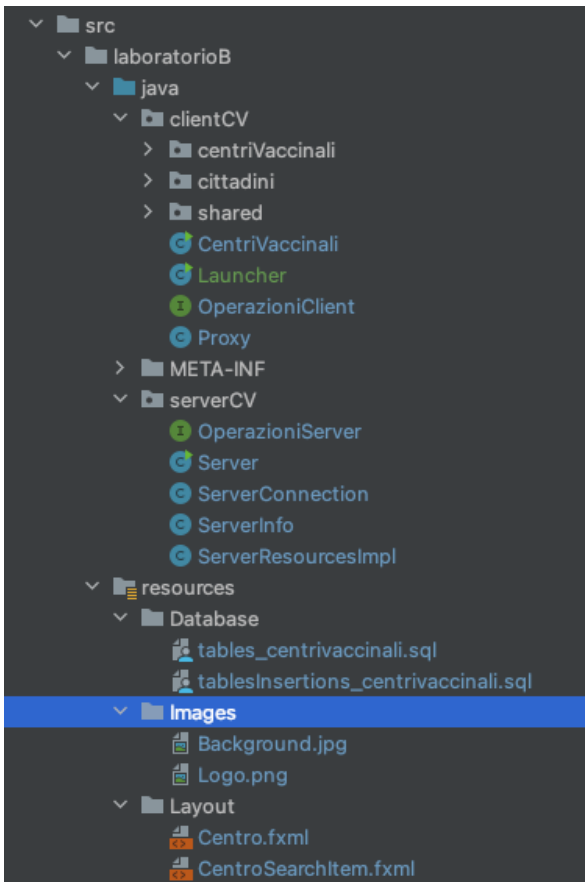
FERNANDO MARCELO EDELSTEIN FERNANDEZ
ELIANA MONTELEONE

CENTRI VACCINALI

MANUALE TECNICO

INDICE

STRUTTURA



L'applicazione consiste in due packages: "clientCV" e "serverCV" che contengono tutti gli elementi che compongono i lati Client e Server. È stato scelto di utilizzare JavaFX per la creazione dei layout, per cui dentro cartella "resources" abbiamo la cartella "Layout" dove troveremo tutti i file fxml, la cartella "Images", che contiene l'immagine di sottofondo ed il logo e per ultimo, la cartella "Database" dove troveremo i files che creano e riempiono le tabelle del DB.

ServerCV

Dentro questo package troveremo tutte le classi e l'interfaccia che compongono il lato Server.

Queste sono:

- OperazioniServer
- Server
- ServerConnection
- ServerInfo
- ServerResourcesImpl

