POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Gestionale
Classe L8 – Ingegneria dell'Informazione

CREAZIONE DI UNA DASHBOARD PER L'EPIDEMIA DI COVID-19 IN ITALIA

Relatore
Prof. Fulvio Corno

Candidato
Torta Roberto
236929

A.A. 2019/2020

Sommario

1 - Proposta del progetto

1.1. Descrizione del problema proposto

Il candidato si propone di creare una interfaccia grafica per la visualizzazione semplificata dei dati nazionali e regionali sul contagio da Covid-19 in Italia, inoltre intende creare una simulazione il quanto più verosimile di una epidemia al variare delle condizioni poste dall'user.

1.2. Descrizione della rilevanza gestionale del problema

Durante la pandemia di Covid-19, in Italia abbiamo assistito a un forte incremento delle ricerche su *internet* riguardo il propagarsi della malattia, per cui si ritiene che riuscire a fornire pubblicamente l'accesso a dati del contagio aggiornati e affidabili, e in modo pratico ed intuitivo, per esempio potendosi concentrare sul periodo e sulla regione di propria preferenza, possa avere una buona utilità pratica.

Inoltre, la sezione simulativa, cercherà di avere il pregio di far capire quanto una piccola variazione dei parametri possa determinare una crescita o decrescita esponenziale del numero dei contagiati totali e dei morti.

1.3. Descrizione dei data-set per la valutazione

Il *data-set* verrà realizzato prendendo i dati dal *repository* del Dipartimento della Protezione Civile italiana, in modo da utilizzare dati il più possibile corretti e verificati.

Il *repository* è reperibile al seguente indirizzo IP: https://github.com/pcm-dpc/COVID-19

1.4. Descrizione preliminare degli algoritmi coinvolti

I principali problemi da affrontare saranno, una volta creati i database, creare i metodi per estrarre i dati in modo funzionale alle operazioni da eseguire in seguito, creare i metodi che permettano di calcolare agilmente i vari tassi richiesti dall'*user*, a seconda della data e della regione richieste da esso. Verranno poi creati metodi per il calcolo dei tassi partendo da dati di input.

L'ultimo passo sarà quello di creare un algoritmo di simulazione del contagio.
Lo stato del mondo corrisponderà al numero dei contagiati totali, a quello dei morti ed a quello dei guariti. L'algoritmo partirà con N persone contagiate (passate da parametro), a ciascuna di loro sarà assegnata, sempre dall'user, una probabilità di contagiare altre persone, una di guarire ed una di morire; queste probabilità verranno modificate ogni volta che una persona passerà nella coda degli eventi, alzando la probabilità di morire o di guarire a scapito di quella di contagiare altre persone. Sempre l'utente sceglierà dopo quanti giorni una persona, venuta a contatto con il virus, inizierà a sviluppare sintomi, e di conseguenza iniziare a sua volta a contagiare.
L'utente inoltre potrà scegliere tramite interfaccia grafica di modificare le probabilità dei 3 eventi (ad esempio aumentare la probabilità di morte simulando la saturazione degli ospedali, aumentare la probabilità di guarigione con trattamenti specializzati, diminuire il livello di contagiosità introducendo il distanziamento sociale).

Inoltre a ogni persona della simulazione verrà assegnato casualmente un valore numerico, che simulerà l'età del paziente in questione e modificherà a sua volta il valori probabilistici.
La simulazione terminerà una volta esaurite tutte le persone nelle code degli eventi oppure dopo N giorni simulati (scelti sempre dall'*user*), le persone smetteranno di contagiare e si andrà ad esaurire i contagiati già presenti, i risultati ottenuti dell'algoritmo saranno quindi stampati a video e resi fruibili in forma grafica ed intuitiva.

1.5. Descrizione preliminare delle funzionalità previste per l'applicazione software

Interfacciandosi con l'applicazione, l'utente si troverà davanti a due scelte e quindi due diverse funzionalità: la prima sarà visibile sulla schermata principale, su cui saranno disponibili vari metodi di ricerca che permettano all'utente di combinare a proprio piacimento data e luogo dei dati che vuole ricevere. Le informazioni ottenute saranno stampate a video sia in formato testuale, sia tramite un grafico a torta, per fornire al fruitore una lettura migliore e più visuale dei dati. Infine, sarà possibile calcolare i vari tassi a seconda del giorno e del luogo scelti, utilizzando quattro tasti specifici. Con un bottone si potrà poi accedere alla seconda schermata e quindi alla seconda funzionalità: la simulazione del contagio. In questo caso il programma restituirà i dati della simulazione a partire dai dati forniti dall' input dell'utente tramite *slider* o *check-box*. le informazioni saranno disponibili, come nella prima funzionalità, sia in formato testuale, sia in formato grafico.

