Università degli Studi di Salerno

Corso di Laurea Magistrale in Informatica

Curriculum Sicurezza Informatica

Farm App

Relatori
Prof. Genoveffa Tortora
Prof. Michele Risi

Candidati
Antonio Robustelli
Giuseppe Sessa

Indice

Capitolo 1

Introduzione

Questo capitolo descrive l'applicazione realizzata e le funzionalità offerte all'utente che andrà ad utilizzarla.

1.1 Farm App

L’applicazione realizzata, di nome **Farm App**, ha il compito di aiutare un utente alla ricerca di una farmacia o parafarmacia a seconda delle proprie esigenze. In particolare l’utente può scegliere tra una farmacia o una parafarmacia oppure ricercare entrambe a seconda di se un prodotto è mutuabile (quindi può essere venduto solo in farmacia) oppure non è mutuabile (quindi può trovarsi sia in farmacia che in parafarmacia).

1.2 Funzionalità offerte

L’applicazione realizzata permette all’utente di ricercare la farmacia o la parafarmacia in varie modalità.

Innanzitutto può ricercarle attraverso un form di ricerca se vuole una farmacia/parafarmacia in una determinata città. Per fare questo verrà mostrato un form di ricerca in cui l’utente va a selezionare la **regione**, la **provincia**, ed in seguito la **città** in cui vuole ricercarla e, poi, attraverso un radio button va a selezionare se ricerca **farmacie**, **parafarmacie** oppure **entrambe**.
Dopo aver completato tutti i campi e aver premuto il tasto **cerca** verrà mostrata la lista completa delle farmacie, delle parafarmacie o entrambe a seconda di cosa è stato selezionato. L’utente può scegliere tra tutte quelle selezionate e farsi mostrare la posizione sulla mappa.

Altra modalità di ricerca è attraverso la propria posizione. Se l’utente ha la necessità di ricercare una farmacia o una parafarmacia vicino a lui può cliccare il tasto **usa GPS** e mostrerà tutti i possibili risultati direttamente sulla mappa e quindi l’utente può scegliere la più utile o la più vicina a lui.

Capitolo 2

I dataset

Questo capitolo descrive i dataset selezionati su cui si basa l’intera applicazione.

2.1. Dataset utilizzati

I dataset utilizzati sono stati selezionati dal sito del Ministero della Salute e sono essenzialmente due:

- Elenco delle farmacie italiane;
- Elenco delle parafarmacie italiane.

Il dataset delle **farmacie** è formato inizialmente da:

- **Codice identificativo:** rappresenta un codice per identificare la farmacia all'interno del dataset;
- **Indirizzo:** contiene l'indirizzo della farmacia;
- **Descrizione:** contiene il nome della farmacia e un eventuale dottore associato;
- **Partita Iva:** rappresenta una sequenza di 11 cifre che identifica univocamente una attività, e nel caso specifico una farmacia, ai fini dell'imposizione fiscale;
- **CAP:** rappresenta il codice di avviamento postale;
- **Codice comune:** rappresenta un codice univoco per identificare un comune;
- **Descrizione comune:** contiene il nome del comune dove è situata la farmacia;
- **Frazione:** contiene eventualmente il nome della frazione del comune dove è situata la farmacia;
- **Codice provincia:** rappresenta un codice univoco per identificare una provincia;
- **Sigla provincia:** contiene le due lettere per indicare la provincia in cui si trova la farmacia;
- **Descrizione provincia:** contiene il nome della provincia;
- **Codice regione:** rappresenta un codice univoco per identificare una regione;
- **Descrizione regione:** contiene il nome della regione dove è situata la farmacia;
- **Data inizio validità:** rappresenta la data di inizio della validità del codice associato ad una gestione di una farmacia;
- **Data fine validità:** rappresenta la data di fine della validità del codice associato ad una gestione di una farmacia;
- **Descrizione tipologia:** indica se una farmacia è ordinaria (se situata in un centro con più di 3300 abitanti) oppure sussidiata (se situata in un centro con meno di 3300 abitanti);
- **Codice tipologia:** rappresenta un codice per indicare se la farmacia è ordinaria o sussidiata;
- **Latitudine:** rappresenta una coordinata utilizzata per individuare una specifica posizione sulla superfice terrestre;
- **Longitudine:** rappresenta l'altra coordinata utilizzata per individuare una specifica posizione sulla superfice terrestre;
- **Locazione:** indica l'emisfero in cui si trova la farmacia.