ClientCV

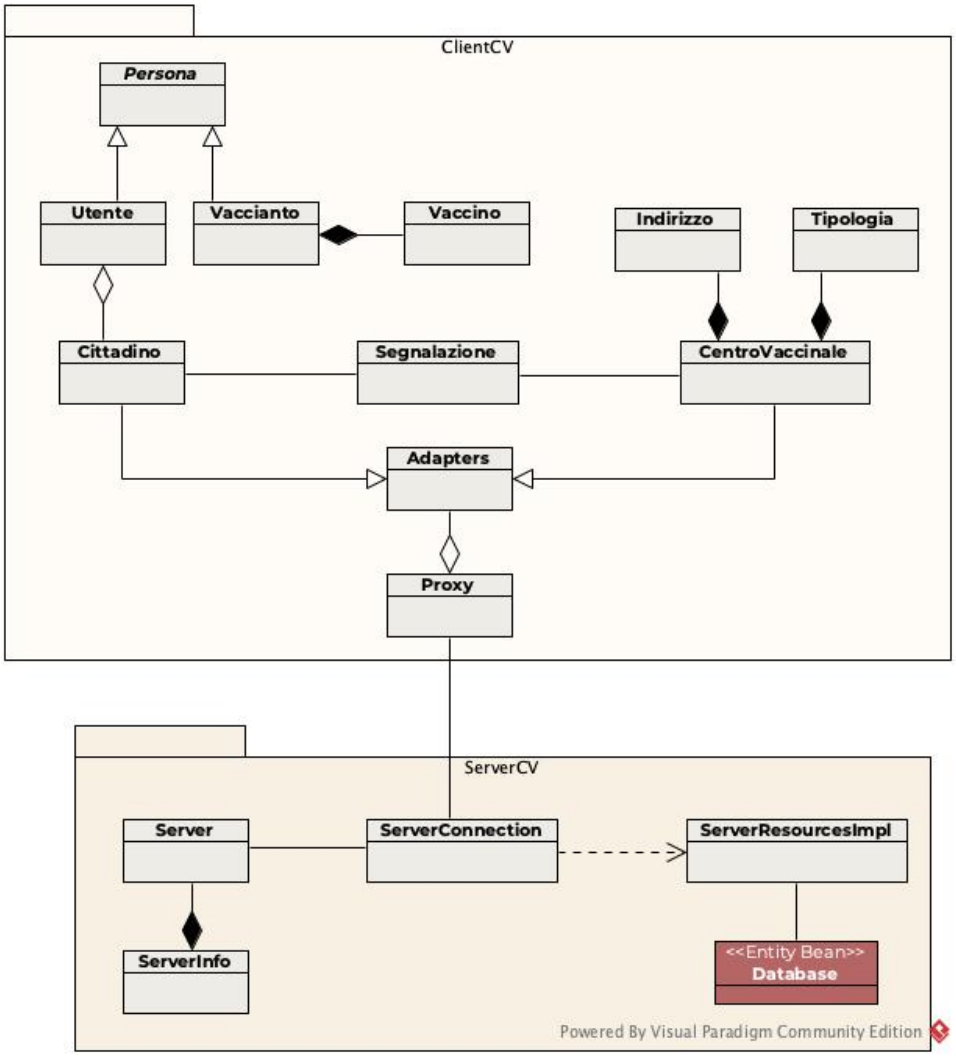
Dentro questo package troveremo altri sottopackages:

- clientCV: Dentro il package si trova il launcher dell'applicazione, l'interfaccia OperazioniClient e il Proxy
- CentriVaccinali : Contengono tutti gli adapters per l'applicazione
- Cittadini : Contiene il modello Cittadino
- Shared : Contiene la classe Check dove si trovano dei metodi di controllo, alcuni modelli e la classe Server Info, che contiene le informazioni per la connessione al Server