2 - Descrizione dettagliata del problema affrontato

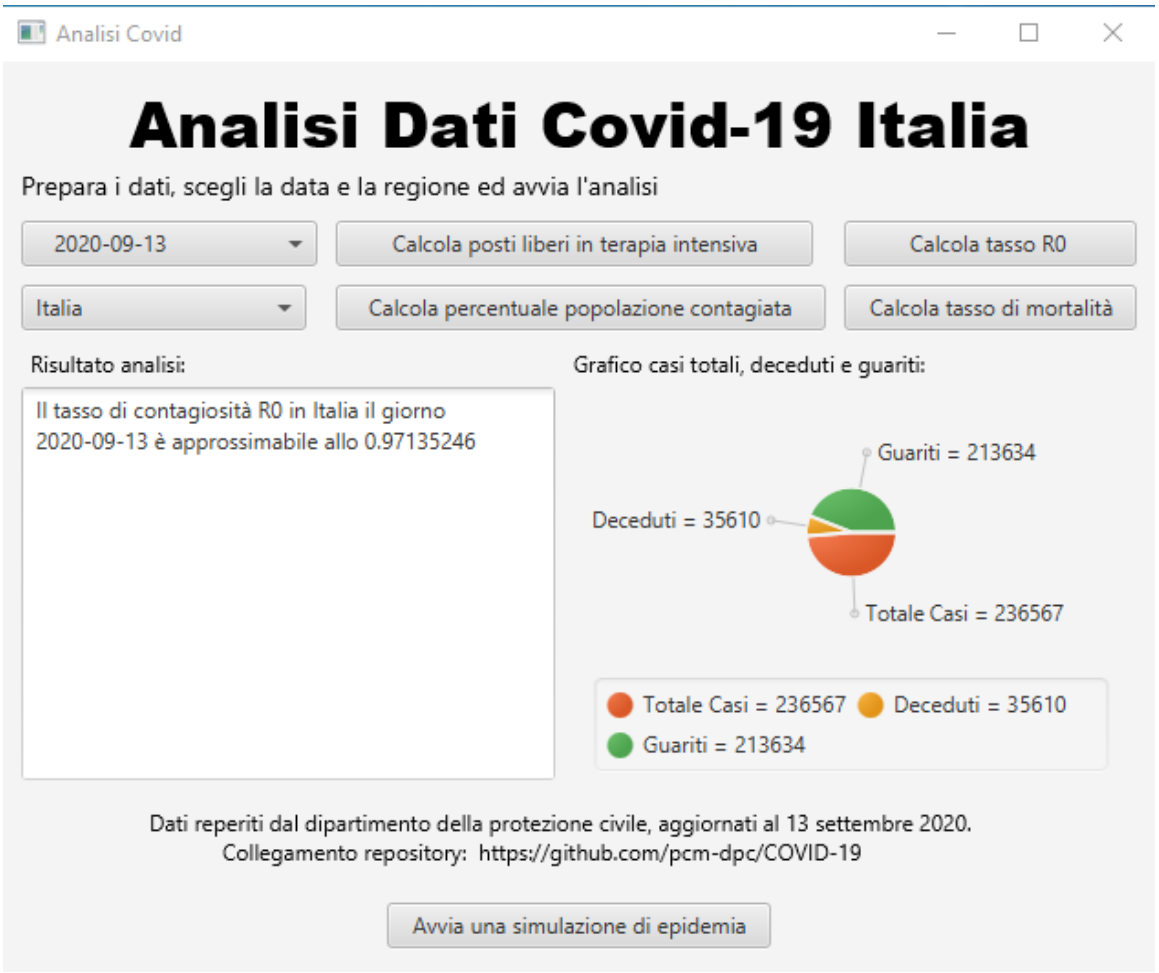
Il programma è stato realizzato con l'obbiettivo di rendere più intuitivo e gratificante il reperimento dei dati. In particolare, l'uso dell'interfaccia grafica mirerà a:

- Rendere più veloce la consultazione del database;
- Ottimizzare le ricerche dei tassi interessati;
- Fornire una visualizzazione grafica ed intuitiva di tutti i risultati delle ricerche;
- Stimolare la familiarizzazione con la crescita esponenziale del contagio.

Tali obiettivi saranno raggiunti attraverso le due funzionalità principali dell'applicazione: la ricerca nel *database* e la simulazione dell'epidemia.

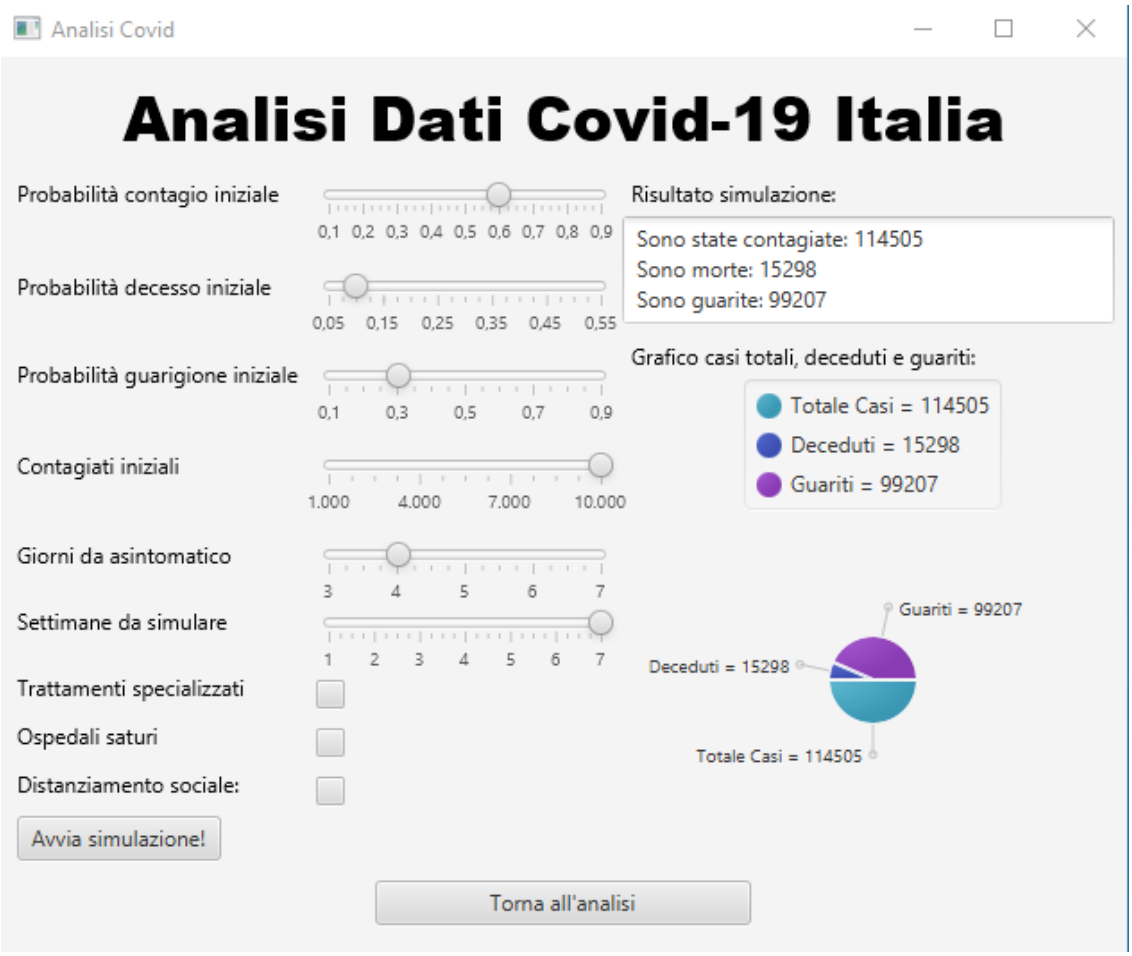
2.1. Ricerca nel database e calcolo dei tassi

