slot

Precondizioni: L'operatore deve essere entrato correttamente con le sue credenziali con cui risulta registrato

Attore: Operatore questura

Passi:

1. L'operatore sceglie il servizio

2. L'operatore seleziona lo slot

3. L'operatore lo rende disponibile(o non disponibile)

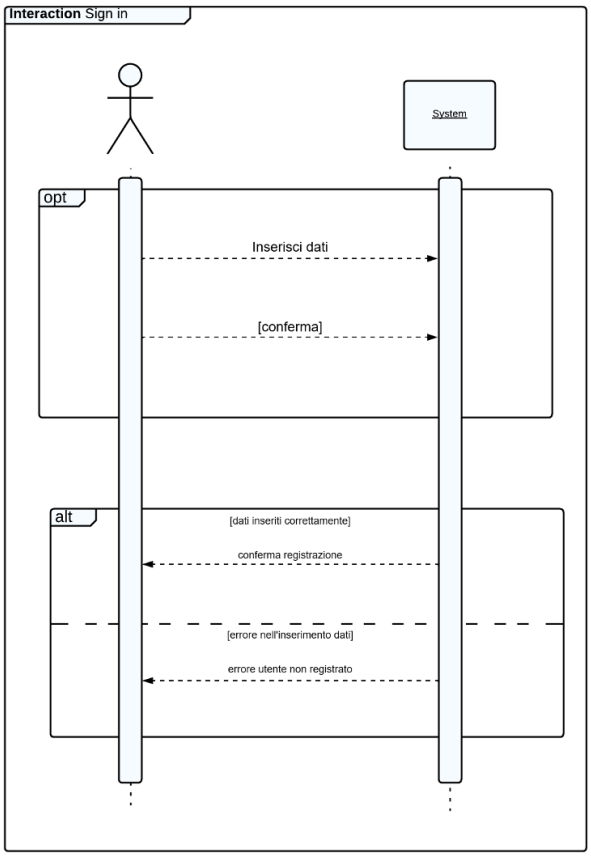
Postcondizioni: Una volta confermata la disponibilità di uno slot per un particolare servizio, la pagina calendario da lato cittadino, se si è entrati selezionando quella sede e quel servizio, visualizzerà lo slot disponibile e sarà possibile prenotarlo.

Scheda di specifica del *Centro di controllo operatore*

2.3 Sequence Diagram dei principali Casi d'uso

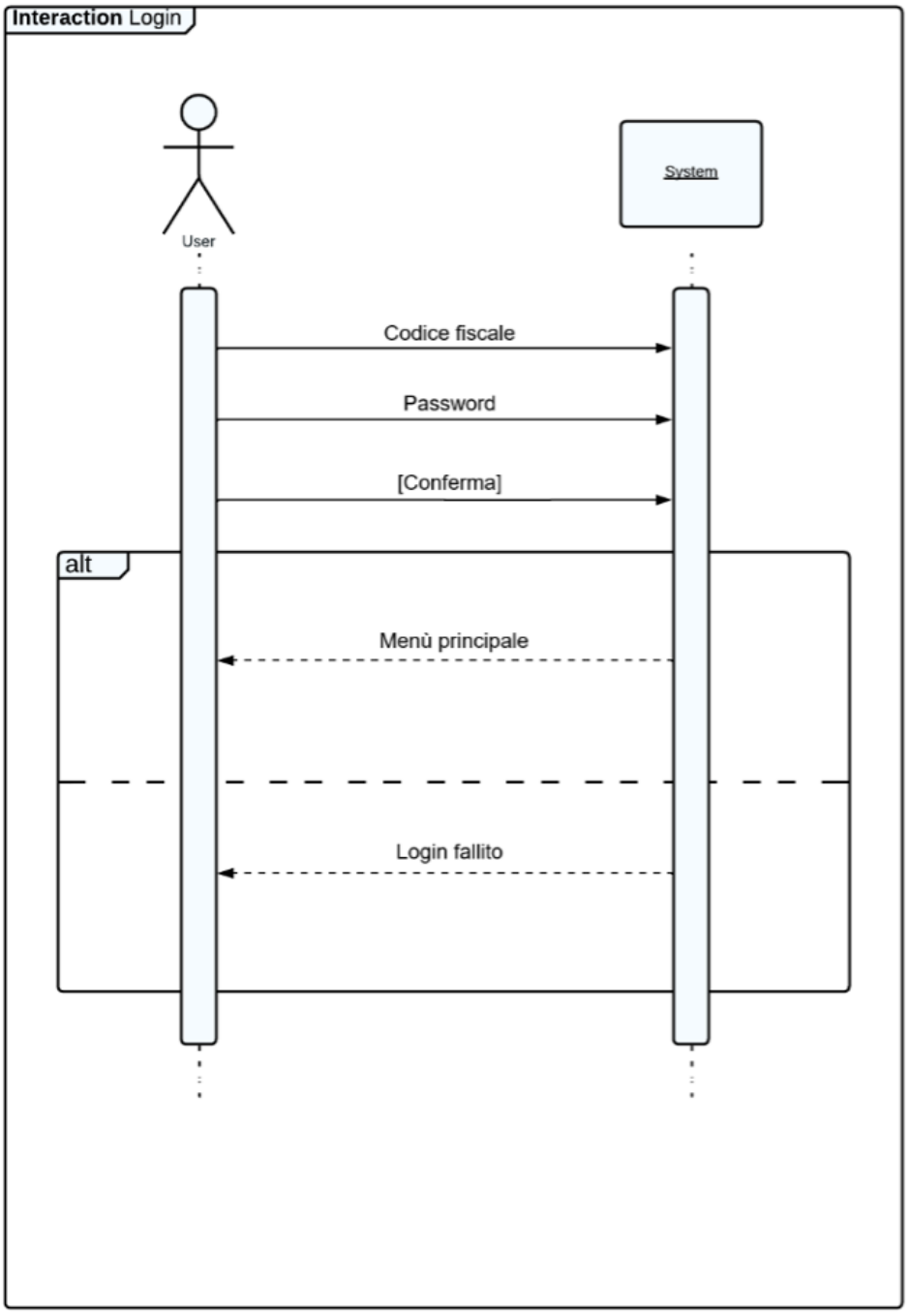
Un diagramma di sequenza dei casi d'uso serve a visualizzare il flusso di esecuzione di uno scenario di un caso d'uso, a evidenziare le collaborazioni tra gli attori e i casi d'uso e a definire le condizioni e le alternative di un'interazione. Si compone di tre elementi principali: linee di vita, messaggi e frame. Le **linee di vita** sono le rappresentazioni grafiche degli attori e dei casi d'uso coinvolti nell'interazione. I **messaggi** sono le comunicazioni che avvengono tra le linee di vita, come richieste, risposte, eventi o segnali. I **frame** sono le aree delimitate da una linea tratteggiata che contengono un'etichetta che indica il tipo di interazione.

