POLITECNICO DI TORINO

Dipartimento di Ingegneria Gestionale e della Produzione

Corso di Laurea
in Ingegneria Gestionale
Classe L-8 Ingegneria dell'Informazione

Prova Finale

Programmazione dei turni lavorativi del personale infermieristico in un reparto ospedaliero

Relatore

prof. Fulvio Corno

Candidato

Riccardo Baldassa

Anno Accademico 2018/2019

Indice

1. Proposta di progetto

Studente proponente

s227583 Baldassa Riccardo

Titolo della proposta

Programmazione dei turni lavorativi del personale infermieristico in un reparto ospedaliero

Descrizione del problema proposto

L'applicazione si propone di generare in modo automatico ed ottimale l'orario lavorativo di una serie di infermieri assegnati ad un certo reparto ospedaliero. La creazione manuale dei turni infermieristici è un'operazione complessa che richiede molto tempo, soprattutto quando il numero di infermieri da gestire aumenta. L'obbiettivo dell'applicazione è quindi quello di automatizzare quest'operazione rispettando i vincoli normativi reali, cercando di assegnare i turni nel modo più equo possibile e cercando di andare incontro alle esigenze dei singoli infermieri.

Descrizione della rilevanza gestionale del problema

La programmazione dei turni di lavoro degli infermieri negli ospedali è materia di studio tutt'oggi, esistono diverse tipologie di gestione in questo ambito che sono state sviluppate nel corso degli anni, ognuna con le proprie problematiche. Tuttavia, tutte queste diverse tipologie di gestione vengono attualmente ancora implementate manualmente. La gestione di questo problema è molto complicata per il gran numero di vincoli, di variabili e di dati da tenere in conto. Una ricerca manuale quasi sicuramente non porterà alla soluzione migliore, soprattutto quando il personale infermieristico aumenta. L'utilizzo di questa applicazione permetterebbe dunque di far risparmiare molto tempo agli addetti che svolgono questo compito e di programmare nel modo più efficace i turni degli infermieri, cercando di non avere disparità di trattamento.

Descrizione dei data-set per la valutazione

I data-set utilizzati comprenderanno i dati anagrafici degli infermieri assegnati al reparto, la lista delle ferie richieste da ciascuno di essi e l’insieme dei vincoli normativi e dei vincoli legati alla tipologia del reparto preso in considerazione.

Descrizione preliminare degli algoritmi coinvolti

L'algoritmo principale utilizzerà una funzione ricorsiva e sarà molto articolato perché dovrà verificare una molteplicità di vincoli. L'algoritmo cercherà di generare la programmazione annuale dei turni di tutti gli infermieri, a partire dai giorni di ferie da loro richiesti. Il periodo tenuto in conto dall'applicazione va dal 01 settembre 2019 al 31 agosto 2020. La soluzione sarà accettata se tutti i turni lavorativi del reparto verranno coperti e se i vincoli contrattuali degli infermieri verranno rispettati. Oltre al rispetto dei vincoli normativi, l'algoritmo cercherà di trovare la soluzione ottimale che permetta di non creare disparità di trattamento tra gli infermieri: si cercherà di assegnare ad ogni infermiere lo stesso numero, per quanto possibile, di mattini, pomeriggi, notti, riposi e riposi in giorni di festività (il sabato viene considerato come giorno festivo).

Descrizione preliminare delle funzionalità previste per l'applicazione software

Nella prima parte dell'applicazione, l'utente potrà visualizzare e modificare per ognuno degli infermieri che lavorano nel reparto ospedaliero i 32 giorni di ferie richiesti da essi. Dopo aver confermato le ferie richieste si potrà avviare la funzione principale che genererà la programmazione dei turni di tutto l'anno. Sarà poi possibile visualizzare l'orario mese per mese ed eventualmente lo si potrà salvare in un file di testo sul proprio computer. Infine, sarà possibile visualizzare le statistiche di ogni infermiere che comprendono la varietà dei turni loro assegnati (mattini, pomeriggi, notti) e quanti riposi hanno fatto nei giorni festivi e feriali durante l'anno.

2. Descrizione dettagliata del problema affrontato

La programmazione dei turni del personale infermieristico è una materia di studio di rilevanza gestionale tutt'ora attuale. Questa programmazione è influenzata da diversi vincoli e variabili. La gestione di questi vincoli e variabili rende la programmazione manuale molto complessa, soprattutto nel cercare una buona soluzione che renda equo il trattamento degli infermieri. L'automazione di questa operazione permetterebbe un grande risparmio di tempo al responsabile della gestione dei turni (esempio: caposala) e, soprattutto, la possibilità di ottenere una programmazione migliore per quanto riguarda l'equità di trattamento degli infermieri.

La prima variabile da considerare è il turno, organizzato in fasce orarie giornaliere e assegnato ad un numero variabile di infermieri. I turni si possono suddividere in: turno a ciclo diurno, turno a ciclo continuo (sulle 24 ore) con sequenza regolare, turno a ciclo continuo (sulle 24 ore) con sequenza libera. La tipologia del turno presa come riferimento per lo sviluppo di questa applicazione è il turno a ciclo continuo (sulle 24 ore) con sequenza libera, ovvero un turno che garantisce una copertura continua, includendo anche il servizio notturno, di tutti i giorni settimanali e annuali. In particolare, la giornata è stata suddivisa in tre turni: mattino (06.00 – 14.00), pomeriggio (14.00 – 22.00) e notte (22.00 – 06.00), e la durata di ogni turno è di 8 ore. La sequenza libera indica il fatto che non è identificabile una sequenza fissa e lo schema di presenze può non essere uniforme nelle settimane e nelle fasce orarie. Un'altra variabile è il tipo di contratto di lavoro che può essere: a tempo pieno, a tempo parziale di tipo orizzontale, a tempo parziale di tipo verticale o di lavoro supplementare. Nello sviluppo dell'applicazione è stato utilizzato il contratto di lavoro a tempo pieno per tutti gli infermieri, che equivale allo svolgimento di 36 ore settimanali e prevede 32 giorni di ferie annuali. Considerato che i turni sono di 8 ore, è evidente che l'infermiere non potrà svolgere esattamente le 36 ore settimanali (poiché 36 non è multiplo di 8). Quindi il programma controlla che gli infermieri abbiano svolto le ore da contratto, prendendo in considerazione un periodo più ampio, precisamente 1872 ore in un anno.

Nel Decreto Legislativo n. 66/2003 vengono indicati i vincoli normativi da rispettare. Il primo vincolo da tenere in conto riguarda l'orario di lavoro la cui durata media non può superare, per un periodo di 7 giorni, le 48 ore. Il vincolo complementare a quello precedente è quello sui riposi settimanali: il lavoratore ha diritto ogni 7 giorni a un periodo di riposo di almeno 24 ore consecutive. Un altro vincolo riguarda il riposo giornaliero: il lavoratore ha diritto a 11 ore di riposo consecutivo ogni 24 ore. Questo vincolo ha un riscontro sull'assegnazione dei turni lavorativi: esso determina che non si possa assegnare il turno del mattino ad un infermiere che il giorno precedente ha svolto il turno del pomeriggio né assegnare il turno del mattino o pomeriggio ad un infermiere che il giorno precedente ha svolto il turno della notte.

Oltre al rispetto dei vincoli sopracitati, il programma verificherà che ogni infermiere abbia effettuato le ore lavorative espresse nel contratto. Viene quindi calcolato il debito orario

annuale effettivo di ogni infermiere e confrontato con il debito orario annuale teorico. Quest'ultimo equivale al prodotto delle ore lavorative settimanali, stabilite da contratto, per il numero di settimane annue: 36 x 52 = 1872 ore/anno (vengono inclusi anche i 32 giorni di ferie).

Invece, al fine di ottenere un orario equo per gli infermieri, il programma verifica che a ciascuno di essi sia assegnato lo stesso numero di mattini, pomeriggi, notti, riposi e riposi nei giorni festivi (vengono considerati giorni festivi anche i sabati).

Nell'applicazione è stato preso come reparto di riferimento un reparto di medicina generale con venti pazienti in cui lavorano undici infermieri aventi contratto a tempo pieno e con copertura completa con sequenza libera suddivisa su tre turni. Questo reparto necessita che ogni giorno tre infermieri lavorino durante il turno del mattino, due durante il turno del pomeriggio e uno durante il turno della notte.

Di base l'applicazione funziona per il reparto appena indicato, ma basterà modificare nel database il numero di infermieri necessari nei diversi turni e il numero degli infermieri complessivi: il programma è in grado di adattarsi a qualunque reparto ospedaliero.