Nella sezione di ricerca, presente nella schermata principale del programma, si dà all'utente la possibilità di effettuare ricerche personalizzate nel database. In particolare, il fruitore si troverà davanti a due *combo-box* che permetteranno di scegliere sia la data sia il luogo su cui applicare la ricerca. Inoltre, questi *combo-box* sono programmati per modificare, senza la necessità di ulteriori *click* aggiuntivi, il grafico a torta; sono presenti anche cinque bottoni: quattro per calcolare i vari tassi, sempre nel giorno e nella regione selezionata, e l'ultimo per passare alla schermata di simulazione dell'epidemia.



2.2. Simulazione

La seconda funzionalità riguarda la simulazione di un'ipotetica epidemia: per quest'opzione verrà richiesto all'utente di inserire i vari parametri tramite *slider* e *check-box*. In particolare, tramite *slider* sarà possibile modificare il tasso di moltiplicazione del *virus*, le probabilità di decesso e di guarigione, il numero iniziale di persone contagiate, quanti giorni passano prima che una persona contagi altri individui e abbia complicanze di salute, e il numero delle settimane che si intende simulare. I limiti degli *slider* sono stati inseriti per evitare problemi di *overflow*, scegliendo valori che un normale *computer* dovrebbe essere in grado di gestire. I tre *check-box* inseriti, se selezionati, potranno cambiare i valori probabilistici, a esempio aumentando i morti o i guariti, o diminuendo i contagi. Infine, due tasti permetteranno rispettivamente di avviare la simulazione, stampando a video i risultati e creando un grafico, e di tornare all'interfaccia precedente.



4 - Diagramma delle classi principali e descrizione algoritmi coinvolti

Per una migliore gestione dell'applicazione, è stato utilizzato il pattern MVC (*Model-View-Controller*), separando la logica di presentazione con cui si interfaccia l'utente, dalla logica di elaborazione che rende realmente operativa l'applicazione. Inoltre, si è voluto sfruttare anche il pattern DAO (*Data-Access-Object*) con l'obiettivo di mantenere separate tutte quelle operazioni che richiedono l'accesso alle tabelle del database attraverso delle query SQL.

Per questi motivi, il software è composto da tre pacchetti:

- Dashboard_Dati_Covid_19_Italia, è composto dalle classi *Main.java* e *EntryPoint.java* che permettono di avviare l'applicazione, e dalle due classi Controller (*AnalisiDatiController.java, SimulazioneController.java*) che gestiscono le interfacce d'utente relative alla ricerca nel database e alla simulazione di epidemia.

- *DB*, è composto dalle due classi che permettono di interagire con il database utilizzando il linguaggio SQL. La prima è la classe *DBConnect.java* in cui è definita la procedura per accedere al database "daticoviditalia" e i cui metodi di accesso sono richiamati ogni volta in cui si vuole eseguire una query. L'altra è *DatiCovidItaliaDAO.java*: essa contiene tutti i metodi, richiamati nel *Model*, attraverso cui l'applicazione effettua le interrogazioni SQL al database.

- *Model*, contiene le classi che definiscono la logica vera e propria dell'applicazione. Qui si trovano le classi *Analisi.java* e *Simulatore.java* in cui sono contenute i metodi logici che verranno chiamati dai rispettivi Controller. Inoltre, in questo pacchetto sono presenti tutte le classi create per andare ad ospitare i dati richiesti dalle *query* e per supportare il simulatore, ovvero *DatiPerGrafico.java, DatiPerPercentuali.java, DatiRegionali.java, DatiNazionali.java, Event.java, Persona.java*.

4.1. Il pacchetto "DB"

Come già anticipato, nel pacchetto DB si trova la classe *DBConnect.java*, responsabile delle procedure per accedere al database, sfruttando la libreria "*HikariCP*":

```
public class DBConnect {
    private static final String jdbcURL = "jdbc:mysql://localhost/daticoviditalia?serverTimezone=UTC";
    private static HikariDataSource ds;

    public static Connection getConnection() {

        if (ds == null) {

            ds = new HikariDataSource();

            ds.setJdbcUrl(jdbcURL);
            ds.setUsername("");
            ds.setPassword("");
            ds.addDataSourceProperty("cachePrepStmts", "true");
            ds.addDataSourceProperty("prepStmtCacheSize", "250");
            ds.addDataSourceProperty("prepStmtCacheSqlLimit", "2048");

        }

        try {

            return ds.getConnection();

        } catch (SQLException e) {
            System.err.println("Errore connessione al DB");
            throw new RuntimeException(e);
        }
    }

}
```

Come possiamo vedere, i campi username e password sono lasciati in bianco per permettere all'utente di inserire le proprie credenziali di accesso.