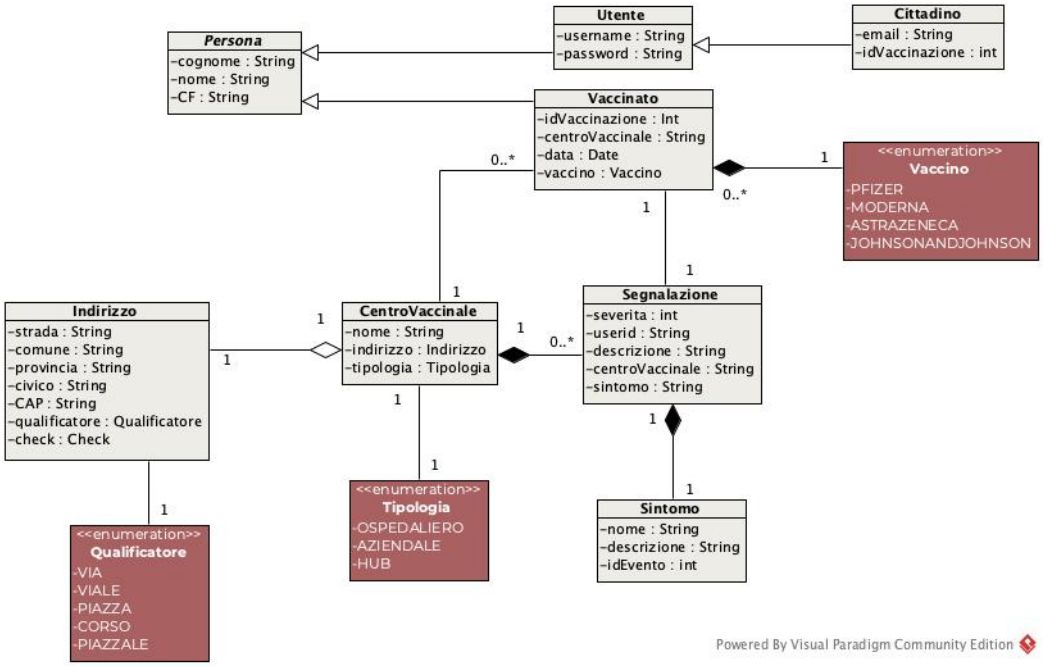
PROGETTAZIONE

APPLICAZIONE

Organizzazione generale



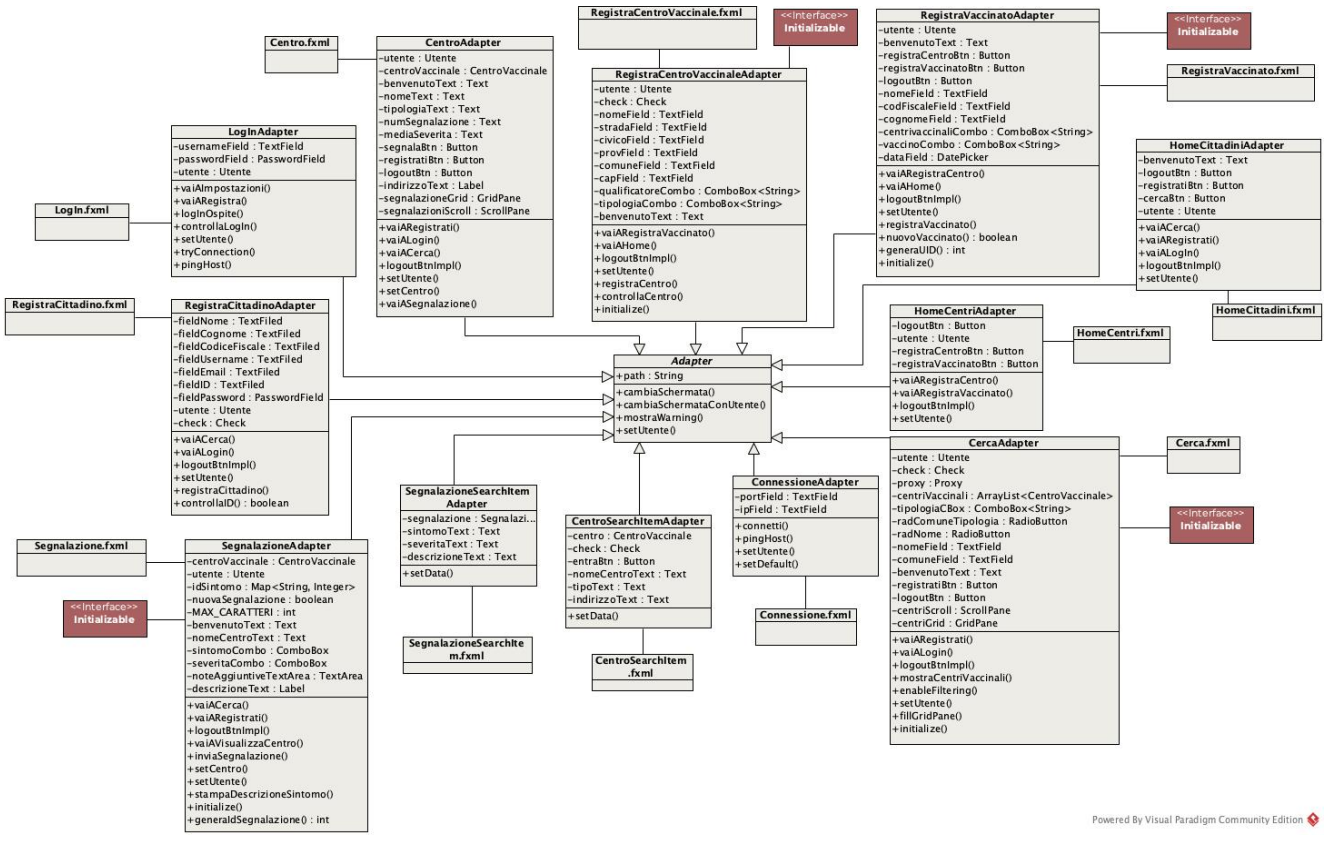
Modelli



