



# Centro Vaccinale

L'ITALIA RINASCE CON UN FIORE

## MANUALE TECNICO

---

BANCORA Davide

743662      Como

---

CASALNOVO Giacomo

740003      Como

---

DONATO Benedetta

742957      Como

---

DUBINI Emanuele

740954      Como

---

# Indice

03	<b>Benvenuto in “Centro Vaccinale”</b>
03	Struttura del progetto
03	Tecnologie utilizzate
04	<b>Documentazione ServerCV</b>
05	<b>Documentazione Common</b>
05	Classe CentroVaccinale
05	Classe Indirizzo
05	Classe ProgUtili
05	Package CFGenerator
05	Classe RegistryViewer
06	Classe Enumerativa Qualificatore
06	Classe Enumerativa TipologiaCV
07	Classe Enumerativa Vaccino
08	<b>Documentazione CentriVaccinali</b>
09	<b>Documentazione Database</b>
09	Package GenerateDataLib
09	Classe DataModel
09	Classe DBHelper
10	Classe QueryDebug

Appendice A

11      **Mappa delle Classi**

---

# Benvenuto in “Centro Vaccinale”

**CentroVaccinale** è un software che offre un sistema per la gestione di Cittadini e Centri Vaccinali per la pandemia Covid-19.

**Programma CentriVaccinali:** <https://github.com/astargisco/CentroVaccinale>

**Bacheca Agile/Kanban:** <https://trello.com/b/TdL756CN/centrovaccinaleboard>

## Struttura del progetto

**CentroVaccinale** è il miglior programma di inserimento e ricerca di Cittadini e Centri Vaccinali ed è composto da due potenti applicazioni in ambiente client/server:

- ServerCV
- ClientCV

Il progetto rappresenta un sistema di segnalazione di eventi avversi in seguito a vaccinazione Covid19.

## Tecnologie Utilizzate

L'applicazione è interamente scritta in linguaggio Java e si appoggia su un database relazionale PostGres SQL per la sua implementazione e su JDBC per l'accesso ai dati.

Per gestire la concorrenza è stata usata la tecnologia Java, RMI (*Remote Method Invocation*), che consente a processi Java distribuiti di comunicare attraverso una rete.

Uso di REGEX per la verifica di tutti i campi di inserimento *client side*.

E' stato usato Maven per fornire i file JAR eseguibili e *cross platform*.

Grazie al seguente plugin configurato nel file pom.xml, siamo riusciti a includere tutte le dipendenze (incluso JavaFX e PostgresSQL) per eseguire il nostro progetto sulle maggiori piattaforme.Tale plugin si occupa di costruire il JAR con tutte le dipendenze di cui abbiamo bisogno, tra cui:

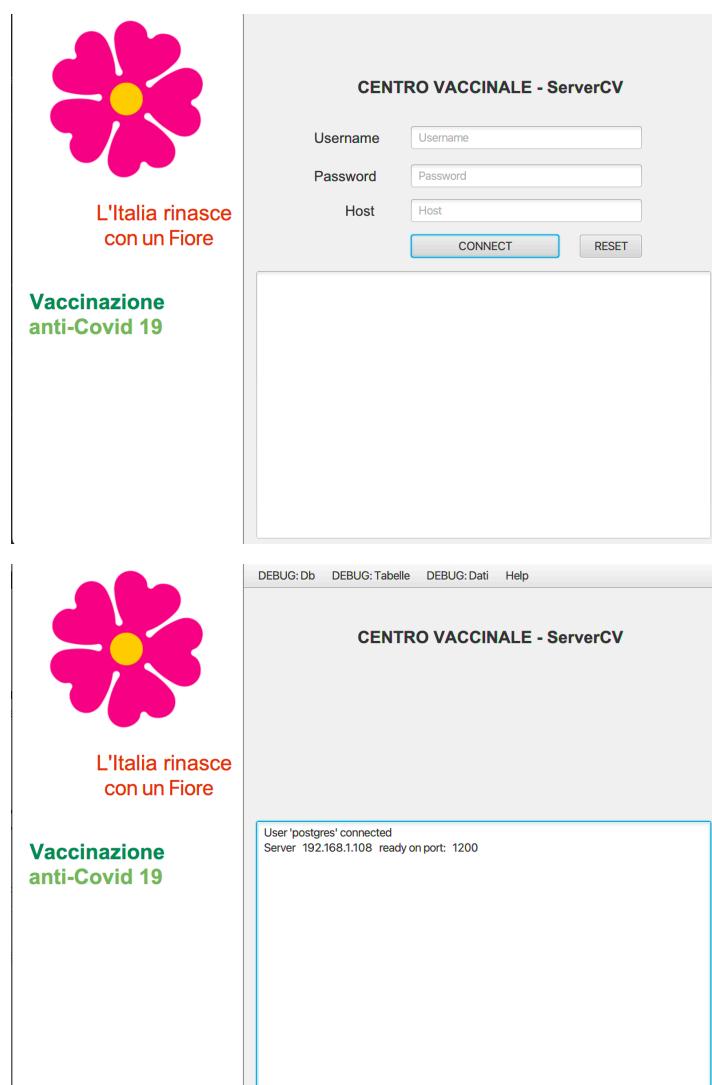
- JavaFX nei moduli FXML
- JavaFX Controllers
- MVC

# Documentazione ServerCV

Questa classe rappresenta l'inizio del software "CentroVaccinale", tramite la quale viene avviato il server che gestisce le connessioni remote e il database.

Crea i servizi server per poter effettuare le connessioni dei vari Client fornendo le interfacce dei metodi remoti che vengono invocati dai Clients.