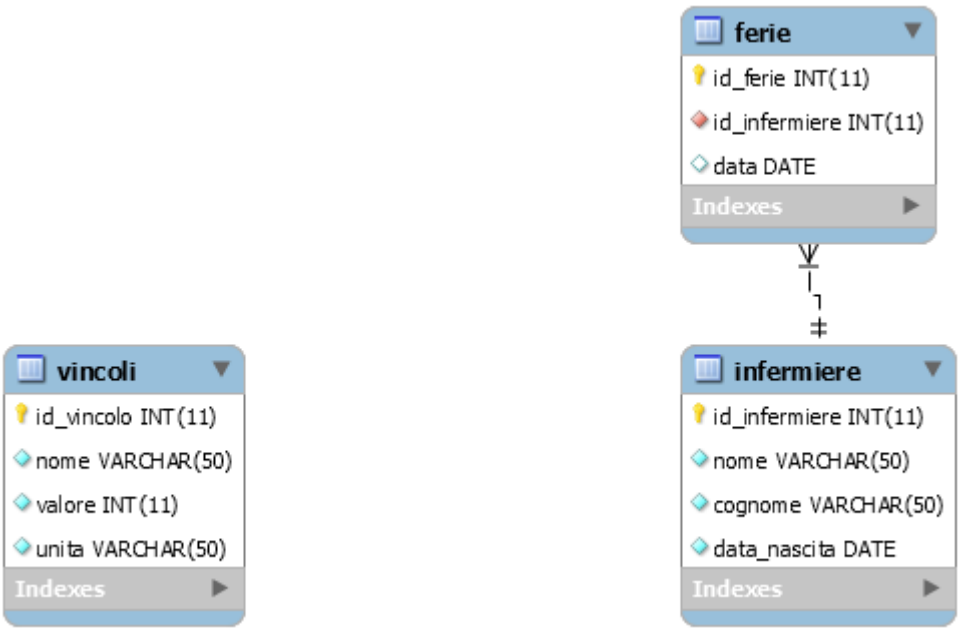
Nel database, oltre agli infermieri e ai vincoli del reparto, sono già inserite (per comodità dell'utente) le ferie richieste dagli infermieri, ma potranno essere tutte modificate attraverso finestra di gestione delle ferie del programma. Per una migliore gestione delle ferie, è stato inserito un vincolo all'interno del programma secondo cui, nello stesso giorno, al massimo due infermieri possono richiedere le ferie.

Partendo dai vincoli del reparto, dagli infermieri e dalle loro richieste di ferie, l'applicazione genererà la programmazione annuale (dal 01 settembre 2019 al 31 agosto 2020) dei turni. Sarà poi possibile visualizzare l'orario mensilmente e lo si potrà salvare in un file di testo. L'ultima finestra del programma comprende due grafici statistici per ciascun infermiere. Il primo mostra il numero di turni a lui assegnati suddivisi in mattini, pomeriggi e notti; il secondo illustra il numero di riposi durante i giorni feriali e festivi.

Trovare la soluzione perfetta che permetta di avere un trattamento identico per tutti gli infermieri è praticamente impossibile. L'applicazione trova però un'ottima soluzione che assegna agli infermieri un equo orario lavorativo, mantenendo un piccolo margine di differenza. A causa di questo piccolo margine, l'applicazione potrebbe trovare più soluzioni, ma la ricerca termina quando si ottiene la prima soluzione che soddisfa i vincoli imposti: questo riduce chiaramente il tempo di ricerca.

3. Descrizione del data-set utilizzato per l'analisi

Il data-set utilizzato è stato creato appositamente per l'applicazione. Contiene l'elenco degli infermieri con i loro dati anagrafici (dati inventati), la lista di tutti i giorni di ferie richiesti dagli infermieri (32 giorni per ogni infermiere) e l'elenco dei vincoli normativi e del reparto considerato.



La tabella dei vincoli include:

- le ferie annue;
- le ore lavorative settimanali contrattuali;
- il massimo delle ore lavorative settimanali;
- il massimo dei giorni lavorativi settimanali;
- il minimo di ore di riposo consecutive dopo la fine di un turno;
- il minimo di giorni di riposo settimanali;
- il debito orario teorico annuale;
- la durata del turno lavorativo;
- il numero di infermieri nel turno del mattino;
- il numero di infermieri nel turno del pomeriggio;
- il numero di infermieri nel turno della notte.

4. Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati

L'applicazione è stata sviluppata in linguaggio di programmazione Java, con l'ausilio di JavaFX per la gestione dell'interfaccia grafica. Sono stati implementati il pattern MVC (Model View Controller), il pattern DAO (Data Access Object) e lo schema ORM (Object Relational Mapping).

La struttura del software è suddivisa in tre packages:

- Controller: contiene la classe Main per l'avvio dell'applicazione, la classe controller TurniInfermieriController che permette l'interazione tra l'utente e il file (TurniInfermieri.fxml) che definisce l'interfaccia grafica e TurniInfermieri.fxml stesso;

- DB: contiene la classe DBConnect utilizzata per la connessione al database e la classe TurniInfermieriDAO utilizzata per ottenere i dati presenti all'interno del database e caricarli all'interno dell'ambiente java;

- Model: contiene la classe Model che gestisce tutta la logica applicativa del software, le classi JavaBean: Infermiere e Ferie e le classi per la visualizzazione dei dati nelle tabelle e nei grafici InfermiereTurni e StatisticheInfermiere.

La funzione principale dell'applicazione è generare la programmazione dei turni lavorativi degli infermieri. Questa viene generata attraverso una funzione ricorsiva che permette di creare la soluzione giorno dopo giorno. Sono inoltre presenti molti controlli che permettono di trovare una soluzione corretta e che sia la migliore possibile.

Per creare l'orario, il programma prende in considerazione un giorno per volta. Le fasi per l'assegnazione dei turni sono le seguenti:

1. trovare i candidati infermieri che possono essere assegnati al turno del mattino;
2. scegliere, tra quelli disponibili, gli infermieri che hanno svolto tra tutti meno turni del mattino nei giorni precedenti e assegnare loro il turno;
3. ripetere le prime due operazioni per i turni del pomeriggio e della notte, in modo da coprire l'intera giornata secondo le richieste del reparto;
4. se il programma non trova candidati sufficienti per il turno del pomeriggio o della notte tornare all'assegnazione del turno precedente e assegnarlo ad altri infermieri. Se non viene trovata una soluzione valida, tornare al giorno precedente e assegnare i turni ad altri candidati;
5. dopo aver trovato una soluzione per il giorno considerato, ripetere le operazioni per il giorno successivo, e così via fino al completamento dell'orario annuale.

Se il giorno preso in considerazione è festivo, il programma agisce diversamente: ordina gli infermieri disponibili per i turni lavorativi in base a chi ha ottenuto più riposi nei precedenti giorni di festività. Avviene dunque che chi ha già svolto più riposi nei giorni festivi, sarà il primo ad essere assegnato ai turni lavorativi di questo giorno.

Questi diversi ordinamenti servono per ottimizzare i tempi di ricerca: l'algoritmo non trova soluzioni casuali ma già dai primi risultati otteniamo delle soluzioni che seguono una certa logica, ovvero quella di non creare disparità di trattamento. Una volta trovato l'orario annuale, il programma controlla che esso garantisca un trattamento equo a tutti gli infermieri (ovvero che ciascun infermiere svolga lo stesso numero di turni al mattino, pomeriggio, notte, e che abbia gli stessi giorni di riposo e di riposo nei giorni festivi), lasciando un piccolo margine di accettazione dai valori ottimali. Inoltre, viene verificato che ciascun infermiere abbia svolto le ore annuali stabilite da contratto, ma poiché è molto difficile garantirne la perfetta esattezza, il programma ammette un margine di differenza del 2%.

Di seguito vengono riportate le diverse fasi per la generazione dell'orario: la funzione ricorsiva principale e le funzioni più importanti che vengono richiamate da essa.

Inizializzazione e richiamo della funzione ricorsiva per la generazione dell'orario

```
// inizializzazione e richiamo della funzione ricorsiva per la generazione dell'orario
public Map<LocalDate, Map<Infermiere, String>> generaOrario() {

    // inizializzazione delle statistiche di ogni infermiere
    for (Infermiere inf : infermieri) {
        inf.setNumero_mattine(0);
        inf.setNumero_pomeriggi(0);
        inf.setNumero_notti(0);
        inf.setNumero_riposi(0);
        inf.setNumero_riposi_festivita(0);
    }
     // mappa che conterrà la soluzione trovata
    soluzione = new TreeMap<LocalDate, Map<Infermiere, String>>();
    // variabile booleana che indica se la soluzione è stata trovata o meno
    trovata = false;
    // soluzione parziale per la ricerca della soluzione completa
    Map<LocalDate, Map<Infermiere, String>> parziale = new TreeMap<LocalDate, Map<Infermiere, String>>();
    // lista in cui vengono salvate le statistiche di tutti gli infermieri
    stat = new ArrayList<StatisticheInfermiere>();

    // inizializzazione orario

    LocalDate data = inizio;

    Map<Infermiere, String> turni = new HashMap<Infermiere, String>();

    for (Infermiere i : infermieri)
        turni.put(i, null);

    // inizializzazione di parziale con tutte le date e tutti gli infermieri con i turni settati a null
    // inizializzazione di soluzione solo con tutte le date
    while (data.isBefore(fine)) {
        parziale.put(data, new HashMap<Infermiere, String>(turni));
        soluzione.put(data, new HashMap<Infermiere, String>());
        data = data.plusDays(1);
    }
    // identity map degli infermieri
    Map<Integer, Infermiere> infermieriMap = new HashMap<Integer, Infermiere>();

    for (Infermiere i : infermieri)
        infermieriMap.put(i.getId_infermiere(), i);

    // inserimento di tutte le ferie degli infermieri nella soluzione parziale
    for (Ferie f : ferie) {
        Map<Infermiere, String> map = parziale.get(f.getData());
        map.put(infermieriMap.get(f.getId_infermiere()), "Ferie");
    }

    // chiamata della funzione ricorsiva
    calcolaOrario(parziale, inizio);

    if (!trovata)
        soluzione = null;

    return soluzione;

}
```