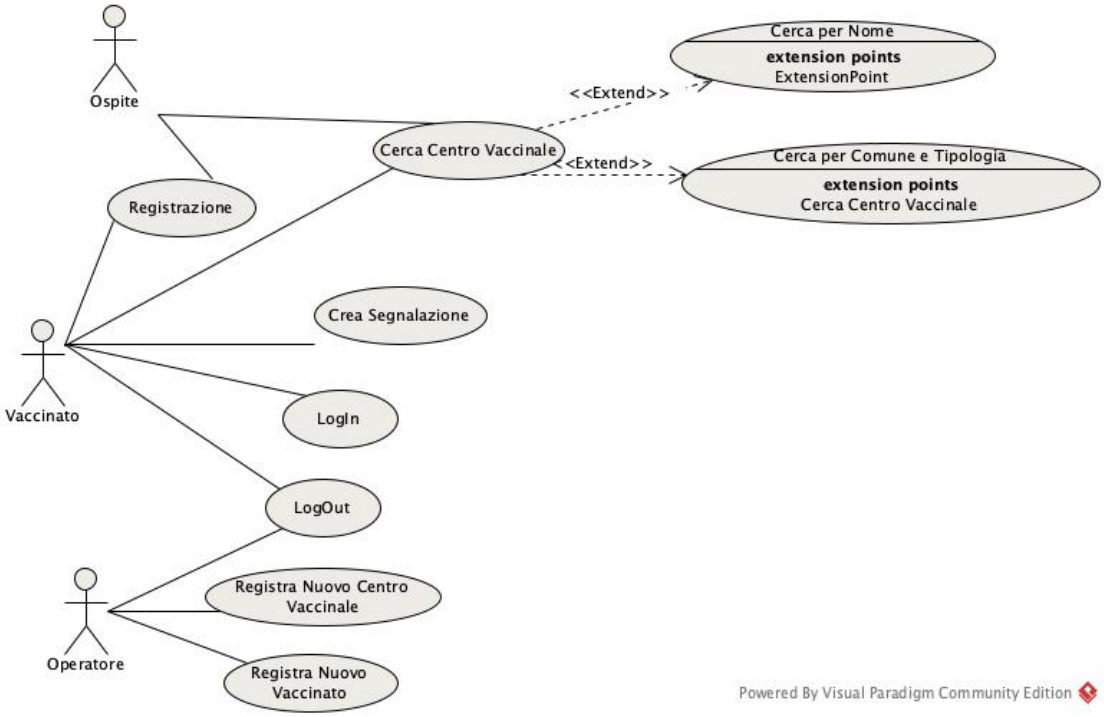
I modelli che vengono salvati e scaricati dalla base di dati. Possiamo trovarli all'interno del package clientCV.