L'altra classe, chiamata *DatiCovidItaliaDAO.java*, contiene 5 metodi per interrogare i database a seconda dell'informazione richiesta:

- estraiDatiNazionali() permette di estrarre data, terapia intensiva, nuovi positivi, dimessi guariti, deceduti e totale casi dalla tabella "datinazionali" nel database "DatiCovidItalia". Il metodo ritorna una *TreeMap* dei dati nazionali, con la data come chiave primaria;
- estraiDatiRegionaliPerGiornata(String dataDaCercare) permette di estrarre data, regione, terapia intensiva, nuovi positivi, dimessi guariti, deceduti e totale casi dalla tabella "datiregionali" nel database "DatiCovidItalia", per la data passata come parametro in formato stringa. Il metodo ritorna una *LinkedList* dei dati di ogni regione nella data passata come parametro;

- estraiDatiRegionePerRegione(String regioneDaCercare) permette di estrarre data, regione, terapia intensiva, nuovi positivi, dimessi guariti, deceduti e totale casi dalla tabella “datiregionali” nel database “DatiCovidItalia”, per la regione passata come parametro in formato stringa. Il metodo ritorna una *LinkedList* dei dati di ogni giorno della regione passata come parametro;
- estraiDatiPerPercentuali() permette di estrarre Luogo, popolazione totale e posti in terapia intensiva dalla tabella statistiche popolazione nel database “DatiCovidItalia”;

  Il metodo ritorna una *TreeMap* dei dati di ogni regione (Italia compresa), usando il nome della regione come chiave primaria.
- estraiDatiPerGrafico(String regione,LocalDate data) permette di estrarre totale casi, deceduti e dimessi a livello regionale e nazionale, a seconda della data e della regione passati da parametro. Il metodo ritorna l’oggetto DatoPerGrafico.

A titolo di esempio si riporta il metodo estraiDatiPerGrafico, in quanto gli altri seguono logiche algoritmiche simili.

```
public DatoPerGrafico estraiDatiPerGrafico(String regione,LocalDate data){
    String sql;
    if(regione.equals("Italia"))
        sql = "SELECT totale_casi, deceduti , dimessi_guariti FROM datiNazionali WHERE SUbstr(data,1,10) like ?";
    else
        sql="SELECT totale_casi, deceduti , dimessi_guariti FROM datiRegionali WHERE SUbstr(data,1,10)=?"+
    "AND denominazione_regione=?";

    DatoPerGrafico dpg;

    try {
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);
        st.setString(1, "%"+data.toString());
        if(!regione.equals("Italia")) {
            st.setString(1, data.toString());
            st.setString(2, regione);
        }

        ResultSet rs = st.executeQuery();
        rs.next();
        dpg= new DatoPerGrafico(rs.getInt("totale_casi"), rs.getInt("deceduti"), rs.getInt("dimessi_guariti"));

        st.close();
        conn.close();

    } catch (SQLException e) {
        e.printStackTrace();
        throw new RuntimeException("Errore di connessione al Database.");
    }
    return dpg;
}
```

4.2. Il pacchetto “Model”

Questo pacchetto contiene tutte le classi pensate per ospitare i dati che verranno prelevati dalle *query* SQL:

- DatoNazionale: è una classe pensata per ospitare tutti i dati estratti dal metodo estraiDatiNazionali, gli attributi sono quindi:
  1. private LocalDate data;

2. private int terapiaIntensiva;
3. private int nuoviPositivi;
4. private int dimessiGuariti;
5. private int deceduti;
6. private int totaleCasi.

- DatoPerGrafico: è una classe pensata per ospitare tutti i dati estratti dal metodo estraiDatiPerGrafico, con i seguenti attributi:
  1. private Integer totaleCasi;
  2. private Integer totaleDecessi;
  3. private Integer guariti.
- DatiPerPercentuali: è una classe pensata per ospitare tutti i dati estratti dal metodo estraiDatiPerPercentuali, caratterizzata dai seguenti attributi:
  1. private String nome;
  2. private int popolazioneTotale;
  3. private int postiTerapiaIntensiva.
- DatoRegionale: è una classe pensata per ospitare tutti i dati estratti dai metodi estraiDatiRegionaliPerGiornata e estraiDatiRegionePerRegione, gli attributi sono quindi:
  1. private LocalDate data;
  2. private String regione;
  3. private int terapiaIntensiva;
  4. private int nuoviPositivi;
  5. private int dimessiGuariti;
  6. private int deceduti;
  7. private int totaleCasi.

Il pacchetto contiene inoltre le due classi su cui si appoggia il simulatore:

- Persona: è una classe che simula e caratterizza le persone che verranno contagiate nella simulazione, ogni persona sarà caratterizzata da una probabilità di contagiare altre persone, da una probabilità di morire e da una di guarire. Inoltre, ad una persona sarà assegnata in modo casuale una età con il metodo *Math.random*, che inciderà sulle probabilità in questione.

```java
public class Persona {

    private double probabilitaContagio;
    private double probabilitaGuarigione;
    private double probabilitaDecesso;
    private double eta;


    public Persona(double probabilitaContagio, double probabilitaGuarigione,
            double probabilitaDecesso) {
        this.probabilitaContagio = probabilitaContagio;
        eta=Math.random();
        if(eta<0.4) {
        this.probabilitaGuarigione = probabilitaGuarigione+0.1;
        this.probabilitaDecesso = 0;
        }
        if(eta>=0.4&&eta<0.6) {
            this.probabilitaGuarigione = probabilitaGuarigione;
            this.probabilitaDecesso = 0;
        }
        if(eta>=0.6&&eta<0.8) {
            this.probabilitaGuarigione = probabilitaGuarigione-0.1;
            this.probabilitaDecesso = probabilitaDecesso+0.1;
        }
        if(eta>=0.8) {
            this.probabilitaGuarigione = probabilitaGuarigione-0.2;
            this.probabilitaDecesso = probabilitaDecesso+0.2;
        }
    }
```