Funzione ricorsiva che genera l'orario

```
// calcola orario attraverso la ricorsione
// i parametri sono la soluzione parziale e la data che rappresenta il livello della ricorsione
private void calcolaOrario(Map<LocalDate, Map<Infermiere, String>> parziale, LocalDate data) {
    // se la soluzione è stata trovata esco dalla ricorsione
    if (trovata)
        return;
    // varaiabile che indicica se il giorno corrente è un feriale o festivo (il sabato viene considerato come giorno festivo)
    boolean dataFest;

    if (data.getDayOfWeek().getValue() == 6 || data.getDayOfWeek().getValue() == 7 || festivita.contains(data))
        dataFest = true;
    else
        dataFest = false;

    // se non ho ancora completato tutto il calendario
    if (data.isBefore(fine)) {
        // ottengo tutti gli infermieri candidati a lavorare nel turno del mattino in questa giornata
        List<Infermiere> candidatiMattino = trovaCandidatiMattino(data, parziale);

        if (dataFest) {
            // ordino gli infermieri in base al numero di riposi nei giorni fetivi che hanno già fatto
            Collections.sort(candidatiMattino, Infermiere.riposiFestivitaComparator());
        }
        else {
            // ordino gli infermieri in base al numero di turni di mattina che hanno già fatto
            Collections.sort(candidatiMattino,Infermiere.mattineComparator());
        }
        // se c'è troppa differenza tra il numero di riposi tra due infermieri
        if (!diffRiposi()) {
            // ordino gli infermieri in base al numero di riposi che hanno già fatto
            Collections.sort(candidatiMattino, Infermiere.riposiComparator());
        }
        //ottengo tutte le possibili combinazioni di infermieri che possono lavorare nel turno del mattino
        List<List<Infermiere>> combMattino = this.subsets(candidatiMattino, numero_infermieri_turno_mattino);

        // per ogni combinazione di infermieri che possono lavorare nel turno del mattino
        for (List<Infermiere> infMat : combMattino) {
            // assegno agli infermieri il turno del mattino
            for (Infermiere i : infMat) {
                parziale.get(data).put(i, "Mattino");
                i.setNumero_mattine(i.getNumero_mattine() + 1);
            }
            // ottengo tutti gli infermieri candidati a lavorare nel turno del pomeriggio in questa giornata
            List<Infermiere> candidatiPomeriggio = this.trovaCandidatiPomeriggio(data, parziale);

            if (dataFest) {
                // ordino gli infermieri in base al numero di riposi nei giorni fetivi che hanno già fatto
                Collections.sort(candidatiPomeriggio, Infermiere.riposiFestivitaComparator());
            }
            else {
                // ordino gli infermieri in base al numero di turni di pomeriggio che hanno già fatto
                Collections.sort(candidatiPomeriggio, Infermiere.pomeriggiComparator());
            }
            // se c'è troppa differenza tra il numero di riposi tra due infermieri
            if (!diffRiposi()) {
                 // ordino gli infermieri in base al numero di riposi che hanno già fatto
                Collections.sort(candidatiPomeriggio, Infermiere.riposiComparator());
            }

            //ottengo tutte le possibili combinazioni di infermieri che possono lavorare nel turno del pomeriggio
            List<List<Infermiere>> combPomeriggio = this.subsets(candidatiPomeriggio, numero_infermieri_turno_pomeriggio);
            // per ogni combinazione di infermieri che possono lavorare nel turno del pomeriggio
            for (List<Infermiere> infPom : combPomeriggio) {
                // assegno agli infermieri il turno del pomeriggio
                for (Infermiere i : infPom) {
                    parziale.get(data).put(i, "Pomeriggio");
                    i.setNumero_pomeriggi(i.getNumero_pomeriggi() + 1);
                }
                // ottengo tutti gli infermieri candidati a lavorare nel turno della notte in questa giornata
                List<Infermiere> candidatiNotte = this.trovaCandidatiNotte(data, parziale);

                if (dataFest) {
                    // ordino gli infermieri in base al numero di riposi nei giorni fetivi che hanno già fatto
                    Collections.sort(candidatiNotte, Infermiere.riposiFestivitaComparator());
                }
                else {
                    // ordino gli infermieri in base al numero di turni della notte che hanno già fatto
                    Collections.sort(candidatiNotte, Infermiere.nottiComparator());
                }
                // se c'è troppa differenza tra il numero di riposi tra due infermieri
                if (!diffRiposi()) {
                    // ordino gli infermieri in base al numero di riposi che hanno già fatto
                    Collections.sort(candidatiNotte, Infermiere.riposiComparator());
                }
                //ottengo tutte le possibili combinazioni di infermieri che possono lavorare nel turno della notte
                List<List<Infermiere>> combNotte = this.subsets(candidatiNotte, numero_infermieri_turno_notte);
```

```
                // per ogni combinazione di infermieri che possono lavorare nel turno della notte
                for (List<Infermiere> infNot : combNotte) {
                    // assegno agli infermieri il turno della notte
                    for (Infermiere i : infNot) {
                        parziale.get(data).put(i, "Notte");
                        i.setNumero_notti(i.getNumero_notti() + 1);
                    }
                    // assegno il giorno di riposo a tutti gli infermieri a cui non è stato assegnato un turno lavorativo e
                    // che non sono in ferie
                    for (Infermiere i : infermieri) {
                        if (parziale.get(data).get(i) == null) {
                            parziale.get(data).put(i, "Riposo");
                            i.setNumero_riposi(i.getNumero_riposi() + 1);
                            if (dataFest)
                                i.setNumero_riposi_festivita(i.getNumero_riposi_festivita() + 1);
                        }
                    }
                    // richiamo la funzione ricorsiva avendo assegnato tutti i turni in questa giornata e indicando di
                    //continuare per il giorno successivo
                    this.calcolaOrario(parziale, data.plusDays(1));

                    // rimuovo i giorni di riposo assegnati
                    for (Infermiere i : infermieri) {
                        if (parziale.get(data).get(i) == "Riposo") {
                            parziale.get(data).put(i, null);
                            i.setNumero_riposi(i.getNumero_riposi() - 1);
                            if (dataFest)
                                i.setNumero_riposi_festivita(i.getNumero_riposi_festivita() - 1);
                        }
                    }

                    // rimuovo i turni della notte assegnati
                    for (Infermiere i : infermieri) {
                        if (parziale.get(data).get(i) == "Notte") {
                            parziale.get(data).put(i, null);
                            i.setNumero_notti(i.getNumero_notti() - 1);
                        }
                    }
                }
                // rimuovo i turni del pomeriggio assegnati
                for (Infermiere i : infermieri) {
                    if (parziale.get(data).get(i) == "Pomeriggio") {
                        parziale.get(data).put(i, null);
                        i.setNumero_pomeriggi(i.getNumero_pomeriggi() - 1);
                    }
                }
            }
            // rimuovo i turni del mattino assegnati
            for (Infermiere i : infermieri) {
                if (parziale.get(data).get(i) == "Mattino") {
                    parziale.get(data).put(i, null);
                    i.setNumero_mattine(i.getNumero_mattine() - 1);
                }
            }
        }
    }
    // quando completo tutto il calendario
    else if (data.isEqual(fine)) {

        if (!trovata) {
            // controllo che la soluzione trovata sia accettabile
            if (!controlloSoluzione())
                return;
            // trovata soluzione che rispetti i vincoli di controllo
            trovata = true;
            LocalDate d = inizio;
            // salvo la soluzione finale
            while (d.isBefore(fine)) {
                soluzione.put(d, new HashMap<Infermiere, String>(parziale.get(d)));
                d = d.plusDays(1);
            }
            // salvo le statistiche degli infermieri
            for (Infermiere i : infermieri)
                stat.add(new StatisticheInfermiere(i, i.getNumero_riposi(), i.getNumero_mattine(), i.getNumero_pomeriggi(),
                        i.getNumero_notti(), i.getNumero_riposi_festivita()));
        }
    }
}
```

Funzione per la ricerca degli infermieri candidati all'assegnazione del turno del mattino

```
// trovo quali infermieri possono essere assegnati al turno del mattino nella data passata come parametro
// andando a controllare la soluzione parziale fino a quel giorno
private List<Infermiere> trovaCandidatiMattino(LocalDate data, Map<LocalDate, Map<Infermiere, String>> parziale) {

    List<Infermiere> infMat = new ArrayList<Infermiere>();

    // primo giorno
    if (data.equals(inizio)) {
        for (Infermiere i : infermieri) {
            // tutti gli infermieri a cui non è ancora stato assegnato un turno sono possibili candidati
            // a parte quelli a cui sono state assegnate le ferie
            if (parziale.get(data).get(i) == null)
                infMat.add(i);
        }
    }
    // tra il primo e il settimo giorno (esclusi)
    else if (data.isBefore(inizio.plusDays(max_giorni_lavorativi_settimanali))
            && data.isAfter(inizio)) {
        for (Infermiere i : infermieri) {
            // sono candidati gli infermieri a cui non è ancora stato assegnato il turno e
            // il cui giorno precedente non è stato assegnato il turno di pomeriggio, il turno di notte o le ferie
            if (parziale.get(data).get(i) == null && !parziale.get(data.minusDays(1)).get(i).equals("Notte")
                    && !parziale.get(data.minusDays(1)).get(i).equals("Pomeriggio"))
                infMat.add(i);
        }
    }
    // in tutti gli altri giorni
    else {
        int cont;

        for (Infermiere i : infermieri) {
            cont = 0;

            for (int j = max_giorni_lavorativi_settimanali; j > 0; j--) {
                // conto quante volte hanno lavorato gli infermieri nei sei giorni precedenti
                if (!parziale.get(data.minusDays(j)).get(i).equals("Riposo")
                        && !parziale.get(data.minusDays(j)).get(i).equals("Ferie"))
                    cont++;
            }
            // sono candidati gli infermieri a cui non è ancora stato assegnato un turno e che nei sei giorni
            // precedenti non hanno lavorato per almeno un giorno, che nel giorno precedente non sono stati assegnati
            // al turno del pomeriggio o della notte, e che in questo giorno non abbiano le ferie
            if (cont < max_giorni_lavorativi_settimanali && parziale.get(data).get(i) == null
                    && !parziale.get(data.minusDays(1)).get(i).equals("Notte")
                    && !parziale.get(data.minusDays(1)).get(i).equals("Pomeriggio"))
                infMat.add(i);
        }
    }
    return infMat;
}
```