Per fornire i suoi servizi, ServerCV deve collegarsi al DB tramite username/password, questi per default sono *postgres/postgres*. L'host a cui collegarsi è *localhost*.



---

# Documentazione Common

In questo package sono state create le classi che permettono l'utilizzo di classi e metodi comuni a entrambe le applicazioni ed alcuni metodi utili nel programma.

## Classe *CentroVaccinale*

Classe per costruire gli oggetti Centri Vaccinali

## Classe *Indirizzo*

Qui costruiamo l'indirizzo di ogni centro vaccinale usando qualificatori differenti (via/viale/piazza/etc..) in aggiunta ai classici dati nome, numero civico, comune, provincia e cap.

L'indirizzo viene ricostruito nelle relazioni dei DB.

## Classe *ProgUtili*

In questa classe sono presenti alcuni metodi utili come ad esempio il metodo che stampa la data:

```
public static String stampaData() {
    Date oggi = new Date(); // Data di oggi
    SimpleDateFormat sdf = new SimpleDateFormat("dd MMMM yyyy");
    return sdf.format(oggi);
}
```

## Package *CFGenerator*

Qui vengono generati i codici fiscali utilizzando come base il codice erariale specifico di ogni comune.

## Classe *RegistryViewer*

Classe di servizio per verificare lato server se il *registry* è attivo e su quale porta. Questa classe è usata solo all'interno dell'IDE.

---

```
public class RegistryViewer {
    public static void main(String[] args)
    {
        final String host = "localhost"; //arguments[0];
        int port = 1200;

        try
        {
            final Registry registry = LocateRegistry.getRegistry(host, port);
            final String[] boundNames = registry.list();

            for (final String name : boundNames)
            {
                System.out.println("RMI registry su " + host + " port " + port +
": " + name);
            }
        }
        catch (ConnectException connectEx)
        {
            System.err.println("ERRORE: Nessun registro RMI disponibile sulla
porta " + port);
        }
        catch (RemoteException remoteEx)
        {
            System.err.println("ERRORE: RemoteException");
        }
    }
}
```

## Classe enumerativa *Qualificatore*

Qui costruiamo il qualificatore di tipo dell'indirizzo:

```
public enum Qualificatore {
    Via,
    Viale,
    Piazza,
    Largo,
    Vicolo,
    Piazzale,
    Corso
}
```

## Classe enumerativa *TipologiaCV*

Qui costruiamo la tipologia dei centri vaccinali:

```
public enum TipologiaCV {
    Ospedaliero,
    Aziendale,
    Hub
}
```

---

## Classe enumerativa *Vaccino*

Qui costruiamo la tipologia del vaccino:

```
public enum Vaccino {  
    PFIZER,  
    ASTRAZENECA,  
    MODERNA,  
    JJ  
}
```

---

# Documentazione *CentriVaccinali*

Qui è contenuta tutta la parte relativa all'applicazione ClientCV e alla costruzione dell'oggetto cittadino.

```
public class Cittadino {  
    private String codiceFiscale; // PRIMARY KEY  
    private String cognome;  
    private String nome;  
    private int eta;  
    private String email;  
    private String userId;  
    private String password;  
    private String idVaccinazione;  
....
```

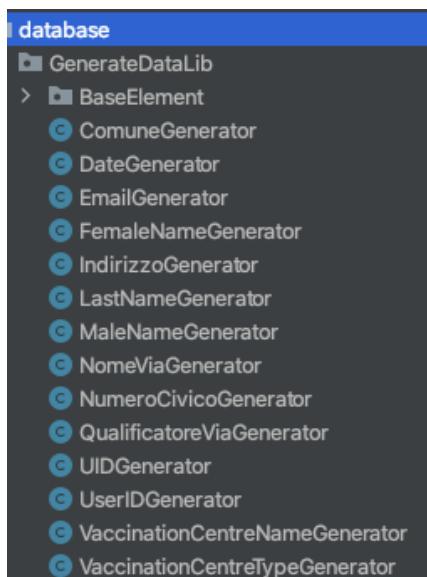
Per quanto riguarda la Graphical User Interface, nel package "gui" abbiamo inserito i controllers, mentre i file FXML e i CSS sono nel package "Resources".

# Documentazione *Database*

In questo package sono state create le classi che permettono l'utilizzo e le connessioni al database.

## Package *GenerateDataLib*

Qui sono contenute tutte le classi per la generazione automatica e casuale dei dati:



## Classe *DataModel*

In questa classe sono presenti i costrutti per la costruzione automatica dei cittadini e dei centri vaccinali, quando utilizziamo i metodi:

```
generateCittadino()     e     generateCentroVaccinale()
```

## Classe *DbHelper*

E' la classe necessaria per la connessione al Database: costruisce l'indirizzo di connessione e definisce lo username/passwd per collegarsi.

```
private final static String url = protocol + host + resource;
```

---

Fornisce inoltre i metodi per l'apertura e la chiusura della connessione.

```
public static Connection getConnection() throws SQLException {
    if(connection == null){
        connection = DriverManager.getConnection(url, username, password);
    }
    return connection;
}

public static void closeConnection() throws SQLException {
    connection.close();
    connection = null;
}
```

## Classe *QueryDebug*

In questa classe sono presenti le *query* per la creazione delle tabelle all'interno del database richiamabili attraverso il menu di *debug* dell'applicazione ServerCV. Queste *query* sono salvate come stringhe richiamabili staticamente attraverso gli appropriati metodi.

# **APPENDICE A**

## Mappa delle classi (generale)

Qui c'e' la mappa delle classi di tutto il codice. Disponibile come file separato per una più comoda visione.