Il dataset delle **parafarmacie** è formato da:

- **Codice identificativo:** rappresenta il codice per identificare la parafarmacia all'interno del dataset;
- **Denominazione:** contiene il nome della parafarmacia e un eventuale dottore associato;
- **Indirizzo:** contiene l'indirizzo in cui è situata la parafarmacia;
- **Partita Iva:** rappresenta una sequenza di 11 cifre che identifica univocamente una attività, e nel caso specifico una parafarmacia, ai fini dell'imposizione fiscale;

- **CAP:** rappresenta il codice di avviamento postale;
- **Codice Comune:** contiene il codice univoco per rappresentare il comune;
- **Descrizione comune:** contiene il nome del comune dove è situata la parafarmacia;
- **Codice provincia:** contiene il codice univoco per rappresentare la provincia;
- **Sigla provincia:** contiene le due lettere utilizzate per indicare la provincia;
- **Descrizione provincia:** indica la provincia dove è situata la parafarmacia;
- **Codice regione:** contiene il codice univoco utilizzato per rappresentare la provincia;
- **Descrizione regione:** rappresenta il nome della regione dove è situata la parafarmacia;
- **Data inizio validità:** rappresenta la data di inizio della validità del codice associato ad una gestione di una parafarmacia;
- **Data fine validità:** rappresenta la data di fine della validità del codice associato ad una gestione di una parafarmacia;
- **Latitudine:** rappresenta una coordinata utilizzata per individuare una specifica posizione sulla superfice terrestre;
- **Longitudine:** rappresenta l'altra coordinata utilizzata per individuare una specifica posizione sulla superfice terrestre;
- **Localize:** indica l'emisfero in cui si trova la parafarmacia;

2.2 Dataset di supporto

Il dataset delle farmacie e il dataset delle parafarmacie sono stati affiancati da un dataset che contiene tutti i comuni italiani dato che nel dataset delle parafarmacie non era indicata la provincia ma solo la sigla della provincia ed inoltre non è sicuro che all'interno di un comune esista una farmacia oppure una parafarmacia.

Il dataset dei **comuni** è formato da:

- **Codice regione:** indica il codice della regione dove è situato il comune;
- **Codice città:** indica il codice univoco per identificare la città dove è situato il comune;
- **Codice provincia:** indica il codice univoco per identificare la provincia dove è situato il comune;
- **Progressivo del comune:** indica il codice per identificare univocamente il comune all'interno del dataset;
- **Codice comune:** rappresenta il codice univoco per identificare il comune;
- **Denominazione:** indica il nome del comune;
- **Ripartizione geografica:** indica la posizione geografica dove è situato il comune;
- **Denominazione regione:** indica il nome della regione dove è situato il comune;
- **Denominazione città:** indica il nome della provincia dove è situato il comune;
- **Sigla automobilistica:** indica le due lettere per identificare la provincia di appartenenza;
- **Codice catastale comune:** indica il codice univoco per indicare il comune;
- **Popolazione:** indica il numero di abitanti del comune;
- **Codice NUTS:** rappresenta la nomenclatura per le unità territoriali per le statistiche italiane.