Funzione per la ricerca degli infermieri candidati all'assegnazione del turno del pomeriggio

```
// trovo quali infermieri possono essere assegnati al turno del pomeriggio nella data passata come parametro
// andando a controllare la soluzione parziale fino a quel giorno
private List<Infermiere> trovaCandidatiPomeriggio(LocalDate data, Map<LocalDate, Map<Infermiere, String>> parziale) {

    List<Infermiere> infPom = new ArrayList<Infermiere>();

    // primo giorno
    if (data.equals(inizio)) {
        for (Infermiere i : infermieri) {
            // tutti gli infermieri a cui non è ancora stato assegnato un turno sono possibili candidati
            // a parte quelli a cui sono state assegnate le ferie
            if (parziale.get(data).get(i) == null)
                infPom.add(i);
        }
    }

    // tra il primo e il settimo giorno (esclusi)
    else if (data.isBefore(inizio.plusDays(max_giorni_lavorativi_settimanali))
            && data.isAfter(inizio)) {
        for (Infermiere i : infermieri) {
             // sono candidati gli infermieri a cui non è ancora stato assegnato il turno il cui giorno
            //  precendete non sono stati assegnati al turno della notte o le ferie
            if (parziale.get(data).get(i) == null && !parziale.get(data.minusDays(1)).get(i).equals("Notte"))
                infPom.add(i);
        }
    }
    // in tutti gli altri giorni
    else {
        int cont;

        for (Infermiere i : infermieri) {
            cont = 0;
```

```
            for (int j = max_giorni_lavorativi_settimanali; j > 0; j--) {
                // conto quante volte hanno lavorato gli infermieri nei sei giorni precedenti
                if (!parziale.get(data.minusDays(j)).get(i).equals("Riposo")
                        && !parziale.get(data.minusDays(j)).get(i).equals("Ferie"))
                    cont++;
            }
            // sono candidati tutti gli infemieri a cui non è ancora stato assegnato un turno e che nei sei giorni
            // precedenti non hanno lavorato per almeno un giorno, che nel giorno precedente
            //  non siano stati assegnati al turno di notte e che in questo giorno non abbiano le ferie
            if (cont < max_giorni_lavorativi_settimanali && parziale.get(data).get(i) == null
                    && !parziale.get(data.minusDays(1)).get(i).equals("Notte"))
                infPom.add(i);
        }
    }
    return infPom;
}
```

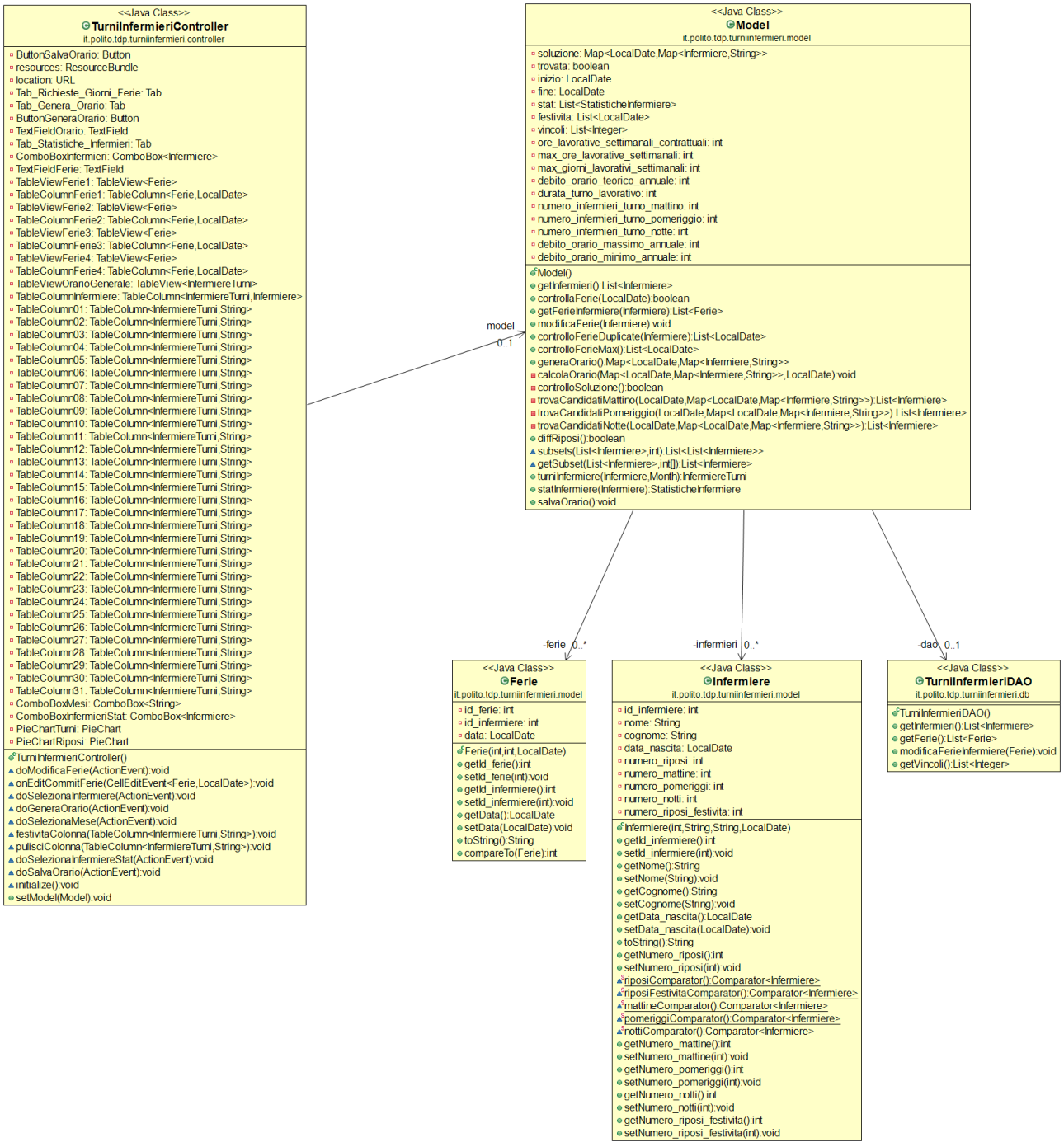
Funzione per la ricerca degli infermieri candidati all'assegnazione del turno della notte

```
// trovo quali infermieri possono essere assegnati al turno della notte nella data passata come parametro
// andando a controllare la soluzione parziale fino a quel giorno
private List<Infermiere> trovaCandidatiNotte(LocalDate data, Map<LocalDate, Map<Infermiere, String>> parziale) {

    List<Infermiere> infNot = new ArrayList<Infermiere>();

    // primo giorno
    if (data.equals(inizio)) {
        for (Infermiere i : infermieri) {
            // tutti gli infermieri a cui non è ancora stato assegnato un turno sono possibili candidati
            // a parte quelli a cui sono state assegnate le ferie
            if (parziale.get(data).get(i) == null)
                infNot.add(i);
        }
    }
    // tra il primo e il settimo giorno (esclusi)
    else if (data.isBefore(LocalDate.of(2019, Month.SEPTEMBER, 7))
            && data.isAfter(inizio)) {
        for (Infermiere i : infermieri) {
            // tutti gli infermieri a cui non è ancora stato assegnato un turno sono possibili candidati
            // a parte quelli a cui sono state assegnate le ferie
            if (parziale.get(data).get(i) == null)
                infNot.add(i);
        }
    }
    // in tutti gli altri giorni
    else {
        int cont;

        for (Infermiere i : infermieri) {
            cont = 0;

            for (int j = max_giorni_lavorativi_settimanali; j > 0; j--) {
                if (!parziale.get(data.minusDays(j)).get(i).equals("Riposo")
                        // conto quante volte hanno lavorato gli infermieri nei sei giorni precedenti
                        && !parziale.get(data.minusDays(j)).get(i).equals("Ferie"))
                    cont++;
            }
            // sono candidati tutti gli infermieri a cui non è stato ancora assegnato un turno e che nei sei giorni
            // precedenti non hanno lavorato per almeno un giorno e che in questo giorno non abbiano le ferie
            if (cont < max_giorni_lavorativi_settimanali && parziale.get(data).get(i) == null)
                infNot.add(i);
        }
    }
    return infNot;
}
```

Funzione per il controllo della soluzione ottenuta, viene controllato il debito orario annuale e l'equità di trattamento tra gli infermieri