Registrazione:
Per poter accedere al sistema, l'utente deve prima inserire i propri dati che verranno salvati in un apposito Database. Dopodiché potrà accedere con le proprie credenziali, se corrette. Dopo la registrazione, l'utente verrà reindirizzato alla pagina di login.



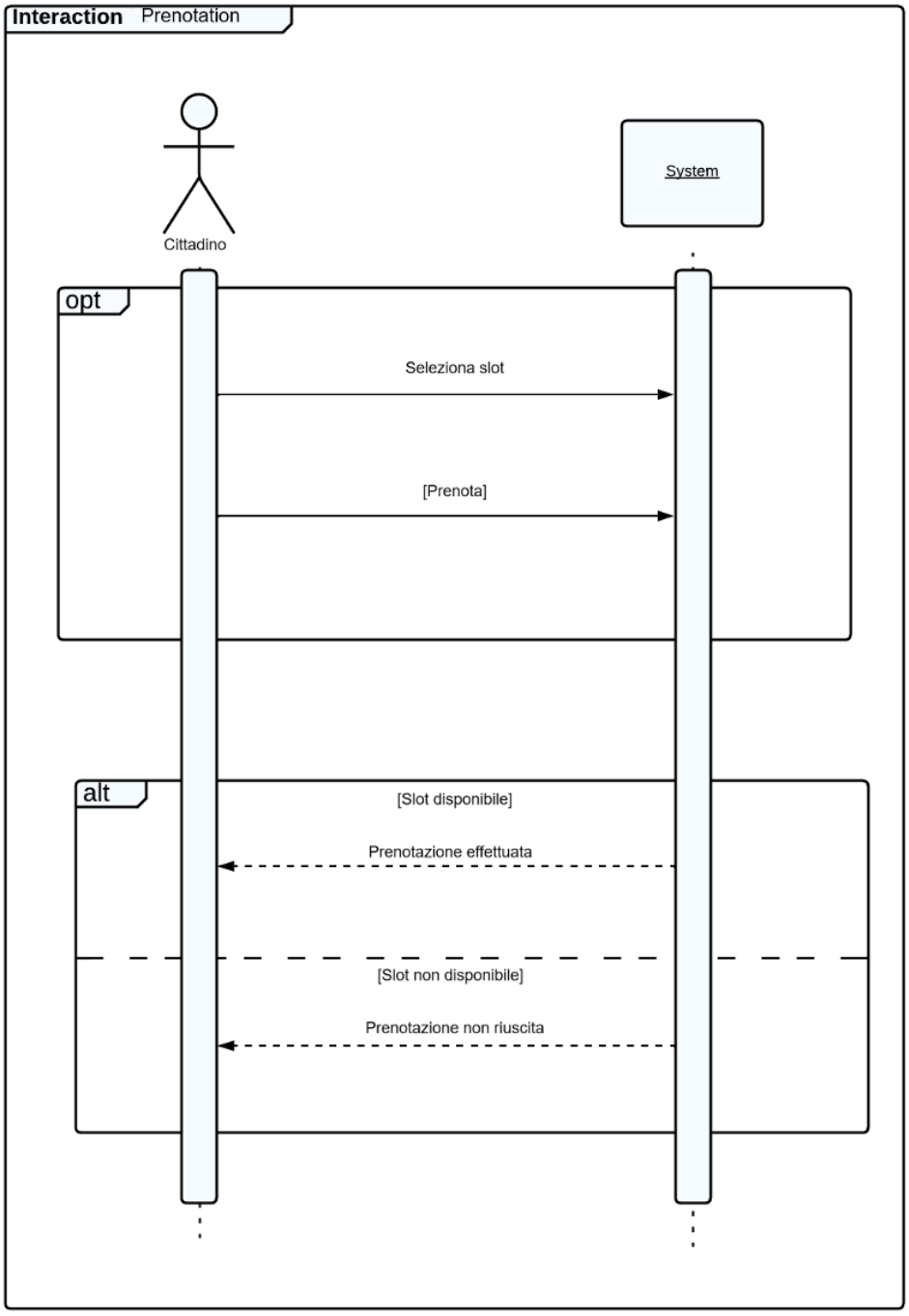
Sequence Diagram della fase di Registrazione

Log in:
Nella fase di Login, l'utente deve inserire i propri dati con cui si è precedentemente registrato ed il sistema effettuerà una verifica di correttezza. Una volta autentificati, i cittadini verranno reindirizzati al menù principale da cui potranno selezionare servizio e sede.



Sequence Diagram della fase di Login

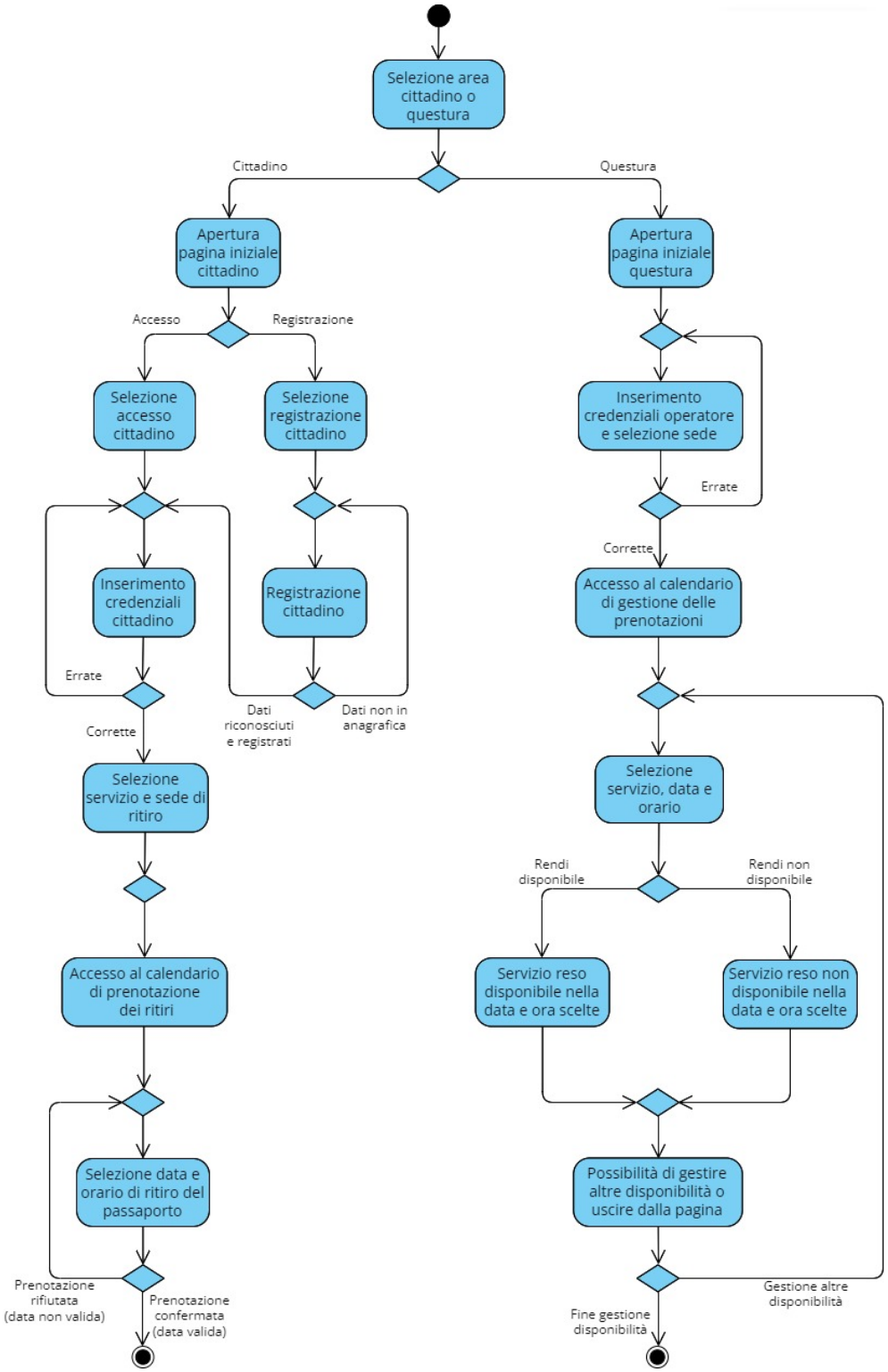
Prenotazione:
In fase di prenotazione, l'utente tenta di prenotare uno slot. Se l'operazione non dovesse concludersi con successo, l'utente rimarrà nella pagina corrente e potrà effettuare un nuovo tentativo.