Adapter

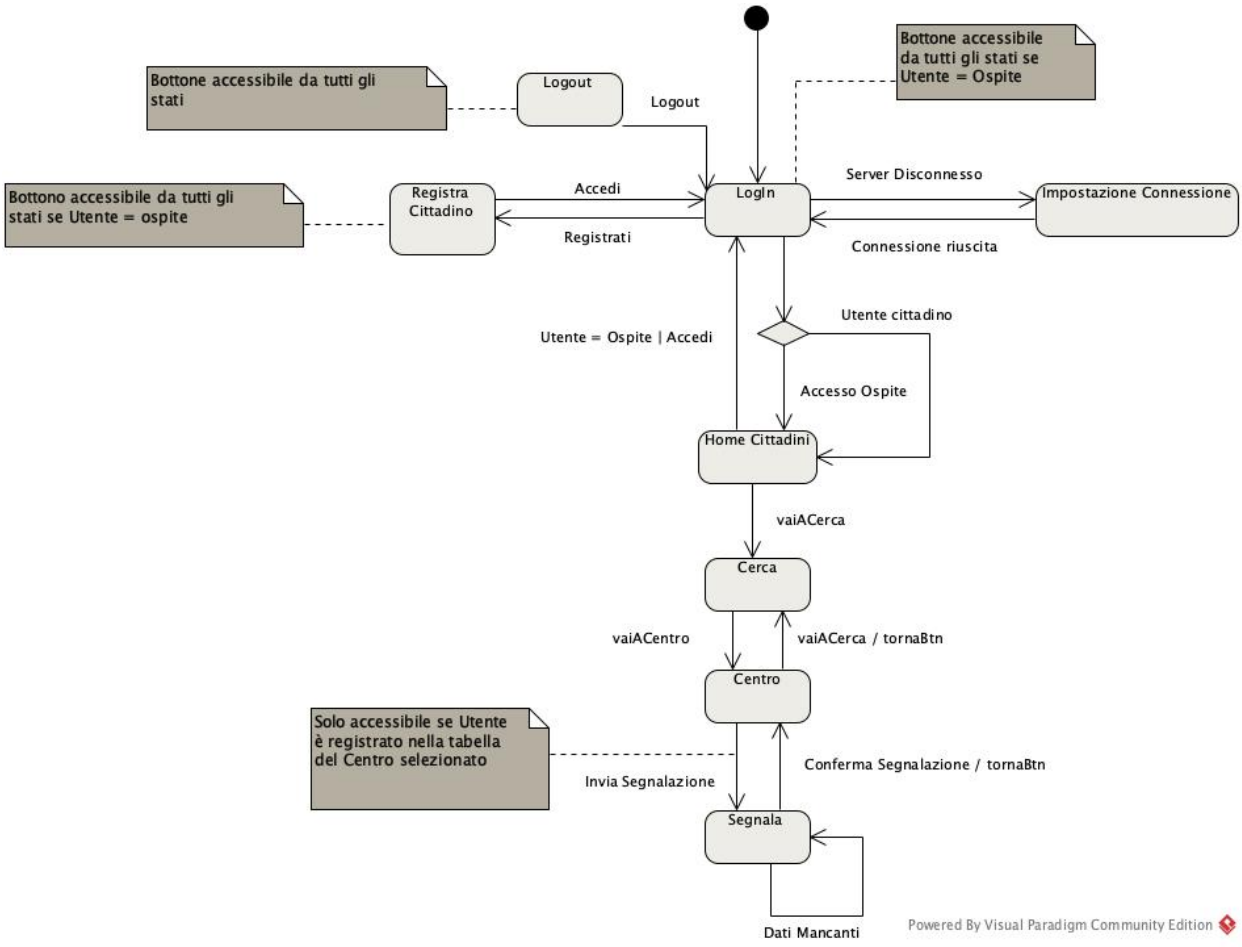
Per l'interfaccia grafica, è stato utilizzato JavaFx, per cui, ogni schermata è stata sviluppata in un file fxml associato ad ogni classe Adapter. Oltre a tutti gli adapter specifici delle schermate, abbiamo la classe astratta Adapter che contiene dei metodi che servono a tutte le altre classi.