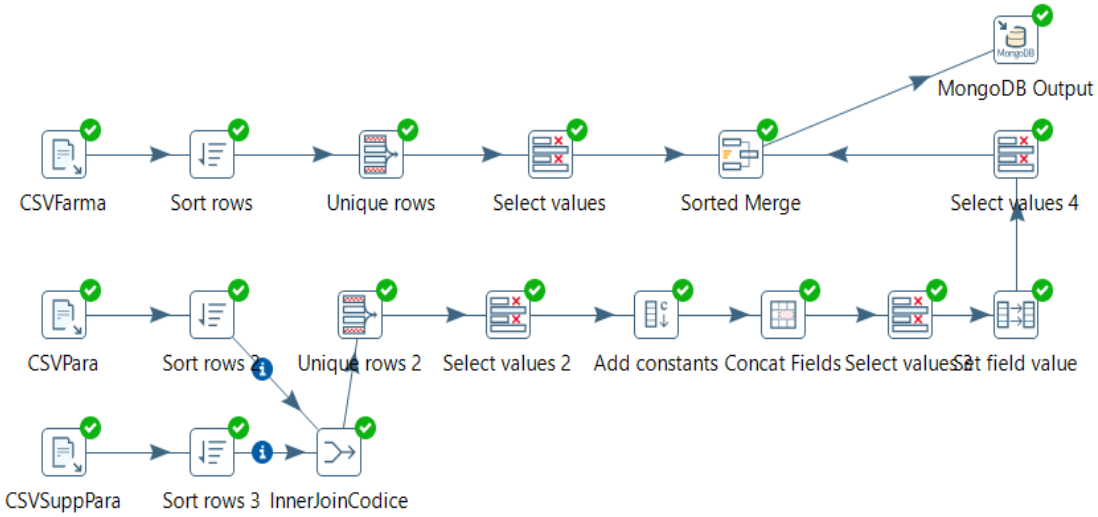
Capitolo 3

Trattamento dei dati

Per il trattamento dei dati nei dataset è stato utilizzato il software Pentaho Data Integration, attraverso il quale è stato possibile effettuare una serie di trasformazioni e il popolamento del database.

3.1 Trattamento dataset utilizzati

Di seguito è descritto il trattamento dei dati inerente ai dataset delle farmacie e delle parafarmacie.



- La prima trasformazione è quella inerente ai file CSV da utilizzare. Questo compito è svolto dalle trasformazioni **CSVFarma**, **CSVPara**, **CSVSuppPara**.

- La seconda trasformazione è quella inerente all'ordinamento dei dati contenuti nei dataset. Questo compito è svolto dalle trasformazioni **Sort Rows**, **Sort Rows 2**, **Sort Rows 3**.

- La terza trasformazione è l'inner join, sul campo codice, tra i dataset in input, rispettivamente, nelle trasformazioni **CSVPara** e **CSVSuppPara**.

- La quarta trasformazione è quella inerente all'eliminazione delle righe duplicate, in quanto i dataset iniziali contengono, in alcuni casi, informazioni ripetute per le farmacie e le parafarmacie. Questo compito è svolto rispettivamente da **Unique Rows** e da **Unique Rows 2**.

- La quinta trasformazione è quella inerente ai passi **Select Value 2**, **Add Constants**, **Concat fields**, **Select Value 3** e **Set Field Value**. In questi passi vengono resi negativi i codici del dataset delle parafarmacie.

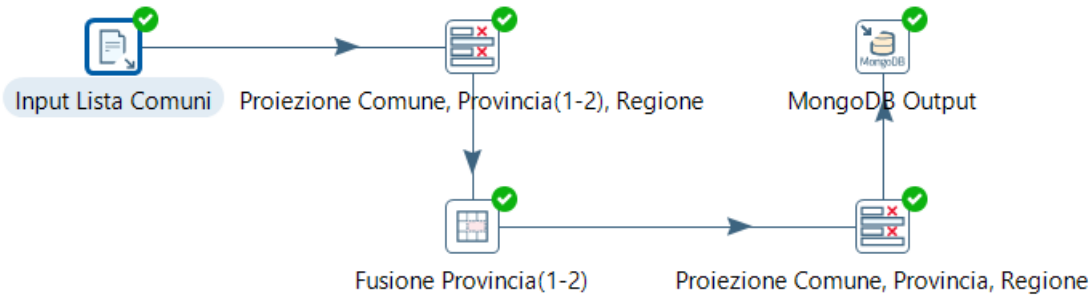
- La sesta trasformazione è quella inerente ai passi **Select Value** e **Select Value 4**, i quali si occupano di selezionare solamente i campi di interesse per il funzionamento dell'applicazione.