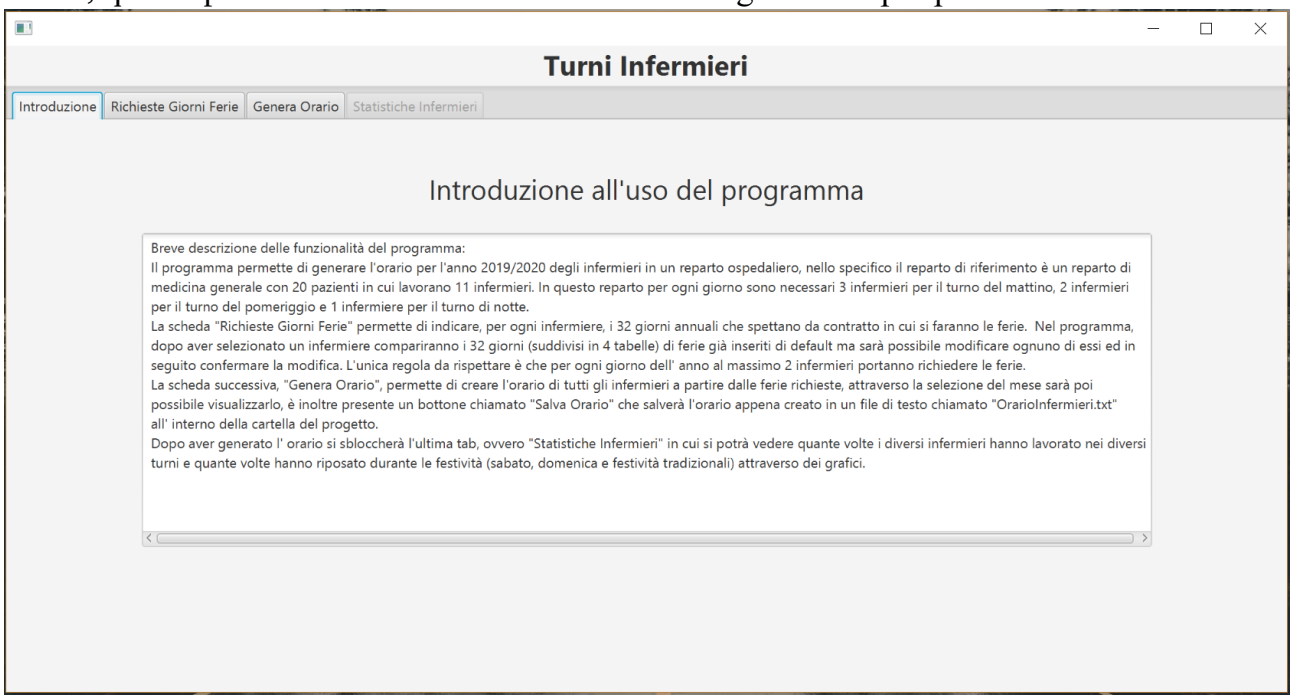
```
private boolean controlloSoluzione() {

    // variabili per la ricerca del numero minimo e massimo di turni e riposi per gli infermieri
    int oreTot = 0;
    int maxMat = 0;
    int minMat = Integer.MAX_VALUE;
    int maxPom = 0;
    int minPom = Integer.MAX_VALUE;
    int maxNot = 0;
    int minNot = Integer.MAX_VALUE;
    int maxRip = 0;
    int minRip = Integer.MAX_VALUE;
    int maxRipFest = 0;
    int minRipFest = Integer.MAX_VALUE;

    for (Infermiere inf : infermieri) {
        // ore totali lavorative annuali per infermiere (vengono considerati anche i 32 giorni di ferie annuali in questo conteggio)
        oreTot = (inf.getNumero_mattine() + inf.getNumero_pomeriggi() + inf.getNumero_notti() + 32) * durata_turno_lavorativo;
        // controllo se l'infermiere ha lavorato per il numero giusto di ore durante l'anno (con uno scarto del 2%)
        if (oreTot > debito_orario_massimo_annuale || oreTot < debito_orario_minimo_annuale)
            return false;

        if (inf.getNumero_mattine() < minMat)
            minMat = inf.getNumero_mattine(); // numero minimo di mattine assegnate ad un infermiere
        if (inf.getNumero_mattine() > maxMat)
            maxMat = inf.getNumero_mattine(); // numero massimo di mattine assegnate ad un infermiere
        if (inf.getNumero_pomeriggi() < minPom)
            minPom = inf.getNumero_pomeriggi(); // numero minimo di pomeriggi assegnati ad un infermiere
        if (inf.getNumero_pomeriggi() > maxPom)
            maxPom = inf.getNumero_pomeriggi(); // numero massimo di pomeriggi assegnati ad un infermiere
        if (inf.getNumero_notti() < minNot)
            minNot = inf.getNumero_notti(); // numero minimo di notti assegnate ad un infermiere
        if (inf.getNumero_notti() > maxNot)
            maxNot = inf.getNumero_notti(); // numero massimo di notti assegnate ad un infermiere
        if (inf.getNumero_riposi() < minRip)
            minRip = inf.getNumero_riposi(); // numero minimo di risposi assegnati ad un infemiere
        if (inf.getNumero_riposi() > maxRip)
            maxRip = inf.getNumero_riposi(); // numero massimo di risposi assegnati ad un infermiere
        if (inf.getNumero_riposi_festivita() < minRipFest)
            minRipFest = inf.getNumero_riposi_festivita(); // numero minimo di riposi durante giorni festivi assegnati ad un infermiere
        if (inf.getNumero_riposi_festivita() > maxRipFest)
            maxRipFest = inf.getNumero_riposi_festivita(); // numero massimo di riposi durante giorni festivi assegnati ad un infermiere
    }

    // vincoli che permettono di trovare una soluzione che sia molto equa tra gli infermieri
    if (maxMat - minMat > 12) // controllo differenza massima del numero di assegnazioni del turno del mattina tra gli infermieri
        return false;
    if (maxPom - minPom > 8) // controllo differenza massima del numero di assegnazioni del turno del pomeriggio tra gli infermieri
        return false;
    if (maxNot - minNot > 4) // controllo differenza massima del numero di assegnazioni del turno della notte tra gli infermieri
        return false;
    if (maxRip - minRip > 5) // controllo differenza massima del numero di assegnazioni di riposi tra gli infermieri
        return false;
    if (maxRipFest - minRipFest > 3) // controllo differenza massima del numero di assegnazioni di riposi durante giorni festivi tra gli infermieri
        return false;

    return true;
}
```