Sequence Diagram della fase di Registrazione

2.4 Activity Diagram

Il seguente diagramma delle attività rappresenta le possibili opzioni che un utente può scegliere di eseguire mentre utilizza il sistema. Sono state rappresentate le possibilità di ripetere le operazioni di inserimento delle credenziali in caso di errori e di eseguire più volte le attività riguardanti le prenotazioni, sia dal lato cittadino sia dal lato questura. Per rendere il diagramma delle attività più leggero e chiaro da interpretare, non sono state rappresentate le possibilità di tornare alla pagina iniziale e alla pagina precedente, che sono invece sempre disponibili.



Activity Diagram

3 Implementazione

3.1 Classi implementate

Per lo sviluppo del software, è stato usato un pattern architetturale MVC (Pagina 26, 5.1). Ciò comporta la suddivisione delle classi in base al ruolo. Ciò comporta la suddivisione delle classi in base al ruolo:

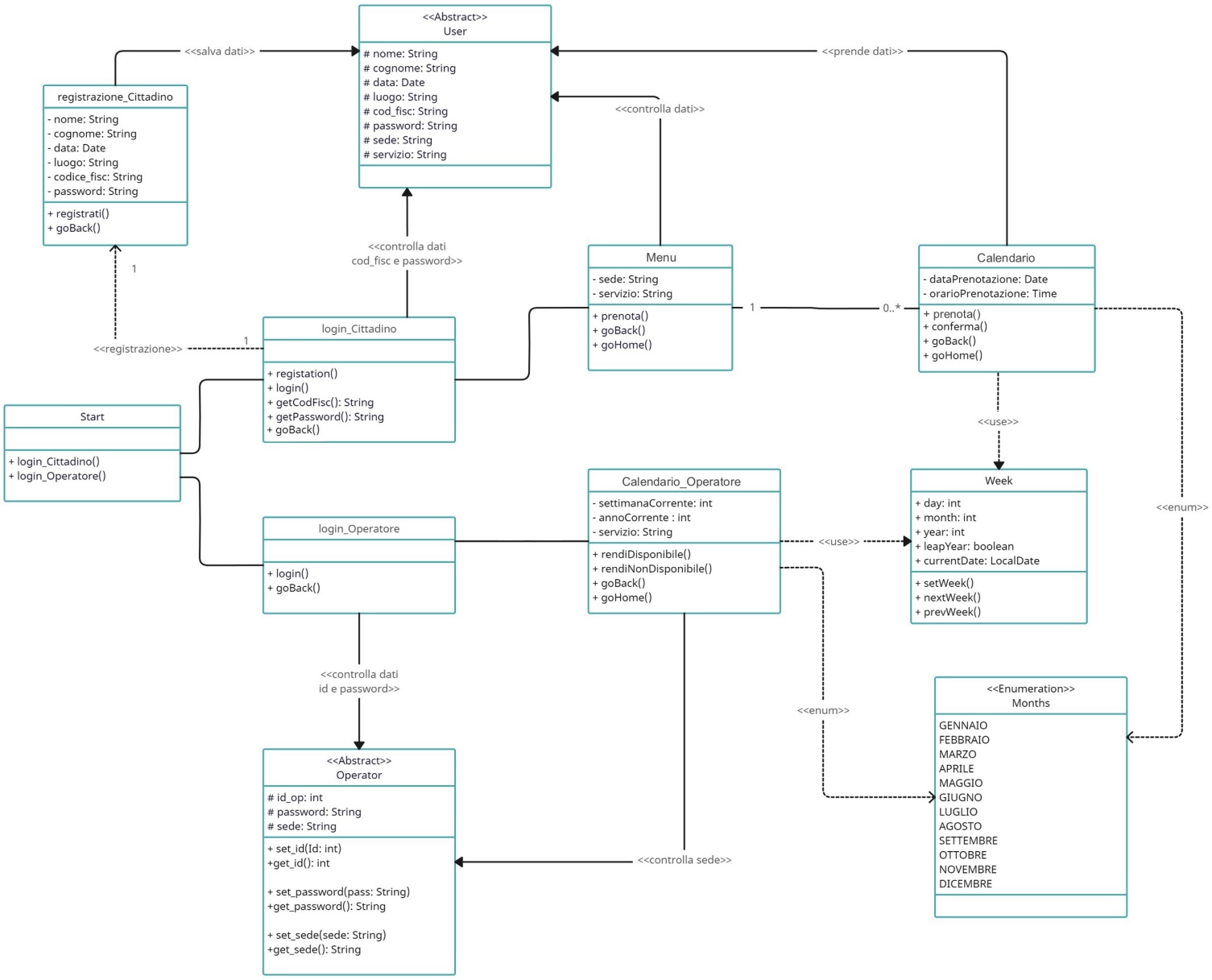
- **Model:**
  - User
  - Operator
  - Week
- **View:**
  - start_page
  - Usr_login_page
  - Usr_registrer_page
  - Op_login_page
  - Usr_menu_page
  - Usr_calendar_page
  - Op_calendar_page
- **Controller:**
  - start_Controller
  - Usr_loginController
  - Op_LoginController
  - Usr_registerController
  - Usr_MenuController
  - Usr_CalendarController
  - Op_CalendarController

Nota: le classi della parte Model e Controller sono in .java, le classi grafiche, ovvero la parte View in .fxml

3.2 Class Diagram

Il diagramma delle classi è uno strumento grafico fondamentale nell'ambito della progettazione software object-oriented, fornendo una rappresentazione visiva delle classi, dei loro attributi e delle relazioni interne all'interno di un sistema. Questo strumento agevola la comprensione della struttura e dell'organizzazione del software, permettendo di visualizzare in modo chiaro e intuitivo le connessioni tra gli elementi del sistema.