- La settima trasformazione è quella inerente al passo **Sorted Merge**. In questo passo si uniscono i dataset principali contenenti i campi di interesse selezionati in precedenza. Inoltre, l'unione è ordinata sul campo CODICE dei due dataset.

- L’ottava trasformazione è quella inerente al popolamento del database. Questa trasformazione è ottenuta dal passo **Mongo DB Output**.

3.2 Trattamento dataset di supporto

Di seguito è descritto il trattamento dei dati del dataset di supporto inerente ai comuni italiani con relativa provincia e regione di appartenenza.



- La prima trasformazione è inerente al file CSV utilizzato. In particolare questo è svolto dal passo **Input Lista Comuni**.

- La seconda trasformazione è inerente alla selezione dei campi di interesse per l’applicazione. In particolare vengono selezionati i campi **Comune**, il campo **Città Metropolitane**, il campo **Provincia** e infine il campo **Regione**.

- La terza trasformazione è quella inerente la fusione dei due campi Provincia dato che alcune province sono contenute nel primo campo altre nel secondo. Questo è dato dal passo **Fusione Provincia (1-2)**.

- Nella quarta trasformazione vengono eliminati i due campi per selezionare la provincia e se ne crea uno soltanto. È dato dal passo **Proiezione Comune, Provincia, Regione**.

- La quinta trasformazione è quella inerente al popolamento del database. Questa trasformazione è ottenuta dal passo **Mongo DB Output**.

Capitolo 4

Implementazione

In questo capitolo sono riportati i tools, le tecnologie e le query utilizzate per implementare l’applicazione.

4.1 Tecnologie utilizzate

Le tecnologie utilizzate per implementare l'applicazione sono: HTML5, CSS3, JavaScript, JSON, AJAX, Java e XML. Tali tecnologie sono state combinate insieme grazie all'utilizzo di Eclipse.

L'applicazione proposta è una web application che poggia su Mongo DB, uno dei più famosi database NoSQL.

4.1.1 Mongo DB

Mongo DB è un database NoSQL orientati ai documenti, ognuno dei quali è memorizzato nel formato JSON. Il documento è, fondamentalmente, un albero che può contenere molti dati, anche annidati.

Ogni documento è identificato da una chiave primaria (ID) e il loro scopo è quello di rappresentare i dati. Ogni documento può essere visto come l'equivalente del record delle tabelle in un database relazionale e possono essere messi in relazione tra loro. Inoltre, i documenti sono raggruppati in collezioni che possono essere anche eterogenee. Tra le varie collezioni non ci sono relazioni o legami garantiti da Mongo DB. In altri termini, non esiste un concetto analogo al vincolo di integrità referenziale presente nei database relazionali.

Le caratteristiche di Mongo DB sono:

1. Consente servizi di alta disponibilità, dal momento che la replicazione di un database (replica set) può avvenire in modo molto semplice;

2. Garantisce la scalabilità automatica, ossia la possibilità di distribuire (Sharding) le collezioni in cluster di nodi, in modo da supportare grandi quantità di dati senza influire pesantemente sulle performance.

3. Mongo DB si adatta a molti contesti e in generale quando si manipolano grandi quantità di dati eterogenei e senza uno schema.

4.2 Query

Le query su cui si basa l'applicazione sono classificabili in due categorie: una per l'aggiornamento e/o creazione dei campi nel database ed un'altra per poter effettuare le varie ricerche.

Le query sono state scritte in Java grazie alla libreria **mongo-java-driver-3.4.3**, fornita stesso da Mongo DB, e facendo un forte uso dei Filters e degli Indexes (indici).