- Event: è una classe che permette di distinguere i vari tipi di eventi che andranno messi nella coda di priorità. I parametri sono il tipo dell'evento, la persona in questione e il giorno. Inoltre, la classe implementa il metodo *compareTo* per rendere due eventi confrontabili all'interno della coda.

```java
public class Event implements Comparable<Event> {

    public enum EventType {
        CONTAGIA, GUARISCI, DECEDI,
    }

    private EventType type;
    private Integer giorno;
    private Persona persona;

    @Override
    public int compareTo(Event o) {
        // TODO Auto-generated method stub
        if (this.giorno == o.giorno) {
            if (this.persona.equals(o.persona)) {
                if (this.type.equals(EventType.CONTAGIA)) {
                    return -1;
                } else {
                    return +1;
                }
            }
            if ((this.type.equals(EventType.CONTAGIA)) && !(o.type.equals(EventType.CONTAGIA))) {
                return -1;
            } else {
                return +1;
            }
        }
        return this.giorno.compareTo(o.giorno);
    }
```

Infine, sempre questo pacchetto contiene le classi che si occupano di calcolare i tassi e di gestire il simulatore:

- Analisi: questa è la classe che si occupa di calcolare tutti i tassi che verranno richiesti dall'user e presenta dunque i seguenti metodi:
    1. calcolaPercentualeAmmalatiPerLuogoNelGiorno(String luogo, String data): permette di ricavare la percentuale di contagiati sulla popolazione, nel luogo e nella data passati da parametro; funziona sia a livello regionale che a livello nazionale. Il metodo ritorna la percentuale *float* dei contagiati;
    2. calcolaPostiLiberiTerapiaIntensivaPerLuogoNelGiorno(String luogo, String data): permette di calcolare i posti ancora liberi in terapia intensiva, nel luogo e nella data passati da parametro; funziona sia a livello regionale che a livello nazionale. Il metodo ritorna il numero intero dei posti ancora disponibili in terapia intensiva;
    3. calcolaTassoMortalitaPerLuogoNelGiorno(String luogo, String data): permette di calcolare il tasso di mortalità (deceduti rispetto ai contagiati) nel luogo e nella data passati da parametro; funziona sia a livello regionale che a livello nazionale. Il metodo ritorna il tasso *float* di mortalità nella regione e nel giorno passati come parametro;
    4. calcolaTassoContagiosità(String luogo, String data): permette di calcolare il tasso di contagiosità R0 (quanti persone infetta ogni contagiato) nel luogo e nella data passati da parametro; funziona sia a livello regionale che a livello nazionale. Il metodo ritorna il tasso *float* di contagiosità nella regione e nel giorno passati come parametro;
    5. estraiDatiPerGrafico(String regione,LocalDate data): permette di estrarre i dati che andranno inseriti dentro al grafico, sempre a seconda del giorno e del luogo desiderato.

  A titolo di esempio è stato riportato il metodo calcolaTassoMortalitaPerLuogoNelGiorno in quanto la logica algoritmica degli altri è simile.

```java
public float calcolaTassoMortalitaPerLuogoNelGiorno(String luogo, String data) {
    float tasso = 0;
    LocalDate giorno = LocalDate.parse(data);
    if (luogo.equals("Italia")) {
        if (datiNazionali.containsKey(giorno)) {
            tasso = ((float) datiNazionali.get(giorno).getDeceduti() * 100)
                    / (float) datiNazionali.get(giorno).getTotaleCasi();
            return tasso;
        }
    }

    DatoRegionale temp = null;
    DatiCovidItaliaDAO dao = new DatiCovidItaliaDAO();
    for (DatoRegionale dr : dao.estraiDatiRegionaliPerGiornata(data)) {
        if (dr.getRegione().equals(luogo))
            temp = dr;
    }
    tasso = ((float) temp.getDeceduti() * 100) / (float) temp.getTotaleCasi();

    return tasso;
}
```

- Simulatore: questa classe contiene tutta la logica necessaria ad effettuare una simulazione del contagio e si divide in quattro fasi:
    1. Costruttore: il costruttore del simulatore riceve tutti i parametri che sono modificabili tramite l'interfaccia grafica:
        a) private double trattamentoSpecializzato;
        b) private double ospedaliSaturi;
        c) private double distanziamento;
        d) private int contagiatiIniziali;
        e) private int settimanaDiFine;
        f) private double probContagioBase;
        g) private double probMorteBase;
        h) private double probGuarigioneBase;
        i) private int giorniDaAsintomatico.
    2. Inizializzazione: il metodo di inizializzazione permette di pulire eventuali simulazioni vecchie, calcola le probabilità iniziali scelte dall'*user*, crea una nuova coda degli eventi e va ad inserire tutti gli eventi iniziali, ovvero gli eventi del tipo "CONTAGIA" e di tipo "GUARISCI" per tutte le persone scelte come contagiati iniziali;

```java
public void init() {
    //Pulisco eventuali simulazioni vecchie ed aggiorno le probabilità a seconda dei parametri di input
    this.queue = new PriorityQueue<Event>();
    this.totaleContagiati = this.totaleGuariti = this.totaleMorti = 0;
    this.probContagioIniziale = this.probContagioBase - this.distanziamento;
    this.probMorteIniziale = this.probMorteBase + this.ospedaliSaturi;
    this.probGuarigioneIniziale = this.probGuarigioneBase + trattamentoSpecializzato;

    // genero gli eventi iniziali
    for (int i = 0; i < contagiatiIniziali; i++) {
        Persona persona = new Persona(probContagioIniziale, probGuarigioneIniziale, probMorteIniziale);
        queue.add(new Event(EventType.CONTAGIA, 1, persona));
        queue.add(new Event(EventType.GUARISCI, 1, persona));
        totaleContagiati++;
    }
}
```