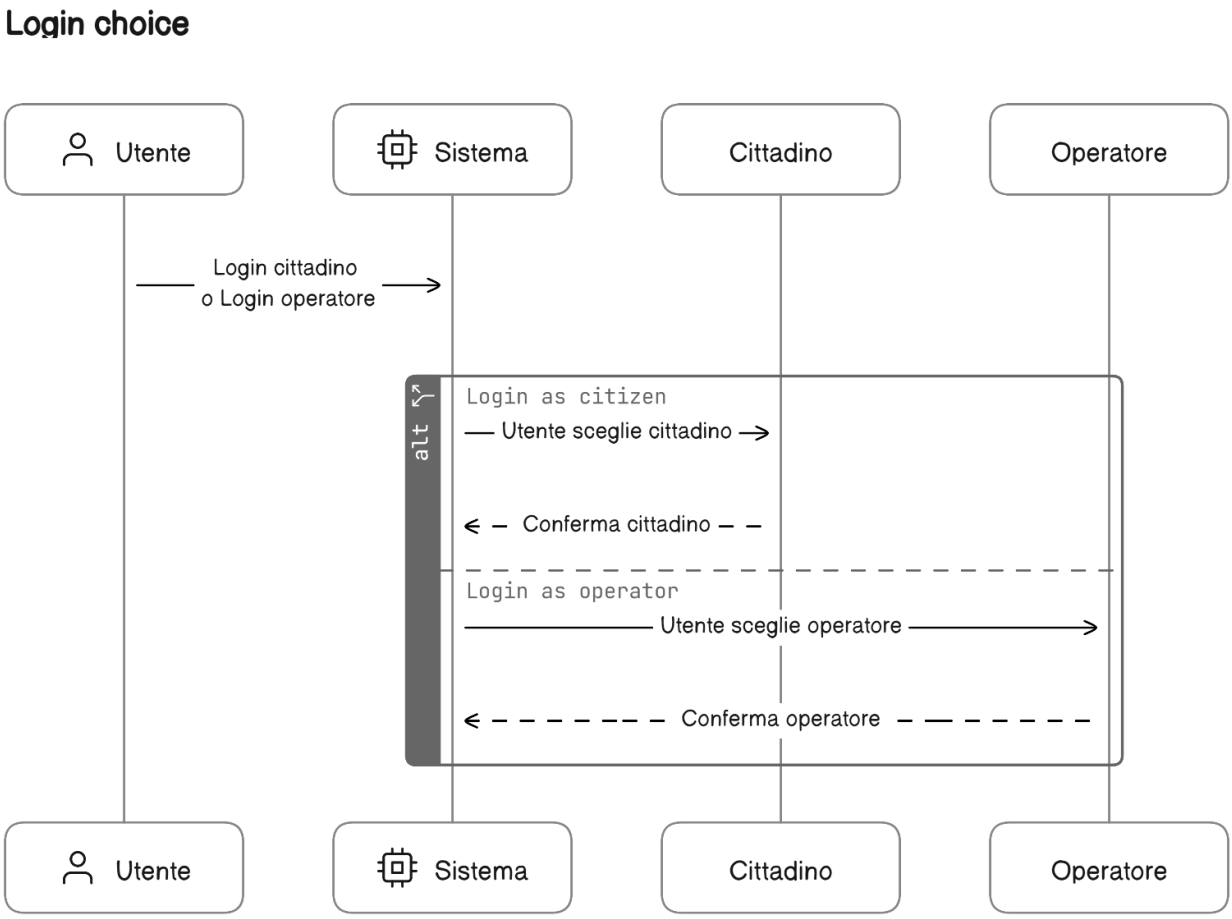
All'interno del sistema in esame, le classi *User*, *Operator*, e *Week* costituiscono la componente di **Model**, la quale svolge un ruolo nell'archiviazione dei dati. Ciascuna di queste classi esegue una verifica nel database per assicurarsi della corretta registrazione dei dati inseriti da parte dell'utente. Questo processo di controllo garantisce l'integrità e l'affidabilità delle informazioni archiviate.



Nota: La parte di interfaccia grafica e Controllo DB sono state omesse.

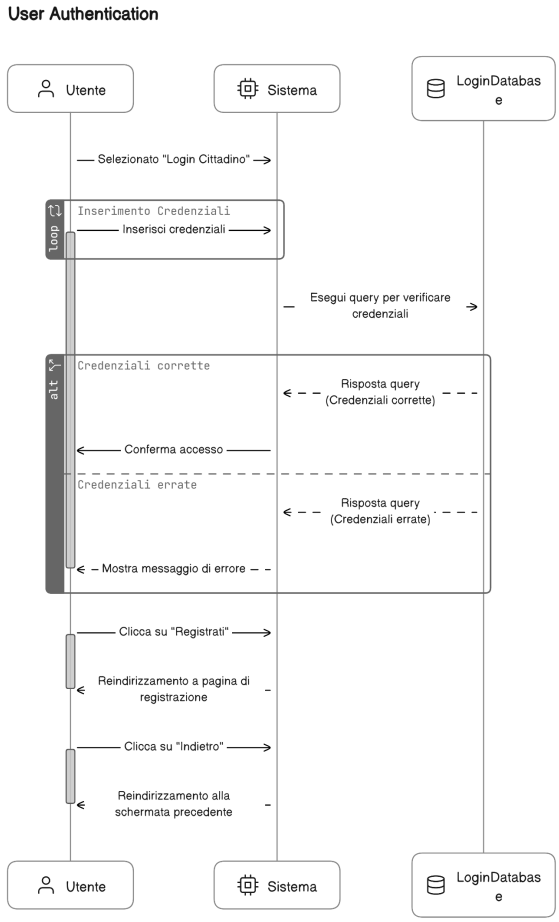
3.3 Sequence Diagram

I diagrammi di sequenza permettono di rappresentare interazioni tra classi in modo grafico. Essi rappresentano una parte importante della documentazione per facilitare la comprensibilità del codice e del suo scopo. Le interazioni rappresentate graficamente sono quelle maggiormente utilizzate nel software finale.