Di seguito sono riportare le query per l'aggiornamento e/o creazione dei campi nel database.

```
Connection c = new Connection("test","farma");

MongoCollection<Document> coll = c.getColl();

coll.updateMany(or(eq("DESCRIZIONEREGIONE","PROV. AUTON. BOLZANO"),eq("DESCRIZIONEREGIONE","PROV. AUTON. TRENTO")
        ,eq("DESCRIZIONEREGIONE","Prov. auton. bolzano"),eq("DESCRIZIONEREGIONE","Prov. auton. trento")),
        combine(set("DESCRIZIONEREGIONE","Trentino-Alto Adige")));

coll.updateMany(eq("DESCRIZIONEREGIONE",150),combine(
        set("DESCRIZIONECOMUNE","Bellizzi"),set("DESCRIZIONEREGIONE","Campania"),set("DESCRIZIONEPROVINCIA","Salerno")));

coll.updateMany(eq("DESCRIZIONEREGIONE",190),combine(
        set("DESCRIZIONECOMUNE","Adrano"),set("DESCRIZIONEREGIONE","Sicilia"),set("DESCRIZIONEPROVINCIA","Catania")));

coll.updateMany(or(eq("DESCRIZIONEREGIONE","Valle d`aosta"),eq("DESCRIZIONEREGIONE","VALLE D`AOSTA")),
        combine(set("DESCRIZIONEREGIONE","Valle d'Aosta")));

coll.updateMany(or(eq("DESCRIZIONEREGIONE","FRIULI VENEZIA GIULIA"),eq("DESCRIZIONEREGIONE","Friuli venezia giulia")),
        combine(set("DESCRIZIONEREGIONE","Friuli-Venezia Giulia")));

coll.updateMany(or(eq("DESCRIZIONEREGIONE","EMILIA ROMAGNA"),eq("DESCRIZIONEREGIONE","Emilia romagna")),
        combine(set("DESCRIZIONEREGIONE","Emilia-Romagna")));

coll.updateMany(eq("LATITUDINE",null),
        combine(set("LATITUDINE",0.0)));

coll.updateMany(eq("LATITUDINE","31/10/2014"),
        combine(set("LATITUDINE",0.0)));



coll.updateMany(and(eq("REGIONE","Emilia-Romagna"),eq("PROVINCIA","-Reggio nell'Emilia")),
        combine(set("PROVINCIA","-Reggio Emilia")));

coll.updateMany(and(eq("REGIONE","Emilia-Romagna"),eq("PROVINCIA","-Forlì-Cesena")),
        combine(set("PROVINCIA","-Forlì Cesena")));

coll.updateMany(and(eq("REGIONE","Calabria"),eq("PROVINCIA","-Reggio di Calabria")),
        combine(set("PROVINCIA","-Reggio Calabria")));

coll.updateMany(and(eq("REGIONE","Lombardia"),eq("PROVINCIA","-Monza e della Brianza")),
        combine(set("PROVINCIA","-Monza")));

coll.updateMany(and(eq("REGIONE","Toscana"),eq("PROVINCIA","-Massa-Carrara")),
        combine(set("PROVINCIA","-Massa Carrara")));

coll.updateMany(and(eq("REGIONE","Puglia"),eq("PROVINCIA","-Barletta-Andria-Trani")),
        combine(set("PROVINCIA","-Barletta Andria Trani")));

coll.updateMany(and(eq("REGIONE","Sardegna"),eq("PROVINCIA","-Olbia-Tempio")),
        combine(set("PROVINCIA","-Olbia Tempio")));

coll.updateMany(and(eq("REGIONE","Sardegna"),eq("PROVINCIA","-Carbonia-Iglesias")),
        combine(set("PROVINCIA","-Carbonia Iglesias")));

coll.updateMany(eq("REGIONE","Valle d'Aosta/Vallée d'Aoste"),
        combine(set("REGIONE","Valle d'Aosta")));

coll.updateMany(eq("REGIONE","Valle d'Aosta"),
        combine(set("PROVINCIA","-Aosta")));

coll.updateMany(eq("REGIONE","Trentino-Alto Adige/Südtirol"),
        combine(set("REGIONE","Trentino-Alto Adige")));

coll.updateMany(eq("PROVINCIA","-;Bolzano/Bozen"),
        combine(set("PROVINCIA","-;Bolzano")));
```

```
Connection c = new Connection("test","farma");
MongoCollection<Document> coll = c.getColl();

ArrayList<Document> cc = coll.find().into(new ArrayList<Document>());

HashMap<Long,ArrayList<Double>> items = new HashMap<Long,ArrayList<Double>>();

for(int i=0;i<cc.size();i++)
{
    ArrayList<Double> d = new ArrayList<Double>();
    d.add(cc.get(i).getDouble("LONGITUDINE"));
    d.add(cc.get(i).getDouble("LATITUDINE"));

    Long app = cc.get(i).getLong("CODICE");
    items.put(app, d);
}

for(Map.Entry m:items.entrySet())
{
    coll.updateOne(eq("CODICE",m.getKey()),set("LOCATION", m.getValue()));
}
```