5. Diagramma delle classi delle parti principali dell'applicazione

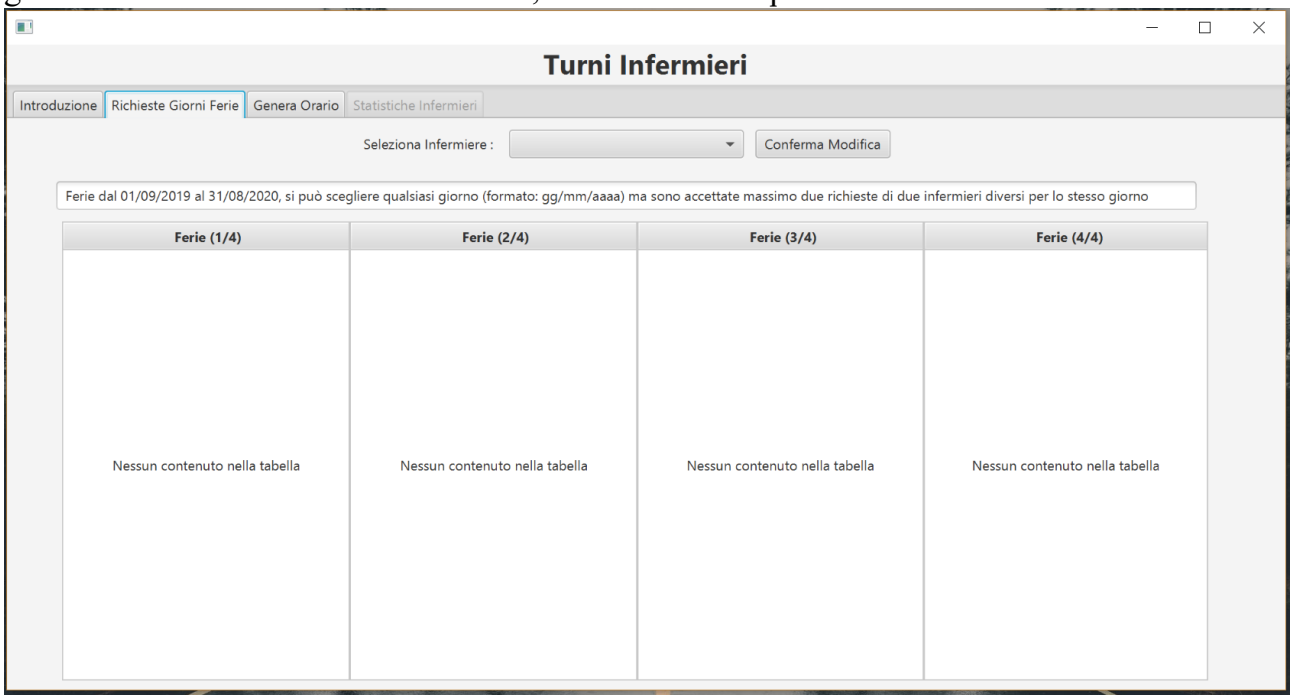


6. Alcune videate dell'applicazione realizzata e link al video dimostrativo del software

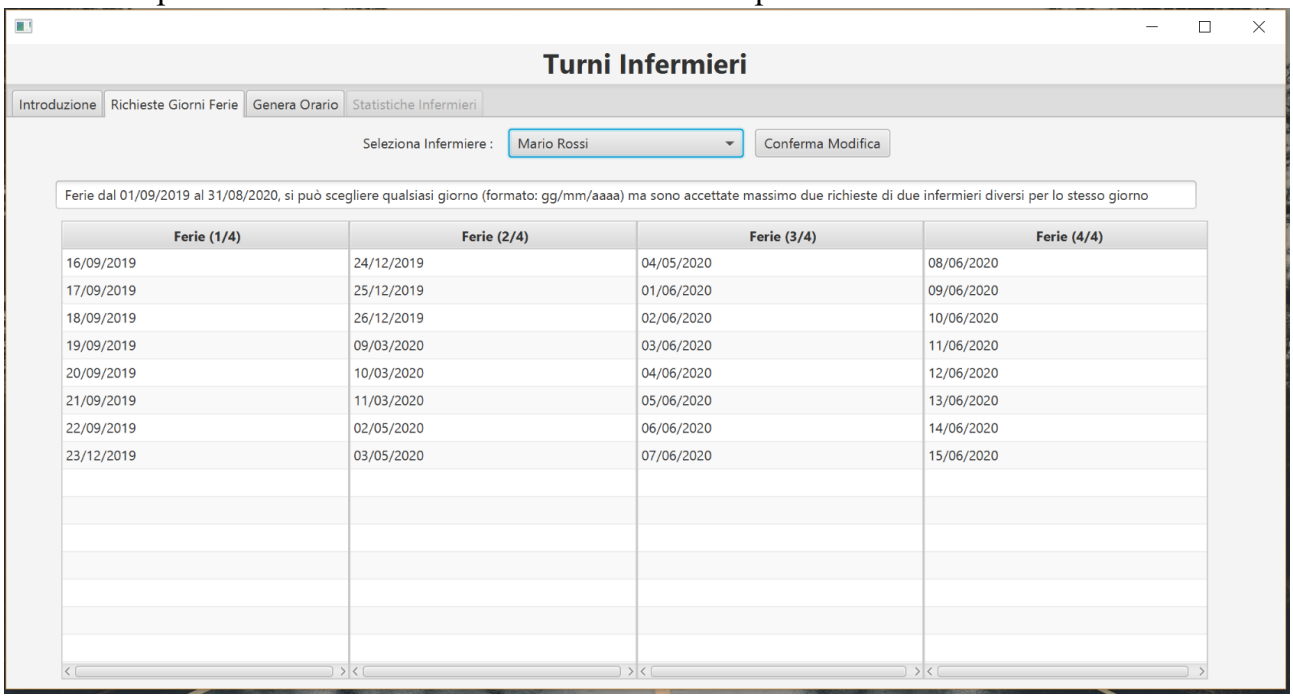
Finestra di introduzione: vengono spiegate brevemente le funzionalità dell'applicazione. È possibile passare sia alla finestra di gestione ferie sia direttamente a quella di generazione orario, questo perché le ferie inserite di default sono già valide per poter creare l'orario.



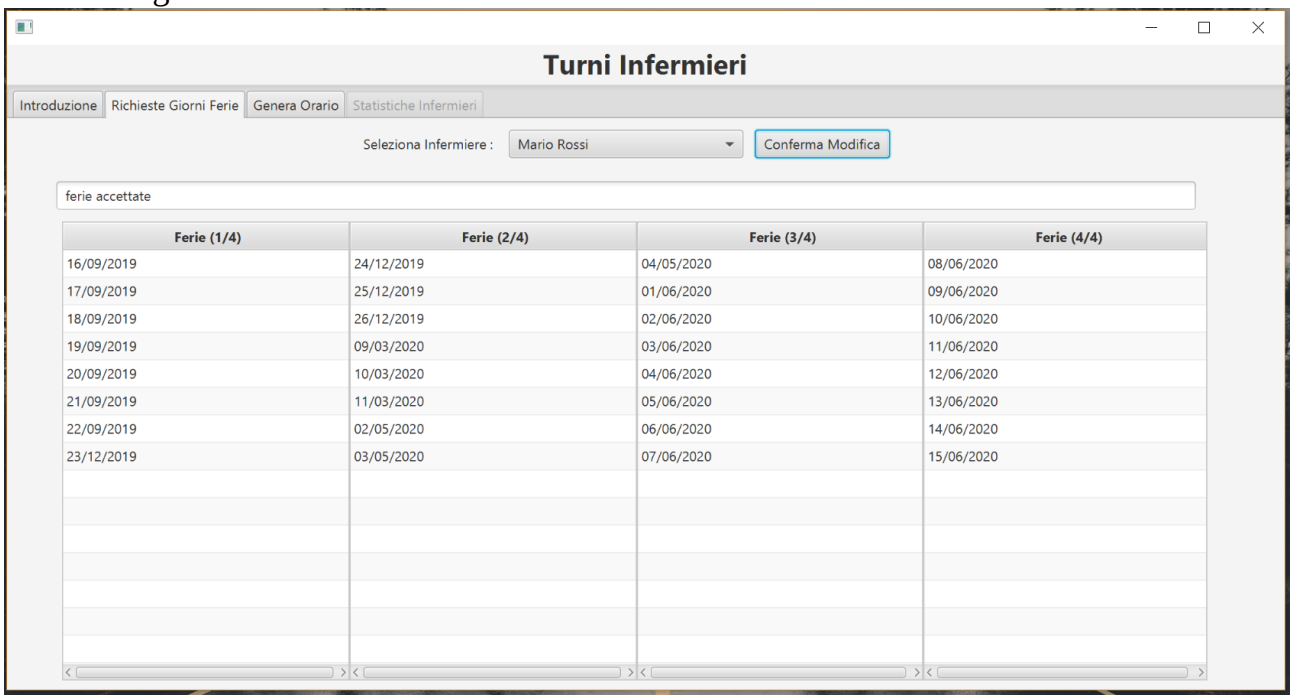
Finestra gestione ferie: selezionando un infermiere dal menu a tendina compariranno i 32 giorni di ferie richiesti dall'infermiere, suddivisi nelle quattro tabelle.



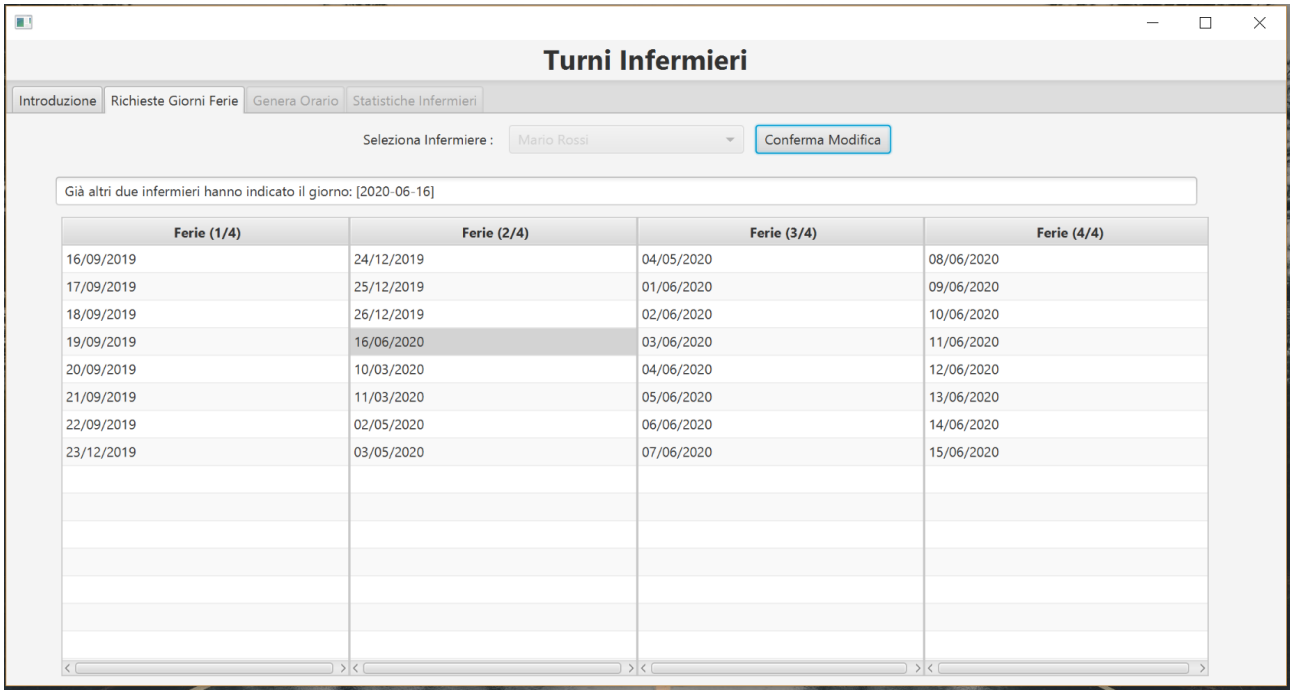
Finestra gestione ferie con tabella riempita dalle ferie dell'infermiere selezionato: è possibile modificare qualsiasi dei giorni di ferie, l'unico vincolo è che al massimo due infermieri diversi possono richiedere le ferie nello stesso giorno. Dopo averle modificate è necessario premere sul bottone "Conferma Modifica" per verificare che siano valide.