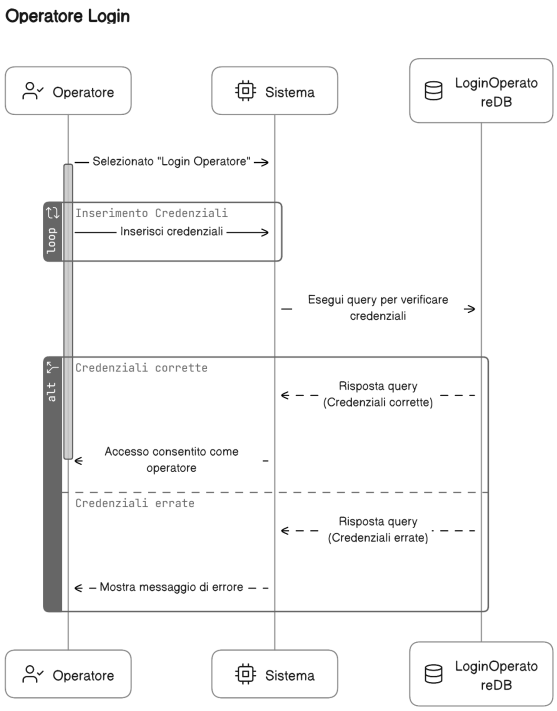


Scelta utente

Fase di login in veste di Cittadino o Operatore della questura. L'inserimento credenziali deve andare a buon fine, quindi viene domandato al database di verificare la consistenza dei dati inseriti. Il cittadino inoltre ha la possibilità di essere indirizzato nella pagina di registrazione dal sistema.

