**INTRODUZIONE**

“Vacciniamo” è un progetto sviluppato per Laboratorio B per il corso di laurea in Informatica dell’università degli studi dell’Insubria.

Il progetto è sviluppato in Java 16, usa un’interfaccia grafica costruita con Swing ed è stato sviluppato e testato su:

* Window 10\11
* Mac Os X
* Linux

**STRUTTURA GENERALE DELLE CLASSI**

Il progetto è strutturato in due suddivisioni: **“ClientLab”**, che si occupa della gestione grafica e di elaborazione/visualizzazione dei dati lato client e **“ServerLab”** che svolge il medesimo compito lato server.

* **CLIENT LAB:**
  + Elaborazione e visualizzazione Dati:
    - ClientMain (main)
    - View <Interfaccia>
    - ViewImpl
    - Model <Interfaccia>
    - ModelImpl
    - ControllerImpl
    - Proxy
    - ServerInterface <Interfaccia>
  + Gestione Grafica:
    - SceltaOperatoreGUI
    - GUICambiaIP
    - OperatoreGUI
    - RegistraCentroGUI
    - RegistraVaccinatoGUI
    - CittadinoGUI
    - RegisterGUI
    - CercaCentroGUI
    - ElencoCentriGUI
    - LoginGUI
    - EventiAvversiGUI
    - VisualizzaEventiAvversiGUI
    - Colori
* **SERVER LAB**:
  + Elaborazione e visualizzazione dati:
    - ServerMain (main)
    - ServerWorker
    - ViewInterface <interfaccia>
    - ModelInterface <interfaccia>
    - ModelImpl
    - Controller
    - SkeletonInterface <interfaccia>
    - DataBaseConnessione
    - EsecutoreQuery
  + Gestione Grafica:
    - ServerGUI

**CLIENT LAB**

Di seguito verranno elencate le classi, con una breve descrizione di carattere architetturale e implementativo.

**CLIENTMAIN**

La classe principale che contiene il metodo **public static void main (String[] args) {}**, necessario per generare i componenti necessari al funzionamento del modulo ClientLab.

**VIEW**

Interfaccia che fornisce l’overload dei metodi alle classi che la implementano.

**VIEWIMPL**

Questa classe estende JFrame ed implementa l’interfaccia View.

Si occupa di gestire tutte le componenti grafiche dell’applicazione, aggiornando anche i singoli elementi.

**MODEL**

Interfaccia che fornisce l’overload dei metodi alle classi che la implementano.

**MODELIMPL**

Questa classe implementa l’interfaccia Model.

Il costruttore salva un riferimento di tipo View passato come parametro e istanzia un riferimento di tipo ServerInterface.

È dotata di un metodo denominato **updateModel ()** che riceve come parametri in input:

* Source 🡪 Il nome dell’oggetto che ha generato l’evento.
* Dati 🡪 inseriti dall’utente tramite interfaccia.

Dopo aver verificato l’oggetto che generato l’evento chiama i propri metodi secondo le condizioni elencate.

Con i dati ricevuti dalla classe Controller, dopo averli elaborati rendendoli compatibili con il formato richiesto, richiama i metodi di ServerInterface che si occuperanno di completare le operazioni richieste.

Infine, grazie al riferimento di tipo View, viene richiamato il metodo **updateView ()** utile per aggiornare, in base alle informazioni passate come parametro, le componenti grafiche.

**CONTROLLERIMPL**

Implementa l’interfaccia ActionListener.

Il costruttore salva i riferimenti di Model e View passati come parametri, e si occupa di catturare e aggiungere ai componenti grafici la possibilità di generare eventi.

All’interno di essa è presente il metodo **actionPerformed ()** che prende in input:

* e 🡪 di tipo ActionEvent, ossia l’evento generato.

Successivamente estrapola il nome dell’oggetto verificando l’istanza di appartenenza e seguendo un serie di condizioni, ricava le informazioni inserite nella View.

Fatto ciò, richiama il metodo del model, che come descritto in precedenza elaborerà i dati sul bottone d’origine.

PROXY

SERVERINTERFACE