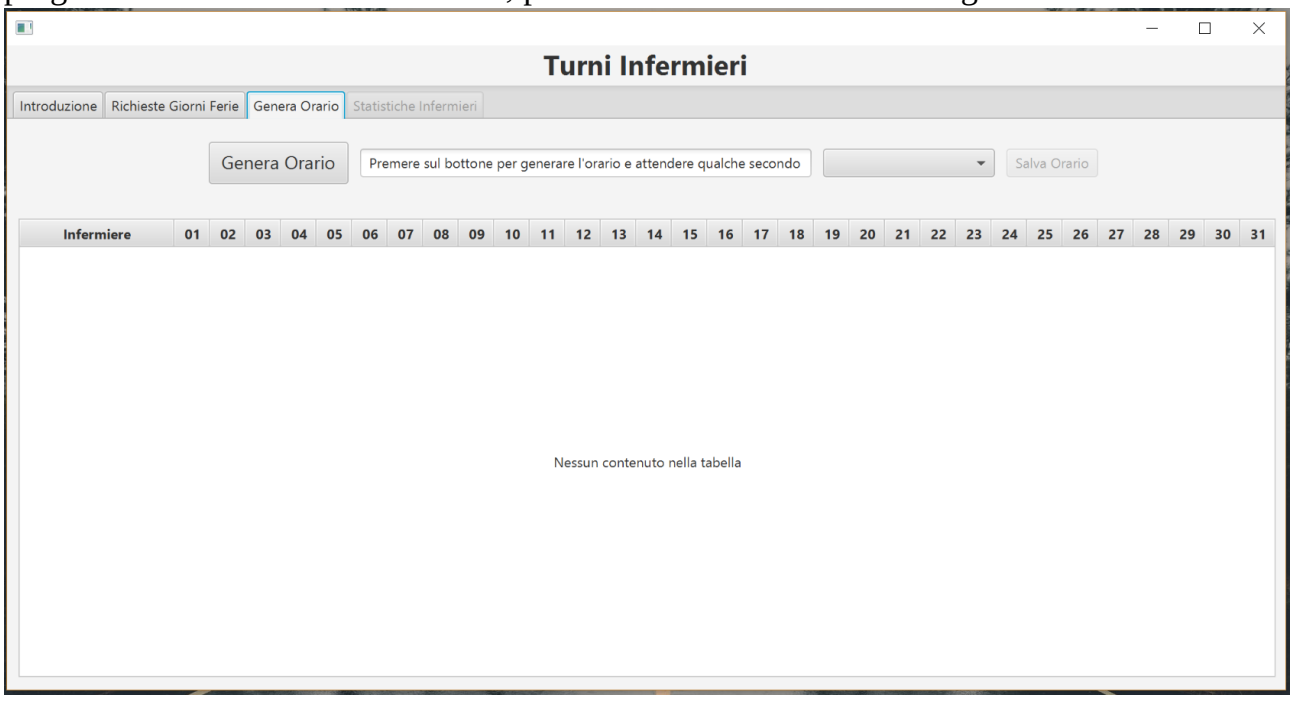
Finestra gestione ferie in seguito all'accettazione della modifica delle ferie: in questo caso sono state accettate le modifiche apportate alle ferie. È quindi possibile passare alla finestra di generazione orario.



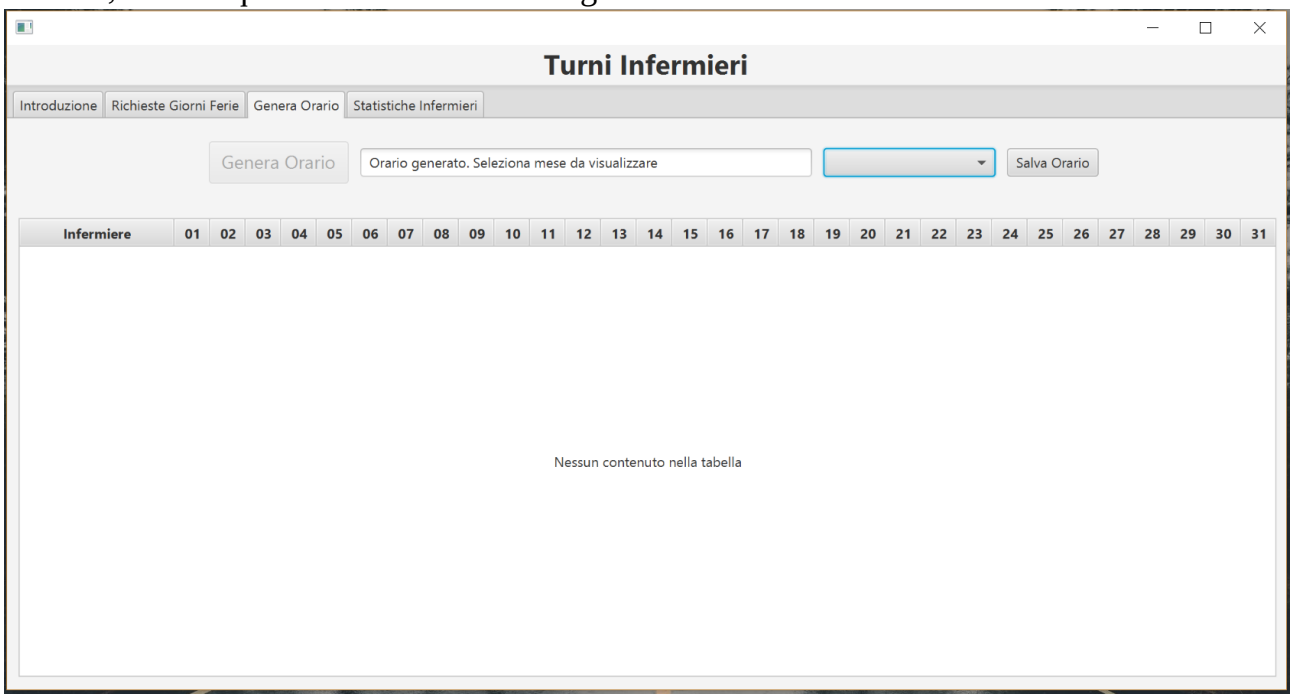
Finestra gestione ferie in seguito alla negazione della modifica delle ferie: in questo caso non sono state accettate le modifiche apportate alle ferie. Sarà necessario modificarle nuovamente fino a quando non diventeranno valide; fino a quel momento tutte le altre funzioni saranno bloccate.



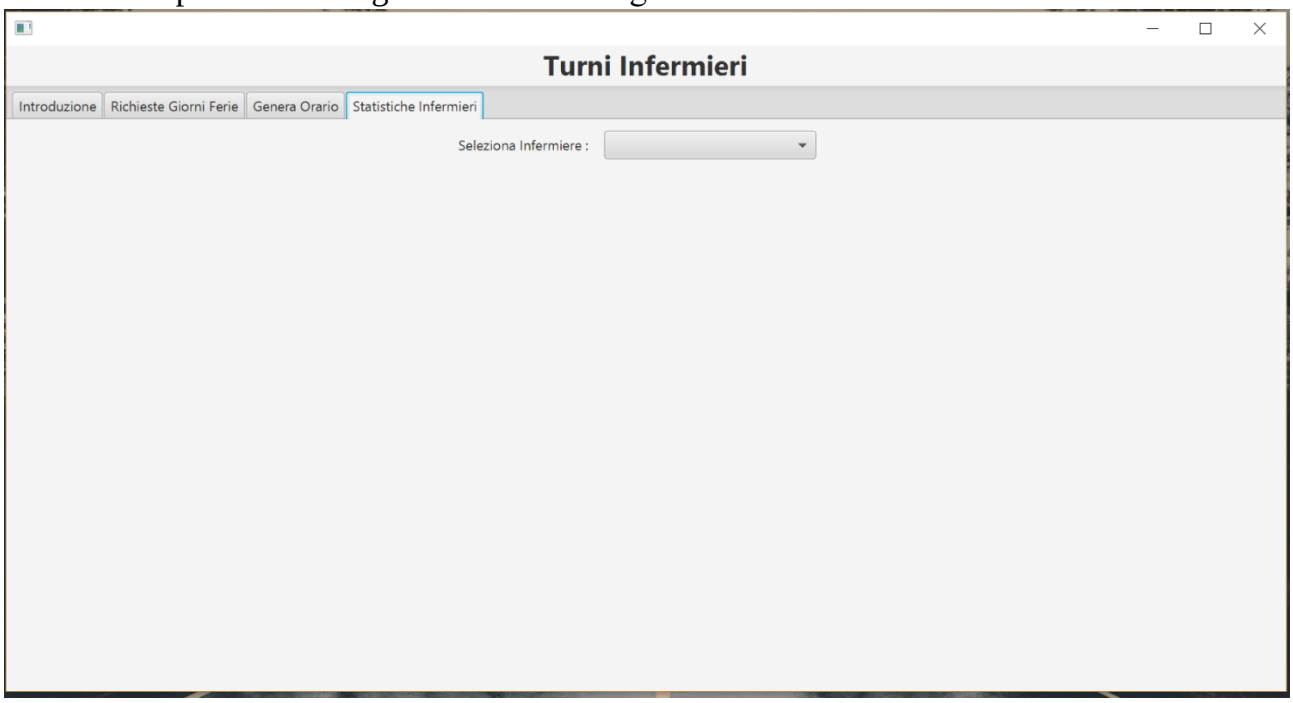
Finestra generazione orario: premendo sul bottone “Genera Orario” verrà generata la programmazione dei turni lavorativi, partendo dalle ferie richieste dagli infermieri.



Finestra generazione orario dopo aver generato l'orario: dopo aver aspettato qualche secondo, l'orario verrà creato. Sarà quindi possibile visualizzarlo mese per mese attraverso l'apposito menu a tendina. Inoltre, premendo sul bottone "Salva Orario" si potrà salvare l'orario annuale in un file di testo. Dopo aver generato l'orario si potrà accedere all'ultima finestra, ovvero quella delle statistiche degli infermieri.



