

# Tatum Vaccini

## Documentazione tecnica

Rossi Giorgio	746571	VA
Garegnani Federico	746789	VA
Canali Luca	744802	VA
Callegari Pietro	746568	VA
Invernizzi Daniele	746484	VA

3 febbraio 2023

## Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Librerie esterne utilizzate . . . . .	2
1.2	Struttura generale delle classi . . . . .	2
<b>2</b>	<b>Core classes</b>	<b>2</b>
2.1	Classi generiche . . . . .	4
2.2	Interfacce . . . . .	5
2.3	Classi enumerative . . . . .	5
<b>3</b>	<b>UML Diagrams</b>	<b>6</b>

# 1 Introduzione

Tatum Vaccini è un progetto sviluppato nell'ambito del progetto di Laboratorio A per il corso di laurea in Informatica dell'Università degli Studi dell'Insubria. Il progetto è sviluppato in Java 19, usa un'interfaccia grafica costruita con OpenJFx 19ed è stato sviluppato e testato sui sistemi operativi Windows 10, Windows 11 e Mac OS Ventura.

Esso è costituito da 3 applicazioni distinte: *Applicazione Centri Vaccinali*, *Applicazione cittadini* e *Applicazione server*.

## 1.1 Librerie esterne utilizzate

L'unica libreria esterna di cui l'applicazione fa uso è OpenJFx 12, la quale contiene tutti gli elementi necessari allo sviluppo dell'interfaccia grafica. Si è reso necessario appoggiarsi a una libreria esterna in quanto, a partire dalla release 9 del linguaggio Java, JavaFx non è più inclusa all'interno del Java Development Kit (JDK) distribuito da Oracle.

## 1.2 Struttura generale delle classi

Il progetto è organizzato in quattro moduli: *modulo server*, *modulo centri vaccinali*, *modulo cittadini* e *modulo common*. La struttura generale è riportata in Figura 1. Descriveremo nel seguito le classi di cui si compone ogni modulo e come esse sono organizzate in package.

**Applicazione server** L'applicazione server rappresenta il cuore dell'intero progetto, ad essa si collegano in remoto le altre applicazioni e gestisce inoltre l'interazione con il DBMS.

**Applicazione centri vaccinali** Il modulo centri vaccinali è la parte del programma dedicata agli operatori sanitari, essa permette di registrare nuovi centri vaccinali ed inserire vaccinazioni.

**Applicazione cittadini** Il modulo cittadini è infine l'applicazione orientata all'uso da parte dei pazienti, essa permette la registrazione e l'inserimento di eventuali evento avversi riscontrati in seguito alla vaccinazione.

**Modulo common** Questo modulo ha il solo scopo di fungere da *contenitore* delle classi condivise dall'intero progetto. Esso non è pertanto un'applicazione avviabile ma solamente un modulo verso cui tutti gli altri moduli hanno dipendenze.

# 2 Core classes

Di seguito viene trattata nel dettaglio ogni classe definita all'interno del codice.