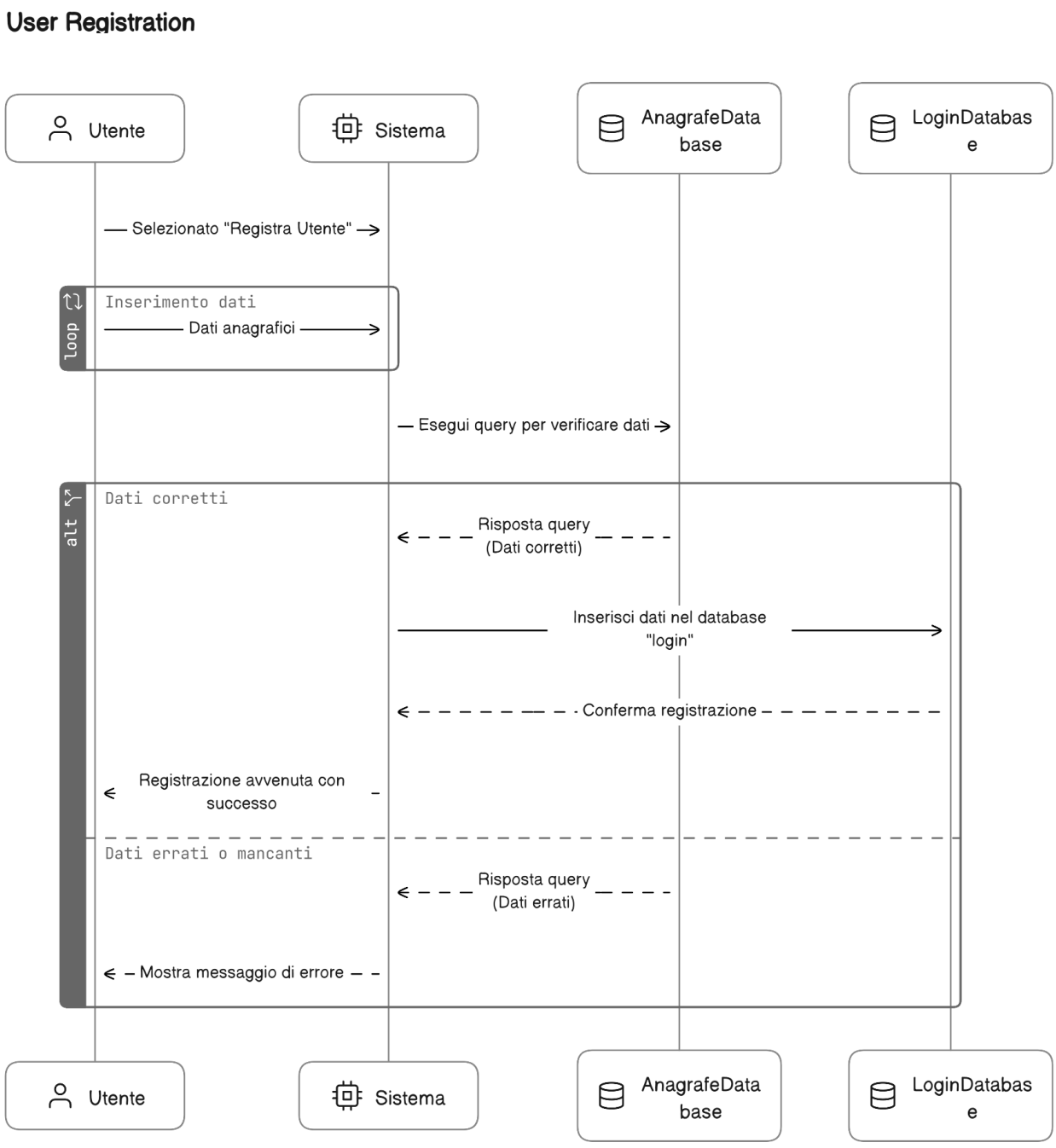


Login come cittadino



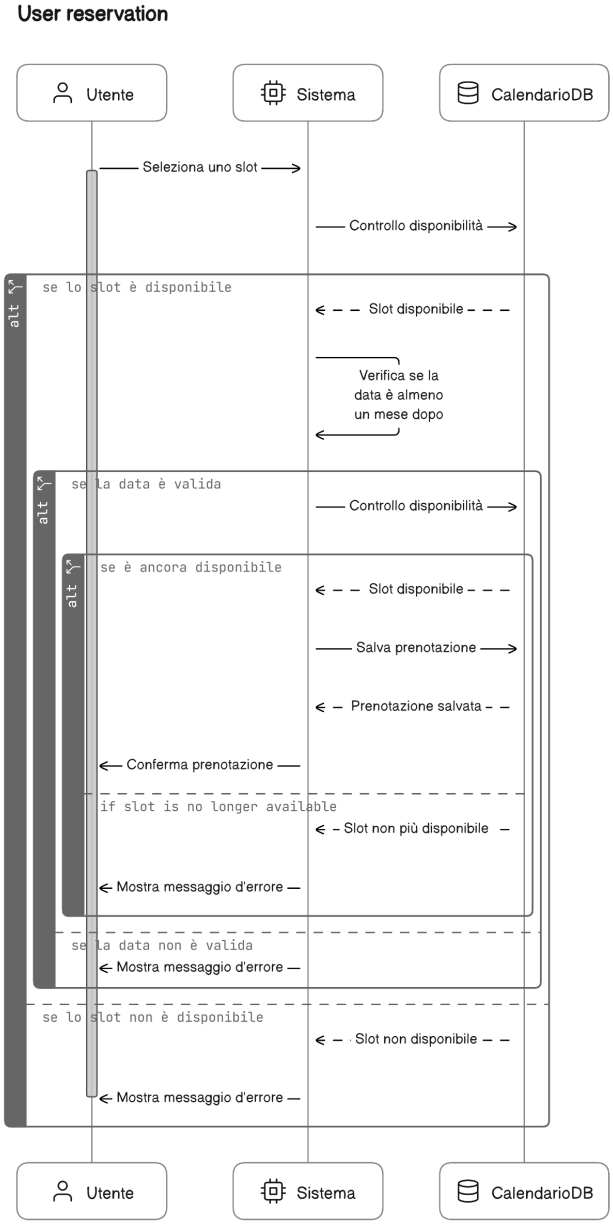
Login come operatore

Durante la fase di registrazione, il cittadino deve inserire dei dati anagrafici che devono essere presenti nell'anagrafe del sistema, questo al fine di evitare che l'utente inserisca dati non veritieri. Se tutti i dati corrispondono a quelli in anagrafe (password esclusa), allora l'utente viene registrato come cittadino ed è in grado di accedere al sistema di prenotazione della questura.



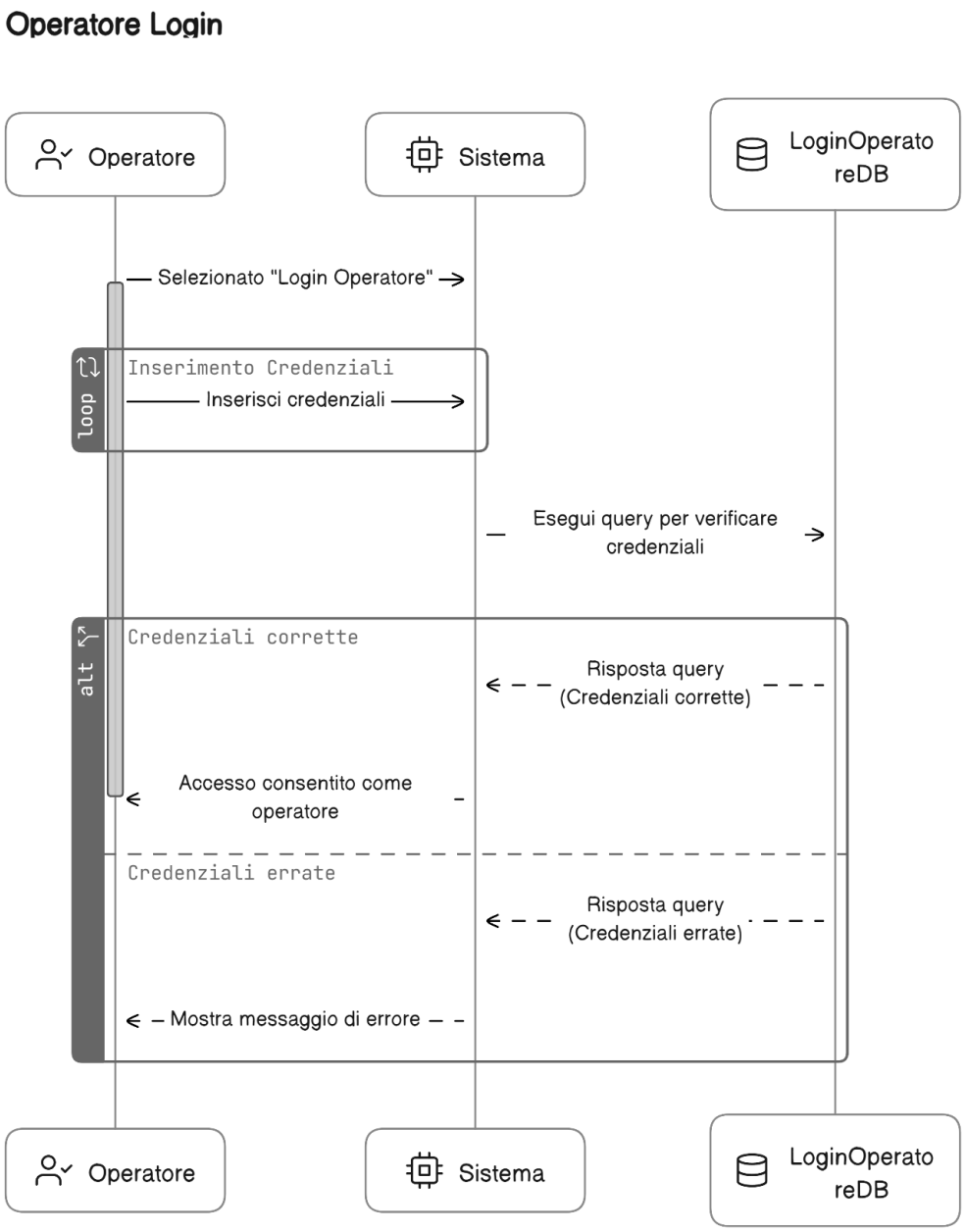
Registrazione cittadino

A questo punto il cittadino può prenotare uno slot nel calendario della questura, dopo aver specificato la sede e il motivo della sua richiesta. Le condizioni affinchè lo slot venga prenotato sono: che sia disponibile, che sia almeno un mese dopo la richiesta, che durante la prenotazione essa non sia stata occupata da un altro cittadino. L'ultima condizione soddisfa la consistenza del programma per il fatto che il sistema permette l'accesso simultaneo di più utenti e ciò potrebbe causare una sovrapposizione di richieste.



Prenotazione slot

Dopo aver effettuato il login, l'operatore ha accesso al calendario e può svolgere due funzioni: rendere slot disponibili oppure non disponibili. Le modifiche, dopo essere state accertate, andranno salvate nel DB che contiene tutte le informazioni sul calendario.



Slot disponibili/non disponibili

4 Attività di test

4.1 Testing e validazione

Le attività di testing di correttezza e solidità sono state un processo eseguito per tutta la durata di creazione del progetto e consistono nelle seguenti attività:

1. Confronto dei diagrammi iniziali e del documento di analisi preliminare delle specifiche con il documento delle specifiche del progetto
2. Verifica costante della consistenza tra codice prodotto e specifiche del progetto
3. Verifica del corretto funzionamento delle funzionalità in fase di combinazione dei codici prodotti dai vari componenti del gruppo
4. Testing approfondito di alcune classi problematiche al fine di eliminare errori, ottimizzarne il funzionamento o adattarle a nuove funzionalità implementate in seguito
5. Testing di consistenza e di corretto funzionamento delle funzionalità che sfruttano il database
6. Test degli sviluppatori sul progetto
7. Test di utenti generici sul progetto

4.2 Ispezione di diagrammi e codice

Da una prima analisi delle specifiche del progetto è stato creato un documento finalizzato a evidenziare quali fossero gli utenti e a raccogliere e suddividere tutte le attività che dovevano essere disponibili alle varie tipologie di utenti del software. Questo documento è stato utilizzato come base di creazione dei vari diagrammi e come guida per scegliere quali fossero le prime funzionalità che andavano implementate. In seguito alla produzione sia dei diagrammi, sia di parti di codice, ci siamo assicurati di non esserci distanziati dalle richieste o di non aver commesso errori facendo riferimento al documento riassuntivo prodotto e in caso di dubbio alla specifica originale.