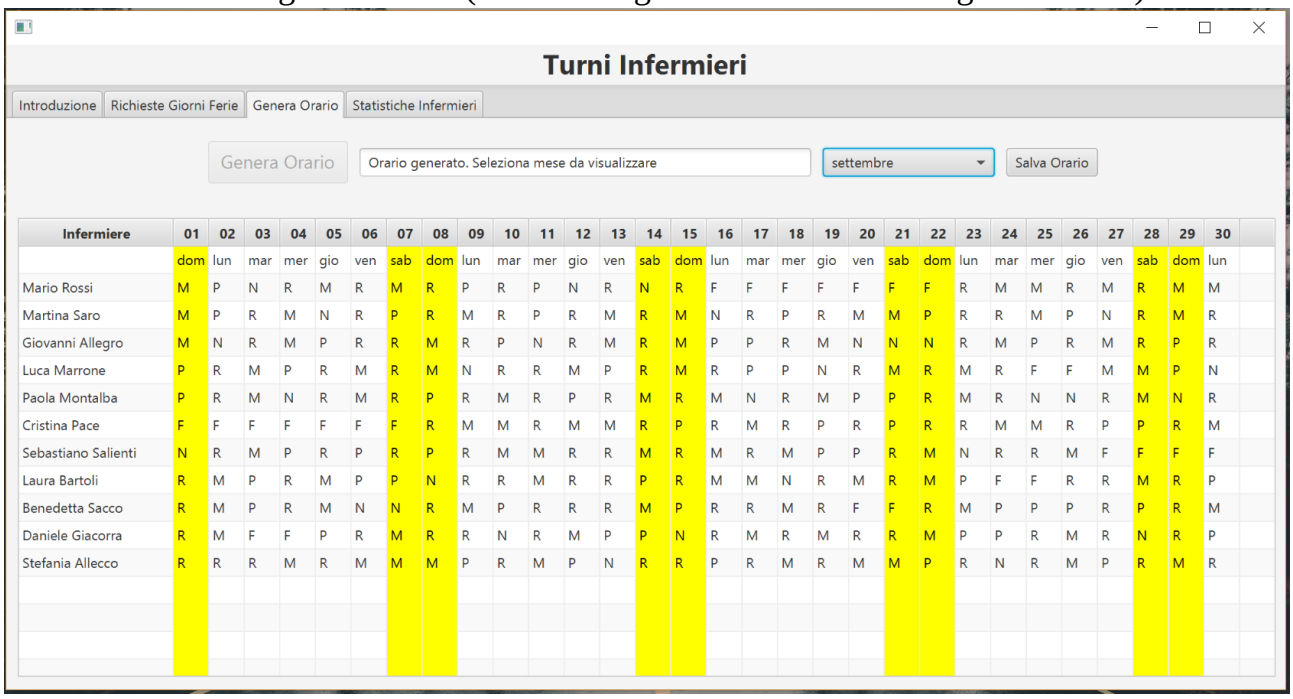
Finestra statistiche infermieri: scegliendo un infermiere dal menu a tendina si potranno vedere le statistiche ad esso correlate: il numero dei diversi turni svolti durante l'anno e il numero di riposi avuti nei giorni feriali e nei giorni festivi durante l'anno.



Link al video dimostrativo del software: https://youtu.be/oSuclnxECWM

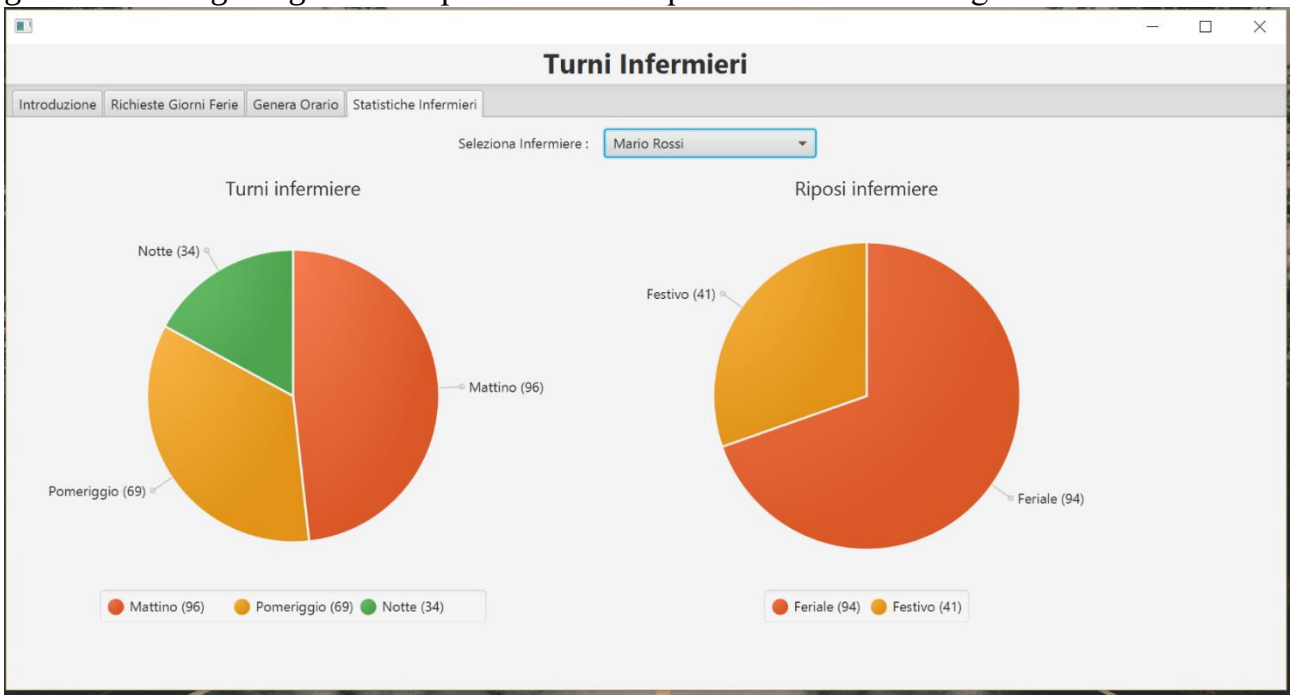
7. Tabelle con risultati sperimentali ottenuti

Finestra generazione orario dopo aver generato l'orario e selezionato un mese: M = mattino, P = pomeriggio, N = notte, R = riposo, F = ferie. Vengono evidenziate in giallo tutte le colonne dei giorni festivi (i sabati vengono considerati come giorni festivi).



Infermiere	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
	dom	lun	mar	mer	gio	ven	sab	dom	lun	mar	mer	gio	ven	sab	dom	lun	mar	mer	gio	ven	sab	dom	lun	mar	mer	gio	ven	sab	dom	lun
Mario Rossi	M	P	N	R	M	R	M	R	P	R	P	N	R	N	R	F	F	F	F	F	F	F	R	M	M	R	M	R	M	M
Martina Saro	M	P	R	M	N	R	P	R	M	R	P	R	M	R	M	N	R	P	R	M	M	P	R	R	M	P	N	R	M	R
Giovanni Allegro	M	N	R	M	P	R	R	M	R	P	N	R	M	R	M	P	P	R	M	N	N	N	R	M	P	R	M	R	P	R
Luca Marrone	P	R	M	P	R	M	R	M	N	R	R	M	P	R	M	R	P	P	N	R	M	R	M	R	F	F	M	M	P	N
Paola Montalba	P	R	M	N	R	M	R	P	R	M	R	P	R	M	R	M	N	R	M	P	P	R	M	R	N	N	R	M	N	R
Cristina Pace	F	F	F	F	F	F	F	R	M	M	R	M	M	R	P	R	M	R	P	R	P	R	R	M	M	R	P	P	R	M
Sebastiano Salienti	N	R	M	P	R	P	R	P	R	M	M	R	R	M	R	M	R	M	P	P	R	M	N	R	R	M	F	F	F	F
Laura Bartoli	R	M	P	R	M	P	P	N	R	R	M	R	R	P	R	M	M	N	R	M	R	M	P	F	F	R	R	M	R	P
Benedetta Sacco	R	M	P	R	M	N	N	R	M	P	R	R	R	M	P	R	R	M	R	F	F	R	M	P	P	P	R	P	R	M
Daniele Giacorra	R	M	F	F	P	R	M	R	R	N	R	M	P	P	N	R	M	R	M	R	R	M	P	P	R	M	R	N	R	P
Stefania Allecco	R	R	R	M	R	M	M	M	P	R	M	P	N	R	R	P	R	M	R	M	M	P	R	N	R	M	P	R	M	R

Finestra statistiche infermieri dopo aver selezionato un infermiere: attraverso i due grafici che vengono generati si può valutare l'equità di trattamento degli infermieri.



8. Valutazioni sui risultati ottenuti e conclusioni

In conclusione, l’applicazione è in grado di programmare i turni annui degli infermieri in un reparto ospedaliero, rispettando vincoli contrattuali ed equità di trattamento tra gli infermieri.

Il tempo impiegato per la generazione dell’orario può variare da qualche secondo ad alcune decine di secondi. Se si considera che lo stesso compito eseguito manualmente può richiedere diversi giorni lavorativi, si evince chiaramente il vantaggio in termini di tempo.

In secondo luogo, creare a mano un orario che distribuisca in maniera equa i turni lavorativi e i riposi, come fa l’applicazione, è davvero molto difficile.

L’applicativo non funziona solo per il reparto di riferimento: andando infatti a modificare nel database fornito l’elenco degli infermieri e il numero di infermieri richiesti per ogni turno, è possibile adattare il programma a qualsiasi reparto si voglia.

Una problematica dell’applicativo è la seguente: il trattamento degli infermieri è equo sul periodo annuale, può capitare però che, preso in considerazione un periodo più breve, alcuni infermieri facciano dei turni lavorativi non ottimali.

L’applicazione sviluppata non tiene conto di tutte le variabili e degli eventi straordinari che possono verificarsi nella realtà. Il personale infermieristico può essere composto da infermieri con contratto di tipo diverso (tempo pieno, tempo parziale, fruizione della legge 104). Non vengono tenuti in conto i permessi straordinari e la mutua.

L’applicazione potrebbe essere già utilizzata nella realtà così com’è oppure la programmazione dei turni lavorativi generata, potrebbe essere utilizzata come base di partenza per poi essere modificata in base alle preferenze, necessità o cambi turno richiesti dagli infermieri.