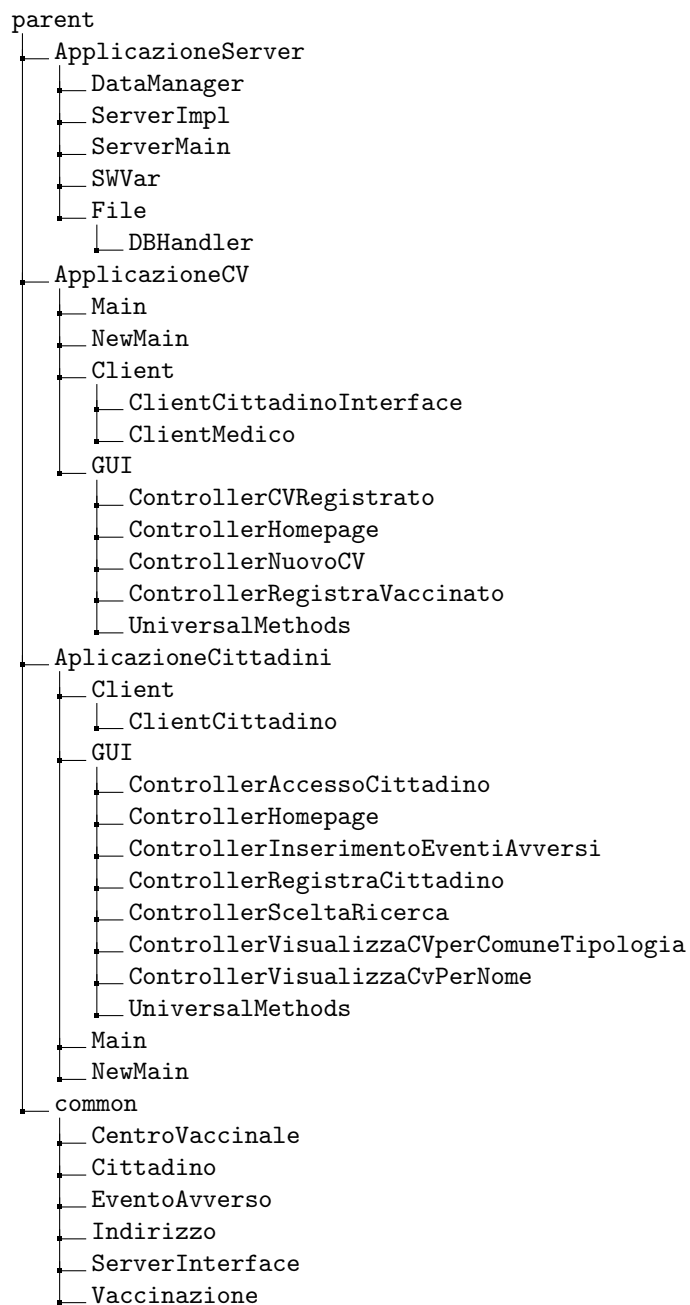


Figura 1: Gerarchia delle classi del progetto

## 2.1 Classi generiche

**ServerImpl** La classe `ServerImpl` si occupa dell'interazione del server con i client attraverso RMI, ovvero l'insieme di procedure fornite da Java che permettono a due processi distribuiti di comunicare tra loro. La comunicazione avviene grazie alle interfacce `ServIntCentroVaccinale` e `ServIntCittadino` che la classe implementa e che le permettono di fornire alcuni metodi invocabili dai client.

Sono presenti quattro campi, due sono costanti contenenti i numeri di porta sui quali verranno pubblicati il registro e l'applicazione stessa, mentre i rimanenti sono due variabili di tipo `List<>`; esse sono inizializzate nel costruttore come `LinkedList<>` e i loro elementi andranno a memorizzare i riferimenti remoti dei client. In questo caso si è scelta una `LinkedList` in quanto sono più frequenti le operazioni di aggiunta e rimozione di elementi rispetto a quelle di ricerca di un elemento specifico.

All'avvio del programma la prima classe ad essere istanziata è proprio `ServerImpl`, il costruttore si occupa di creare il registro e registrarlo sulla porta 1099, generare un riferimento remoto dell'oggetto e pubblicarlo all'interno del registro; oltre a inizializzare i due campi di tipo `List<>`.

**DataManager** Questa classe rappresenta un gestore di dati per l'applicazione. La classe contiene metodi per registrare un centro vaccinale, un cittadino e una vaccinazione. Ogni volta che viene chiamato uno di questi metodi, viene eseguita una query sul database per inserire i dati necessari. La classe utilizza un design pattern singleton, il che significa che verrà creata una sola istanza di `DataManager` durante l'esecuzione dell'applicazione. Questo viene fatto utilizzando un metodo factory, `getInstance()`, che crea un'istanza solo se non esiste già.

**SWVar** La classe contiene alcune variabili costanti che rappresentano i nomi delle tabelle in un database. Essa è sfruttata nelle classi `DBHandler` e `DataManager` per tenere traccia dei nomi delle tabelle.

**DBHandler** La classe `DBHandler` rappresenta un gestore di database. Utilizza la libreria JDBC per la connessione a un database PostgreSQL. La classe ha metodi per connettersi e disconnettersi dal database, creare tabelle, inserire dati nelle tabelle e inizializzare il database (se non esiste). La classe utilizza il pattern Singleton per garantire che ci sia una sola istanza della connessione al database.