3. Esecuzione: il metodo estrae un evento dalla coda fino a quando quest'ultima non rimane vuota, e chiede ad un altro metodo di processarlo;
4. Processamento degli eventi: questo metodo è il vero cuore logico della simulazione: ogni evento processato sarà gestito in modo diverso a seconda del proprio tipo:
    i. "CONTAGIA": la persona potrebbe contagiarne un'altra (sempre casualmente con la funzione *Math.random*), in questo caso si crea una persona nuova e si aggiungono due nuovi eventi "CONTAGIA" e "GUARISCI" per il primo giorno finito il periodo da asintomatico, infine si aggiornano i parametri di *output*;
    ii. "GUARISCI": la persona potrebbe guarire dal virus, nel caso positivo non si aggiunge nessun nuovo evento alla coda e si aggiornano i parametri di *output*, se invece la persona non guarisce, rischierà di morire, quindi si aggiunge un nuovo evento "DECEDI" per la stessa persona e per lo stesso giorno;
    iii. "DECEDI": la persona potrebbe morire a causa del virus, nel caso positivo non si aggiunge nessun evento alla coda e si aggiornano i parametri di *output*, se invece la persona non muore si vanno a modificare tutte le probabilità, aumentando quella di morire e di guarire e diminuendo quella di contagiare, e si creano due nuovi eventi "CONTAGIA" e "GUARISCI" per la stessa persona nel giorno successivo.

```java
private void processEvent(Event e) {
    double probabilita;
    switch (e.getType()) {
    case CONTAGIA:
        // la persona potrebbe contagiare un'altra persona, in questo caso creo una persona nuova ed aggiungo
        //due nuovi eventi (contagia e guarisci) per il primo giorno finito il periodo da asintomatico
        probabilita = e.getPersona().getProbabilitaContagio();
        if (Math.random() <= probabilita && e.getGiorno() < settimanaDiFine*7) {
            Persona persona = new Persona(probContagioIniziale, probGuarigioneIniziale, probMorteIniziale);
            queue.add(new Event(EventType.CONTAGIA, e.getGiorno() + giorniDaAsintomatico, persona));
            queue.add(new Event(EventType.GUARISCI, e.getGiorno() + giorniDaAsintomatico, persona));
            totaleContagiati++;
        }
        break;
    case GUARISCI:
        // la persona potrebbe guarire, se non guarisce mi aggiuge alla coda la
        // probabilità di morire nello stesso giorno
        probabilita = e.getPersona().getProbabilitaGuarigione();
        if (Math.random() <= probabilita) {
            totaleGuariti++;
        } else {
            queue.add(new Event(EventType.DECEDI, e.getGiorno(), e.getPersona()));
        }
        break;
    case DECEDI:
        // la persona potrebbe morire, altrimenti si modificano le probabilità e si crea
        // un evento contagia e uno guarisci per il giorno dopo
        probabilita = e.getPersona().getProbabilitaDecesso();
        if (Math.random() <= probabilita) {
            totaleMorti++;
            break;
        }

    // Aggiorno le probabilità, diminuendo la probabilità di contagiare ed aumentando quella di guarire e di morire
    if (e.getPersona().getProbabilitaContagio() > 0)
        e.getPersona().setProbabilitaContagio(e.getPersona().getProbabilitaContagio() - diminuazioneProbContagio);
    if (e.getPersona().getProbabilitaDecesso() < 1)
        e.getPersona().setProbabilitaDecesso(e.getPersona().getProbabilitaDecesso() + aumentoProbMorte);
    if (e.getPersona().getProbabilitaGuarigione() < 1)
        e.getPersona().setProbabilitaGuarigione(e.getPersona().getProbabilitaGuarigione() + aumentoProbGuarigione);

    queue.add(new Event(EventType.CONTAGIA, e.getGiorno() + 1, e.getPersona()));
    queue.add(new Event(EventType.GUARISCI, e.getGiorno() + 1, e.getPersona()));
    break;
}
```

4.3. Il pacchetto "Dashboard_Dati_Covid_19_Italia"

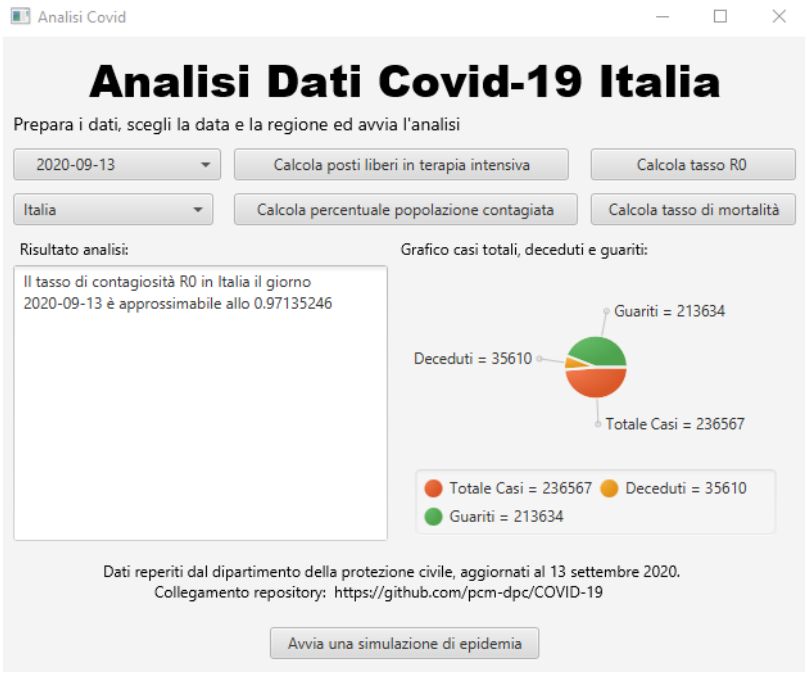
Il pacchetto Dashboard_Dati_Covid_19_Italia contiene, oltre alle classiche classi *Main.java* ed *Entrypoint.java*, responsabili del lancio del programma, anche le due classi che gestiscono le interfacce utente, ovvero *AnalisiDatiController.java* e *SimulazioneController.java*.