Di seguito sono riportate le query per effettuare la ricerca nel database.

```
Connection c = new Connection("test","comuni");
MongoCollection<Document> coll = c.getColl();

MongoCursor<String> app = coll.distinct("REGIONE", String.class).iterator();
```

```
Connection c = new Connection("test","comuni");
MongoCollection<Document> coll = c.getColl();

String regione = (String) request.getParameter("regione");

MongoCursor<String> app = coll.distinct("PROVINCIA", eq("REGIONE",regione), String.class).iterator();
```

```
Connection c = new Connection("test","comuni");
MongoCollection<Document> coll = c.getColl();

String regione = (String) request.getParameter("regione");
String provincia = (String) request.getParameter("provincia");

MongoCursor<String> app = coll.distinct("COMUNE", and(eq("REGIONE",regione),
      or(eq("PROVINCIA","-" + provincia),eq("PROVINCIA",provincia + "-"))), String.class).iterator();
```

```
Connection c = new Connection("test","farma");
MongoCollection<Document> coll = c.getColl();

String comune = (String) request.getParameter("comune");
String radio = (String) request.getParameter("linguaggio");

ArrayList<Document> employees = null;

if(radio.equals("-1"))
{
    employees = (ArrayList<Document>)
            coll.find().filter(and(or(eq("DESCRIZIONECOMUNE",comune),eq("DESCRIZIONECOMUNE",comune.toUpperCase()))
            ,lt("CODICE",0))).into(new ArrayList<Document>());
}
else if(radio.equals("1"))
{
    employees = (ArrayList<Document>)
            coll.find().filter(and(or(eq("DESCRIZIONECOMUNE",comune),eq("DESCRIZIONECOMUNE",comune.toUpperCase()))
            ,gte("CODICE",0))).into(new ArrayList<Document>());
}
else
{
    employees = (ArrayList<Document>)
            coll.find().filter(or
        (eq("DESCRIZIONECOMUNE",comune),eq("DESCRIZIONECOMUNE",comune.toUpperCase()))).into(new ArrayList<Document>());
}
```

```
Connection c = new Connection("test","farma");
MongoCollection<Document> coll = c.getColl();

coll.createIndex(Indexes.geo2dsphere("LOCATION"));

String longitudine = request.getParameter("lon");
String latitudine = request.getParameter("lat");

Point refPoint = new Point(new Position(Double.parseDouble(longitudine),Double.parseDouble(latitudine)));
ArrayList<Document> app = coll.find(Filters.near("LOCATION", refPoint, 4000.0, 0.0)).into(new ArrayList<Document>());
```

Capitolo 5

Sviluppi futuri

In questo capitolo sono elencati alcuni sviluppi futuri.

5.1 Sviluppi futuri

- Utilizzare dei dataset migliori e, all’occorrenza, più aggiornati in maniera tale da migliorare la qualità delle informazioni in possesso.

- Migliorare ulteriormente l’interfaccia grafica.

- Rilasciare una versione mobile per i diversi device sul mercato.

- Aggiungere la funzionalità del navigatore per poter guidare l’utente dalla sua posizione fino alla destinazione desiderata.