**CentroVaccinale** Definisce un centro vaccinale, si compone di tre campi: nome, indirizzo, tipologia.

**Cittadino** La classe `Cittadino` è pensata per rappresentare una persona che si registri all'interno dell'applicazione per essere vaccinata. Sono definiti da sette campi: id, nome, cognome, cf, email, psw, nomeCentroVaccinale.

**EventoAvverso** EventoAvverso è la classe pensata per modellizzare eventuali effetti collaterali del vaccino; ogni istanza di questa classe rappresenta una particolare complicazione scelta tra sei, con la relativa intensità riscontrata dal paziente ed è accompagnata da un'eventuale nota testuale.

**Indirizzo** Classe di tipo record che ingloba al suo interno tutti i campi di cui si compone un indirizzo. I campi sono in totale sette

- **id** identificativo univoco associato all'indirizzo assegnato automaticamente dal DBMS
- **identificatore** riferimento a un oggetto della classe enumerativa **Identificatore**
- **nome**, **comune**, **provincia**, **ZIP** campi di tipo stringa per memorizzare rispettivamente il nome della via/viale/piazza, il comune, la provincia (2 lettere), lo ZIP code (ossia il CAP, 5 numeri)
- **numCivico** parametro di tipo short per il numero civico

La classe prevede due costruttori, il primo definito di default che accetta sette parametri, ognuno corrispondente in tipo al campo della classe, mentre il secondo non prevede il parametro per l'id, che viene assegnato di default a 0 ed inoltre richiede un oggetto **String** come identificatore.

**Vaccinazione** La classe Vaccinazione vuole modellizzare un'avvenuta vaccinazione ed è definita dai seguenti quattro campi: **id**, **data**, **v**, **codicePrenotazione**.

- **Id** è un numero progressivo che identifica automaticamente la vaccinazione ed è assegnato automaticamente dal DBMS
- **data** è un campo di tipo **LocalDate** in cui viene memorizzato il giorno in cui è stata effettuata la vaccinazione (di default la data odierna)
- **v** è un riferimento al tipo enumerativo **Vaccino** e indica quale vaccino tra quelli disponibili è stato somministrato
- **codicePrenotazione** è una stringa contenente il codice della prenotazione assegnato al cittadino al momento della registrazione

## 2.2 Interfacce

**ServerInterface** Questa interfaccia rappresenta un'interfaccia remota del server. Essa è implementata dalla classe **ServerImpl** e definisce i metodi che il server rende disponibili ai proprio client.

## 2.3 Classi enumerative

**Tipologia** Si tratta di una classe enumerativa molto semplice, volta a memorizzare le possibili tipologie di centro vaccinale: ospedaliero, aziendale, hub.

**QualeEvento** Classe enumerativa definita all'interno della classe **EventoAvverso** in cui sono definiti un totale di sei sintomi che si possono riscontrare in seguito a una vaccinazione. I sintomi definiti sono: **mal di testa**, **febbre**, **dolori muscolari e articolari**, **linfadenopatia**, **tachicardia**, **crisi ipertensiva**.

**Identificatore** Classe enumerativa definita all'interno del record **Indirizzo** che definisce gli identificatori dell'indirizzo che è possibile scegliere. I valori presenti sono: **via**, **viale**, **piazza**.

**Vaccino** Classe enumerativa contenente in nomi dei vaccini approvati dall'Agenzia Europea per i Medicinali e che è quindi possibile somministrare. Al 12/12/2021 i vaccini approvati sono quattro: Pfizer, Moderna, AstraZeneca, Johnson & Johnson.

### 3 UML Diagrams

Sono riportati di seguito i diagrammi UML più significativi al fine di comprendere la struttura di ogni applicazione.