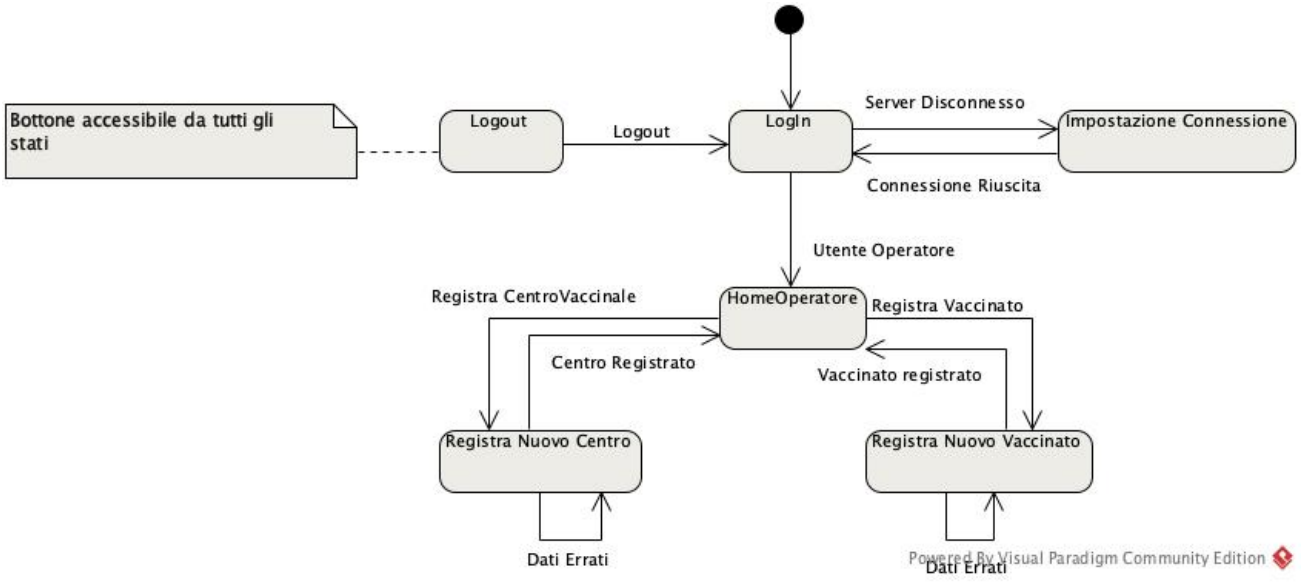
Operazioni per i diversi Utenti



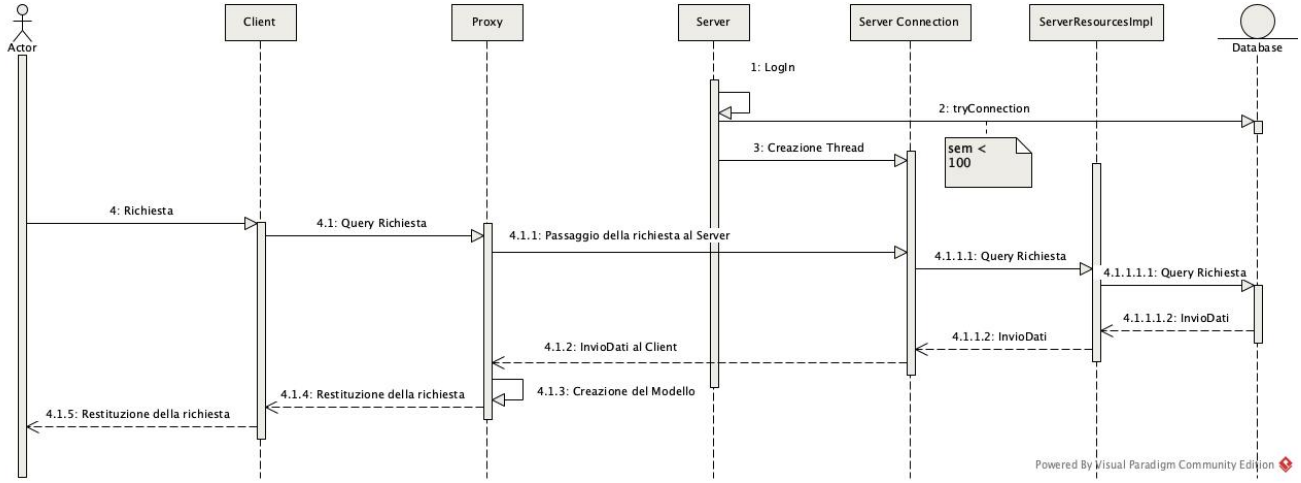
Stati degli Cittadini



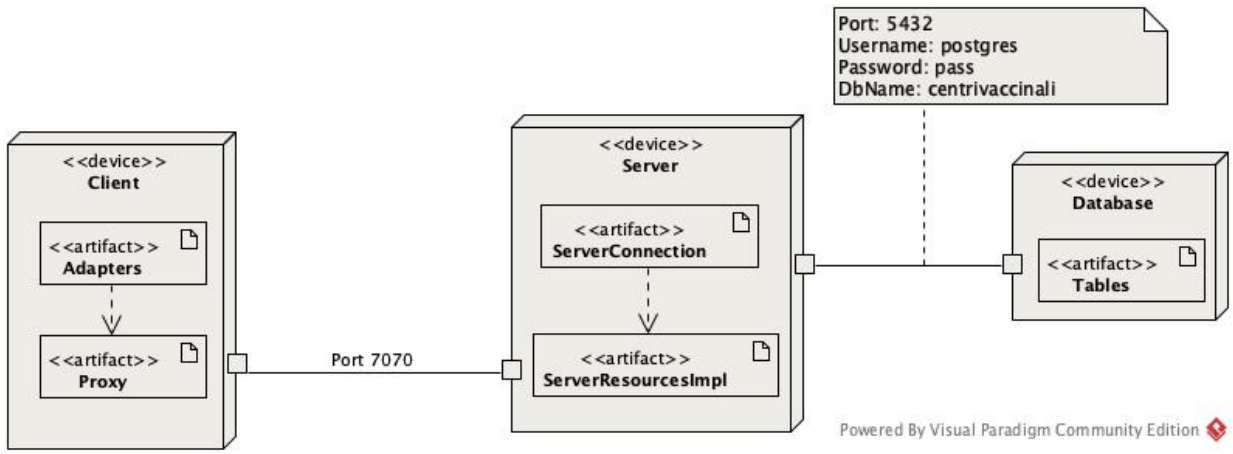
Stati degli operatori



Passaggio dei comandi dal Client al Server



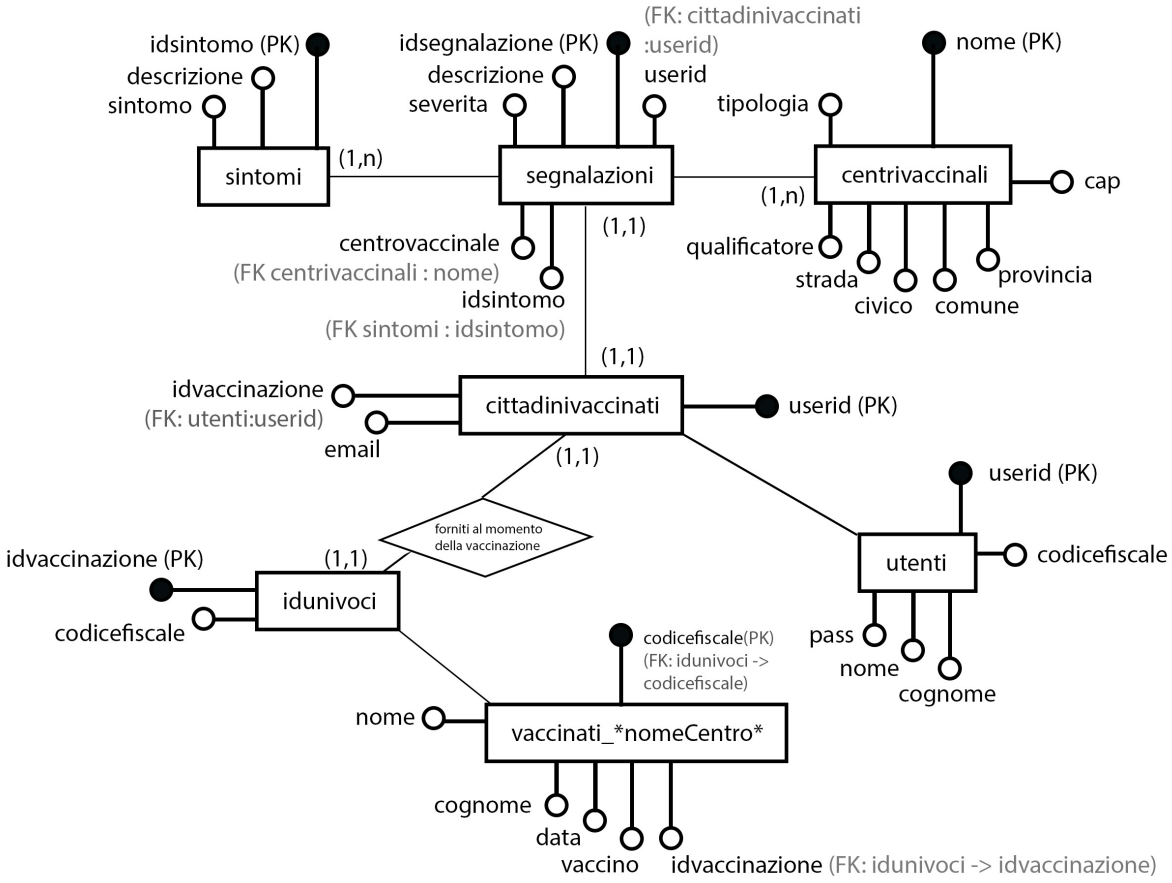
Deployment Diagram



DATABASE

Le tabelle che conformano il database sono le seguenti:

- sintomi
- segnalazioni
- centrivaccinali
- cittadinivaccinati
- idunivoci
- utenti
- vaccinati_*nomeCentro*



CENTRIVACCINALI

Su questa tabella vengono memorizzati tutti i centri vaccinali registrati, avendo come Primary Key il nome per evitare che il centro sia registrato due volte.

UTENTI

Su questa tabella vengono memorizzati tutti gli utenti identificati dall'userid. Gli utenti che non compaiono sulla tabella cittadinivaccinati, sono Operatori.

IDUNIVOCI

Su questa tabella vengono registrati i cittadini vaccinati. Si salvano gli id univoci della vaccinazione che vengono confrontati in fase di registrazione per evitare duplicati. L'applicazione genera un id unico della vaccinazione e lo accoppia ad un codice fiscale (unico di ogni persona). Il nuovo id viene utilizzato come Foreign Key per le tabelle vaccinati_*nomeCentro* per tenere traccia del centro in cui il cittadino è stato registrato.