- AnalisiDatiController: è la classe che si occupa della prima schermata, quella relativa all'analisi dei dati; presenta tutti i vari metodi che richiamano i vari metodi già presentati nel pacchetto model e si occupa di stampare i risultati a video sia in forma testuale che in forma grafica.

Si allega un esempio per l'evento che si genera con un *click* sul bottone "Calcola tasso di mortalità" ed il metodo per disegnare il grafico.

```java
@FXML
void calcolaTassoDiMortalita(ActionEvent event) {
    txtResult.clear();
    LocalDate dataDesiderata = comboBoxData.getValue();
    String regioneDesiderata = comboBoxRegione.getValue();
    if (dataDesiderata == null || regioneDesiderata == null) {
        txtResult.appendText("Selezionare una data e una regione!!");
        return;

    }

    float tassoMortalita = model.calcolaTassoMortalitaPerLuogoNelGiorno(regioneDesiderata,
            dataDesiderata.toString());
    txtResult.appendText("Il tasso di mortalità in " + regioneDesiderata + " il giorno \n"
            + dataDesiderata.toString() + " è dello " + tassoMortalita + "%");

}

void disegnaGrafico() {

    String regioneDesiderata = comboBoxRegione.getValue();
    LocalDate dataDesiderata = comboBoxData.getValue();

    DatoPerGrafico dpg= model.estraiDatiPerGrafico(regioneDesiderata,dataDesiderata);
    ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList(
            new PieChart.Data("Totale Casi = "+dpg.getTotaleCasi(), dpg.getTotaleCasi()),
            new PieChart.Data("Deceduti = "+dpg.getTotaleDecessi(), dpg.getTotaleDecessi()),
            new PieChart.Data("Guariti = "+dpg.getGuariti(), dpg.getGuariti()));
    graficoResult.setData(pieChartData);

}
```

- SimulazioneController: è la classe che si occupa della seconda schermata, quella relativa alla simulazione del contagio; si occupa di ricevere dagli *slider* i valori per la simulazione e chiamare i metodi della classe simulazione, per poi stampare a video in formato testuale e grafico.



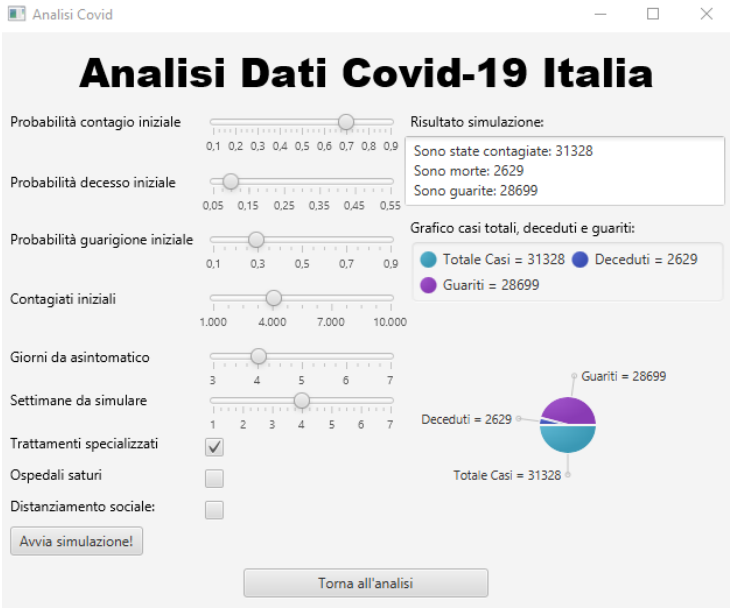

```
contagiatiIniziali = (int) this.sldContagiIniziali.getValue();
settimanaDiFine = (int) this.sldSettimane.getValue();
giorniDaAsintomatico = (int) this.sldGiorniDaAsintomatico.getValue();
probabilitaContagioIniziale = (double) this.sldProbabilitaContagioIniziale.getValue();
probabilitaDecessoIniziale = (double) this.sldProbabilitaDecessoIniziale.getValue();
probabilitaGuarigioneIniziale = (double) this.sldProbabilitaGuarigioneIniziale.getValue();

if(this.cBoxTrattamenti.isSelected()) {
    trattamentoPlasma=0.2;
}
if(this.cboxDistanziamento.isSelected()) {
    distanziamento=0.1;
}
if(this.cBoxOspedali.isSelected()) {
    ospedaliSaturi=0.2;
}

Simulatore sim= new Simulatore(contagiatiIniziali, settimanaDiFine, distanziamento, trattamentoPlasma, ospedaliSaturi,
                               giorniDaAsintomatico,probabilitaContagioIniziale,probabilitaDecessoIniziale,
                               probabilitaGuarigioneIniziale);
sim.init();
sim.run();

this.txtResult.appendText("Sono state contagiate: "+sim.getTotaleContagiati());
this.txtResult.appendText("\nSono morte: "+sim.getTotaleMorti());
this.txtResult.appendText("\nSono guarite: "+sim.getTotaleGuariti());

ObservableList<PieChart.Data> pieChartData = FXCollections.observableArrayList(
        new PieChart.Data("Totale Casi = "+sim.getTotaleContagiati(), sim.getTotaleContagiati()),
        new PieChart.Data("Deceduti = "+sim.getTotaleMorti(), sim.getTotaleMorti()),
        new PieChart.Data("Guariti = "+sim.getTotaleGuariti(), sim.getTotaleGuariti()));
graficoResult.setData(pieChartData);
```