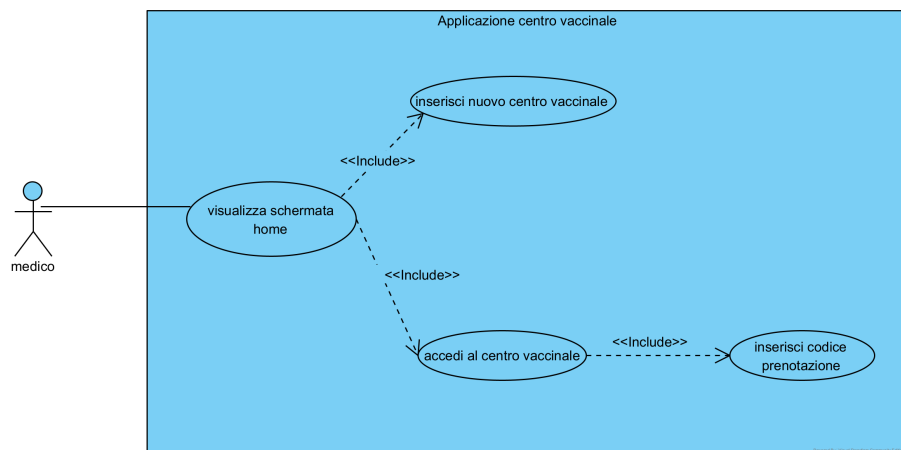


Figura 2: Use case diagram del modulo centri vaccinali

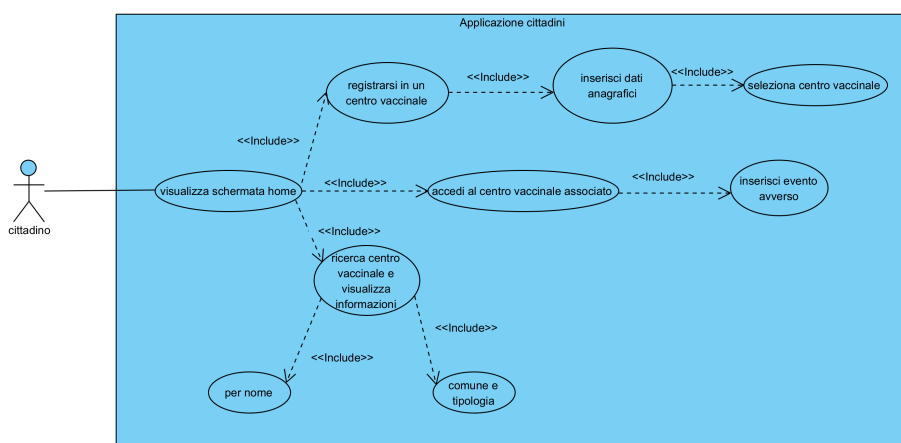


Figura 3: Use case diagram del modulo cittadino

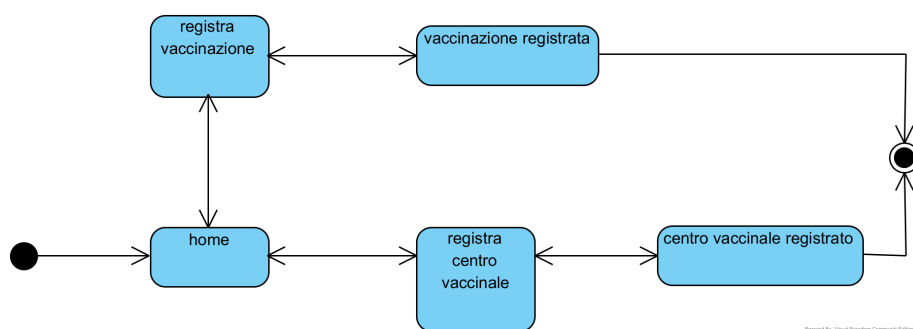