4.3 Verifiche di corretto funzionamento del codice

Le varie parti di codice sono state testate sia in fase di creazione, sia una volta ultimate prima di essere combinate con le altre parti di progetto già ultimate, al fine di ridurre al minimo il rischio di implementare codice che potesse compromettere il funzionamento corretto del progetto costringendoci a dover apportare modifiche importanti a parti già funzionanti. Nello specifico il testing maggiore si è concentrato su due parti del progetto:

- **Calendario prenotazioni:** È stata scelta una visualizzazione settimanale del calendario. Il testing era finalizzato ad assicurarsi che non si creassero errori in fase di calcolo delle date della settimana visualizzata e dell'allineamento giorno della settimana/data incentrandosi sui punti critici come i cambi di mese e di anno. Inoltre ci si è assicurati che i dati relativi alle prenotazioni fossero sempre correttamente recuperati e visualizzati sul calendario.

- **Database:** Svariate parti del progetto leggono da/scrivono sul database. I test hanno riguardato la gestione delle registrazioni e dei login e la gestione delle prenotazioni. Per quanto riguarda la gestione delle registrazioni, i test si sono concentrati sull'assicurarsi che venissero registrati solo utenti i cui dati corrispondevano con quelli presenti nell'anagrafica a disposizione della questura. Anche per quanto riguarda i login i test si sono concentrati sull'assicurarsi che i dati inseriti dall'utente corrispondessero a quelli a disposizione del sistema. Infine i test riguardanti il calendario erano volti al verificare non solo che le disponibilità fossero sempre mostrate correttamente, ma anche che il sistema garantisse la consistenza delle prenotazioni. Abbiamo infatti forzato modifiche allo stato di prenotazioni agendo direttamente sul database, al fine di simulare l'attività di altri utenti in fase di prenotazione e ci siamo assicurati che il sistema non creasse prenotazioni errate o non sovrascrivesse prenotazioni fatte contemporaneamente da altri utenti. Le attività di testing in questo caso si sono concentrate sul mettere alla prova l'aggiornamento corretto dei dati in fase di prenotazione e il controllo che l'operazione fosse ancora eseguibile nel nuovo stato del database modificato.

4.4 Unit Testing

Di seguito si riporta il codice rappresentante lo Unit Test eseguito sulla classe Week, che gestisce i calcoli delle date delle settimane da mostrare nel calendario e lo Unit Test eseguito sulla classe Usr_LoginController nel controllo di correttezza delle credenziali in fase di accesso alla piattaforma.

```
package com.example.demo;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class WeekTest {
    @Test
    void testSetWeek() {
        Week week = Week.getInstance(true);
        //settare testWeekArray con valori settimana corrente
        int[] testWeekArray = {5, 6, 7, 8, 9, 10, 11};
        assertArrayEquals(week.getWeek(), testWeekArray);
    }
    @Test
    void testNextWeek() {
        Week week = Week.getInstance(true);
        int n_weeks_forward = 1;
        do{
            week.nextWeek();
            n_weeks_forward--;
        }while (n_weeks_forward != 0);
        //settare testWeekArray con valori
          della n-esima settimana successiva
        int[] testWeekArray = {12, 13, 14, 15, 16, 17, 18};
        assertArrayEquals(week.getWeek(), testWeekArray);
    }
    @Test
    void testPreviousWeek() {
        Week week = Week.getInstance(true);
        int n_weeks_backward = 1;
        do{
            week.previousWeek();
            n_weeks_backward--;
```

```
        }while (n_weeks_backward != 0);
        //settare testWeekArray con valori
          della n-esima settimana precedente
        int[] testWeekArray = {29, 30, 31, 1, 2, 3, 4};
        assertArrayEquals(week.getWeek(), testWeekArray);
    }
}
```

Unit test classe Week

```
package com.example.demo;
import org.junit.jupiter.api.Test;
import java.sql.*;

import static org.junit.jupiter.api.Assertions.*;

class LoginControllerTest {

    @Test
    void testCheckDB() throws SQLException {
        Usr_LoginController loginController = new Usr_LoginController();
        Connection connection = DriverManager.getConnection
          ("jdbc:mysql://localhost:3306/questura", "root", "");
        Statement statement = connection.createStatement();
        ResultSet res = statement.executeQuery
          ("SELECT cod_fiscale, password FROM login");
        String c_fisc = "a";
        String password = "123456";
        assertTrue(Usr_LoginController.checkDB(res, c_fisc, password));
    }
}
```

Unit test correttezza dati nel DB

4.5 Test degli sviluppatori

La fase di testing ha compreso sia testing individuali, sia testing fatti in gruppo per cercare di trovare situazioni che mettessero in difficoltà il sistema. Alcuni dei test significativi svolti sono: Verifiche di correttezza di registrazione o autenticazione Verifiche del comportamento del sistema in caso di inserimento di dati errati o assurdi Verifiche di corretta visualizzazione da lato cittadino delle prenotazioni aggiunte da lato questura e viceversa controllo della visualizzazione da lato questura delle prenotazioni occupate. Verifica della corretta visualizzazione delle disponibilità relative alla sede selezionata Verifica del corretto funzionamento del calendario per evitare errori in fase di cambio settimana Verifica della corretta gestione di tentativi di prenotare in orari non disponibili o già occupati

4.6 Test utente generico

Il software è stato infine sottoposto a test da parte di individui esterni allo sviluppo del progetto, al fine di valutarne la semplicità d'uso e l'intuitività dal punto di vista di un effettivo possibile utente finale. Si è cercato di limitare al minimo le informazioni riguardo al funzionamento del progetto, descrivendolo solo come una piattaforma a cui cittadini e dipendenti di questure possono accedere per gestire prenotazioni di passaporti, al fine di dare un contesto di partenza all'utente senza influenzarne l'esperienza. Da questo test non sono emersi particolari problemi di funzionamento, ma piuttosto utili accorgimenti per rendere l'esperienza d'uso più fluida ed intuitiva e che hanno poi portato al miglioramento di alcune funzionalità, principalmente da un punto di vista grafico.