5 - *Esempi di utilizzo dell'applicazione*

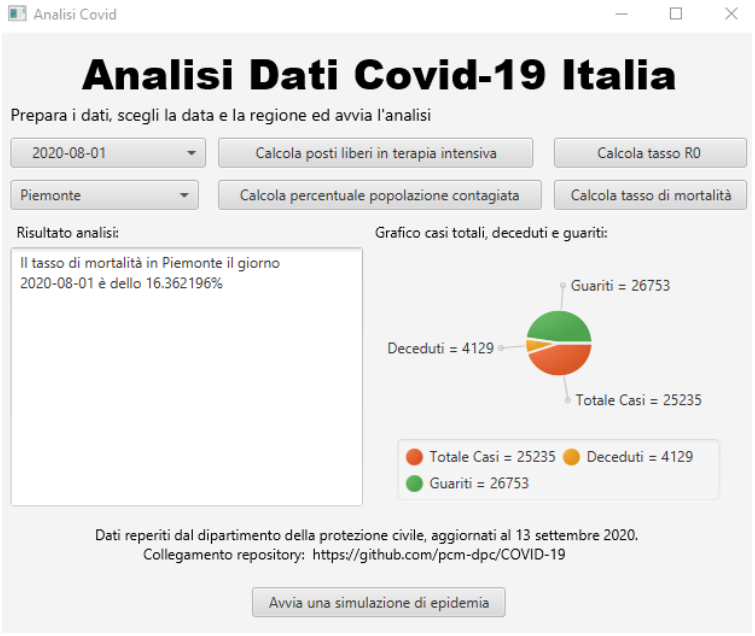
Di seguito si presentano alcuni esempi di utilizzo dell'applicazione.

5.1. Esempio analisi dati

In questo esempio supponiamo che l'utente voglia ricevere le informazioni relative al 13 settembre 2020 in Italia, e che inoltre richieda il calcolo del tasso di contagio R0.



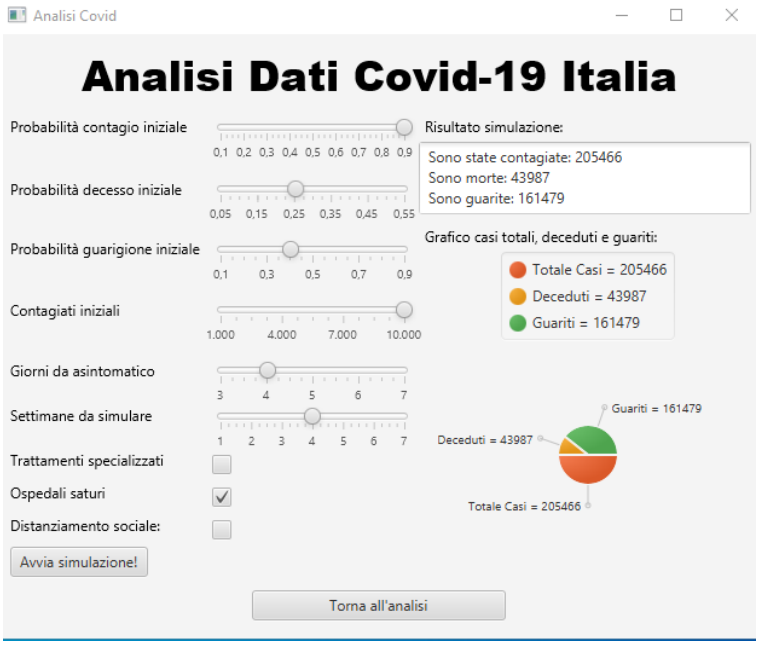
In questo caso invece supponiamo che l'utente richieda informazioni sul Piemonte nel primo giorno di agosto 2020, e che richieda anche il calcolo del tasso di mortalità.



5.2 Esempio simulazione

In questa simulazione abbiamo ipotizzato una probabilità di contagio iniziale del 70%, una di morte del 10%, una di guarigione del 30%, un numero iniziale di 4000 persone contagiate, 4 giorni da asintomatico, la presenza di cure specializzate per questo virus ed abbiamo simulato 4 settimane.

In questa simulazione invece abbiamo ipotizzato una probabilità di contagio iniziale del 90%, una di morte del 25%, una di guarigione del 40%, un numero iniziale di 10000 persone contagiate, 4 giorni da asintomatico, la saturazione degli ospedali e abbiamo simulato 4 settimane.



5.3. Link del video YouTube

Il *link* al video di esempio di utilizzo dell'applicazione caricato su YouTube è il seguente:

https://www.youtube.com/watch?v=xvOfxPU52gg&feature=youtu.be

6 - Valutazione dei risultati e Conclusioni

L'obbiettivo primario di riuscire a fornire all'utente le informazioni sull'epidemia in Italia è stato raggiunto, i dati sono ben leggibili e graficamente appaganti. In futuro sarebbe ottimo continuare a sviluppare il progetto aggiungendo nuovi indici e calcolando quelli già presenti in modo più rigoroso dal punto di vista statistico, andando a eliminare le varie semplificazioni fatte per questo lavoro.

Anche l'obbiettivo di far familiarizzare l'utente con la crescita esponenziale del contagio è stata raggiunta, la simulazione risulta essere semplice, intuitiva e personalizzabile per l'utilizzatore. In futuro questo simulatore potrebbe essere facilmente arricchito di nuove funzionalità, parametri e probabilità, per renderlo più rigoroso da un punto di vista epidemiologico.