Figura 4: State diagram del modulo centro vaccinale

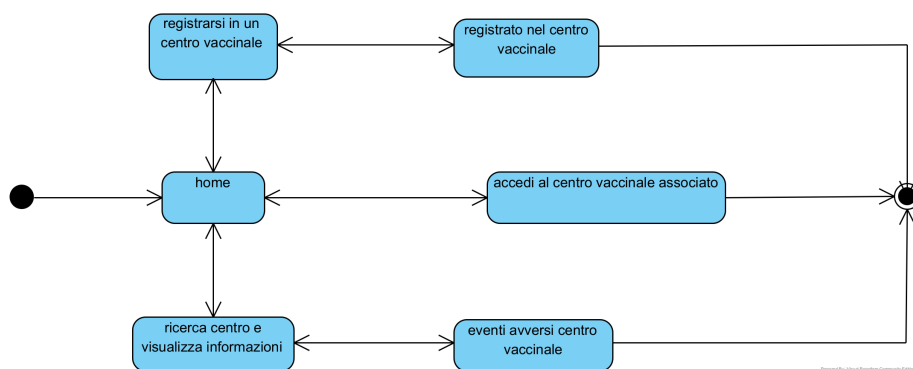


Figura 5: State diagram del modulo cittadino



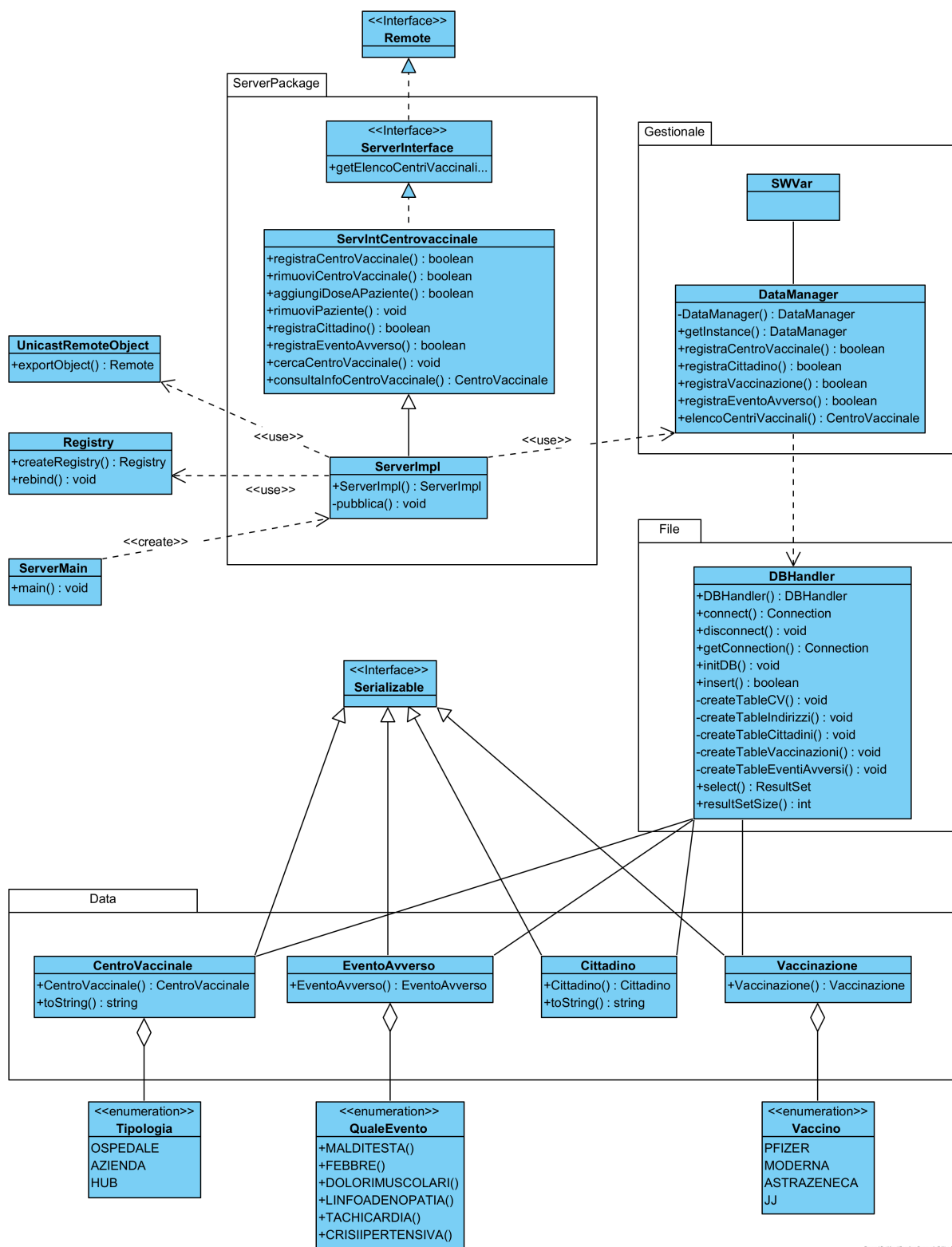


Figura 6: Class Diagram del modulo Server