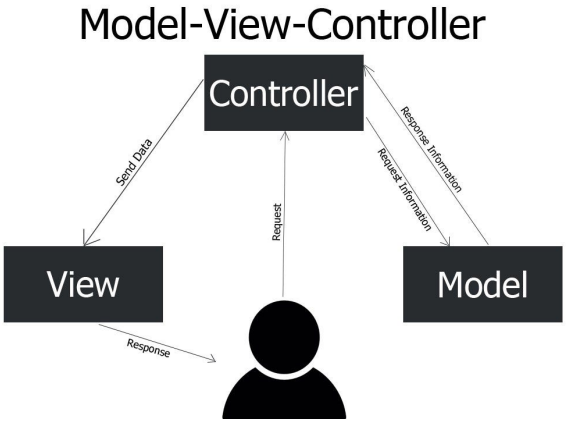
5 Pattern

5.1 Pattern architetturali

Il pattern architetturale **MVC** (Model-View-Controller) è un modello di progettazione software comunemente utilizzato per sviluppare interfacce utente che suddivide la logica del programma in tre elementi interconnessi:

- Modello (Model): Rappresenta il backend che contiene tutta la logica dei dati.
- Vista (View): Rappresenta il frontend o l'interfaccia grafica utente (GUI).
- Controllore (Controller): È il cervello dell'applicazione che controlla come i dati vengono visualizzati

I suoi vantaggi sono la chiarezza del codice data dalla suddivisione dei compiti, la manutenzione facilitata e la maggiore scalabilità del sistema. Inoltre, nel nostro sistema, la parte di Model, che gestisce i dati, è stata collegata ad appositi Database.



Modello MVC

Il pattern architetturale **Repository** è un modello di progettazione software che agisce come un intermediario tra la logica dell'applicazione e lo storage dei dati. Infatti, i componenti del sistema non interagiscono direttamente tra di loro, bensì chiedono al repository di trovare, salvare, aggiornare o eliminare i dati di cui ha bisogno. Questa separazione favorisce la modularità, la manutenibilità e la testabilità.

5.2 Pattern progettuali

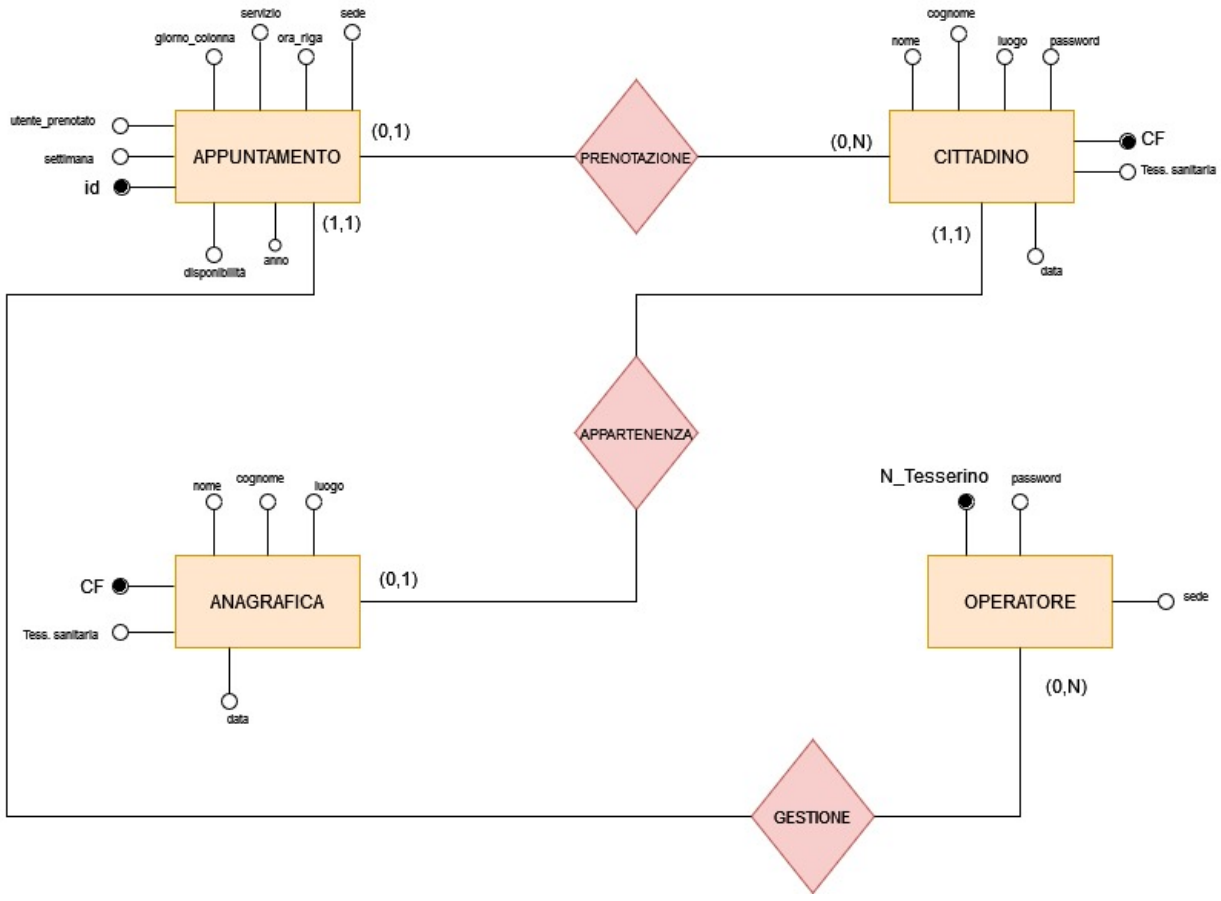
Il pattern creazionale **Singleton** è un modello di progettazione software che assicura che una classe abbia solo una istanza e fornisce un punto di accesso globale a essa. Questo pattern è particolarmente utile quando esattamente un oggetto è necessario per coordinare le azioni in tutto il sistema e viene usato solitamente se si desidera controllare l'accesso a risorse condivise, come connessioni al database. Nel nostro caso è stato usato nella classe `WEEK`, in modo tale da avere un unico accesso globale.

Il pattern comportamentale **Iterator**, comunemente usato nel linguaggio Java, permette di accedere sequenzialmente ad una collezione di dati senza esporre la sua implementazione. Nel sistema, il pattern viene usato continuamente per accedere ai dati dei Database scorrendoli sequenzialmente e, contemporaneamente, tiene privata la struttura interna all'utente. I vantaggi sono la pulizia di codice, in quanto non viene rappresentata la struttura della collezione e l'accesso ai dati in modo sequenziale.

6 Database

Per un maggiore controllo della parte dei dati del sistema, si è deciso fin da subito di implementare una struttura bsata sull'utilizzo di un Database. In particolare, il sistema interagisce con quattro tabelle diverse, ognuna dedicata al salvataggio di un tipo di dato:

- Anagrafica: contiene i dati dei cittadini NON ancora registrati.
- Cittadino: contiene i dati dei soli cittadini che hanno effettuato con successo la registrazione.
- Operatore: contiene i dati di tutti gli operatori delle varie questure.
- Appuntamento: contiene i dati di tutti gli appuntamenti validi prenotati dai cittadini, con informazioni riguardo giorno, servizio e cittadino.



Schema ER della rete di DataBase del sistema.