SINTOMI

Questa tabella è stata creata per avere tutti i possibili sintomi elencati sul db in modo tale che se ci fosse il bisogno di aggiungere altri eventi avversi, questi possano essere inseriti nel db. In questo modo, l'utente non deve scrivere manualmente gli eventuali effetti collaterali e si evitano degli errori di ortografia e duplicati.

SEGNALAZIONI

Questa tabella contiene tutte le segnalazioni dei diversi centri vaccinali. Al suo interno troviamo le colonne "centrovaccinale" Foreign Key della tabella centrivaccinali, "idsintomo" Foreign Key della tabella sintomi, "userid" Foreign Key della tabella cittadinivaccinati.

VACCINATI_*NOMECENTRO*

Queste tabelle vengono generate per ogni centro ed al suo interno contengono i dati rispettivi alle vaccinazioni dei cittadini che sono stati registrati al centro, in questo modo si evita che un cittadino possa creare una segnalazione per un centro nel quale non è stato vaccinato.

DESIGN PATTERN

Basandoci sulla progettazione Android, per l'interfaccia grafica dell'applicazione Client, è stato utilizzato il pattern Model Control View. Ogni interfaccia fxml è associata ad un Adapter che contiene i metodi utili per la schermata corrente.

La connessione tra Server e Client è stata sviluppata sfruttando il Proxy pattern. Questa classe è quella incaricata di fare tutte le chiamate al server e ricevere i dati creando i modelli che servono al Client.

ARCHITETTURA

L'applicazione è compatibile con tutti i 3 sistemi operativi : Windows, Mac (architettura x86, può essere eseguito su architettura M1 su intelliJ con la versione beta di JavaFx 17) e Linux. Essa utilizza JavaFx e il driver JDBC.

L'applicazione ha come operazione più complessa le richieste al database, queste vengono inserite in strutture dati come ArrayList per la facile gestione di essi. Quando il server viene avviato, crea un semaforo con capacità di 100 (Limite di connessioni di Postgresql) evitando così il problema della concorrenza.

CONCLUSIONI

L'applicazione ha due files .Jar, uno per lanciare il server e l'altro per lanciare il client. Per semplicità, entrambi i file sono nella cartella target da cui con le seguenti linee di comando si possono lanciare entrambe le applicazioni.

```
java -jar Server.java

java -jar LaboratorioB.java
```