

# POLITECNICO DI TORINO

## DIPARTIMENTO DI INGEGNERIA GESTIONALE E DELLA PRODUZIONE

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



### **Applicazione di analisi di percorsi aerei tra città americane**

RELATORE  
Prof. Fulvio Corno

CANDIDATO  
Ghassane Ben el aattar  
S236843

# INDICE E COLLEGAMENTI

<a href="#"><u>I. Proposta di progetto.....</u></a>	3
<a href="#"><u>II. Descrizione del problema affrontato .....</u></a>	7
<a href="#"><u>III. Descrizione del data-set utilizzato per l'analisi.....</u></a>	8
<a href="#"><u>IV. Descrizione ad alto livello delle strutture dati e degli algoritmi utilizzati.....</u></a>	9
<a href="#"><u>V. Diagramma delle classi.....</u></a>	12
<a href="#"><u>VI. Video dell'applicazione realizzata e link al video dimostrativo del software ...</u></a>	15
<a href="#"><u>VII. Tabelle con risultati sperimentali ottenuti.....</u></a>	19
<a href="#"><u>VIII. Valutazioni sui risultati ottenuti e conclusioni.....</u></a>	20

## ***I- PROPOSTA DI PROGETTO***

### **DESCRIZIONE DEL PROBLEMA PROPOSTO:**

L'applicazione ha l'obiettivo primario di determinare l'insieme delle tratte che, in un dato anno selezionabile dall'utente, massimizzino il numero di passeggeri totali, con un vincolo di distanza massima dello stesso insieme (sempre da input dell'utente), prendendo come riferimento il database dei voli americani al 1990 al 2009. Il problema è risolto mediante algoritmo ricorsivo, che dovrà valutare nell'insieme totale delle tratte in un dato anno, il sottoinsieme ottimale.

Oltre a ciò, astrando le varie città come vertici di un grafo orientato, è possibile verificare la presenza di percorsi indiretti tra due località e il relativo cammino di minima distanza mediante efficienti algoritmi su grafi.

### **DESCRIZIONE DELLA RILEVANZA GESTIONALE DEL PROBLEMA:**

Un'azienda che fornisce servizi di voli mediante aeroplani ha tutto l'interesse a verificare l'efficienza delle proprie tratte, ovvero su quali di esse ha logicamente, mediante parametri quantitativi, investire. Inoltre, è fondamentale la verifica immediata della presenza o meno di percorsi diretti o indiretti tra due città, per comprendere potenziali margini di futuri investimenti.

## DESCRIZIONE DEI DATA-SET PER LA VALUTAZIONE:

Il data-set utilizzato è il database Us\_airports, che contiene i voli effettuati all'interno del suolo americano nel periodo temporale dal 1990 al 2009. Esso è stato reperito su Kaggle.com al seguente link: <https://www.kaggle.com/flashgordon/usa-airport-dataset>.

I dati fondamentali presenti nel data-set che sono stati utilizzati sono:

- Origin\_city: città di origine del volo.
- Destination\_city: città di destinazione del volo.
- Passengers: passeggeri del volo.
- Distance: distanza coperta dal volo.
- Fly\_date: data del volo.

Per evitare ridondanze nei calcoli, si considera la media dei passeggeri per una data tratta nell'anno selezionato.

## DESCRIZIONE PRELIMINARE DEGLI ALGORITMI COINVOLTI:

-Algoritmo ricorsivo di ricerca esaustiva dello spazio delle soluzioni:

L'algoritmo che dovrà cercare le "migliori" tratte sarà un algoritmo ricorsivo, che dovrà esplorare l'intero spazio delle possibili soluzioni con i vincoli indicati.

Esso è paragonabile all'algoritmo ricorsivo dello zaino (knapsack 0-1), in cui le varie chiamate ricorsive danno origine ad un albero

binario. In questo caso, la decisione da prendere è se includere o meno una tratta nella risposta, effettuando quindi due diverse chiamate ricorsive. La complicità maggiore è sicuramente la ricostruzione delle tratte selezionate, che richiede di utilizzare array aggiuntivi durante la ricorsione. La complessità prevista è esponenziale.

-Algoritmi su grafi, visita in profondità:

Per verificare la connettività di due città in un dato anno, si possono immaginare le città come vertici di un grafo pesato ed orientato ed applicare un algoritmo ricorsivo di visita in profondità. Siccome le città sono rappresentate come stringhe (sequenze di caratteri), esiste una criticità nel dover costruire una lista di adiacenza che permetta accesso ai vertici adiacenti con accesso costante (complessità  $O(1)$  quindi).

La soluzione è usare una HashMap, una mappa che associerà le stringhe a liste di archi vicini. Una HashMap associa due oggetti mediante hashing, che permette un inserimento ed accesso con complessità costante. In questo modo, la complessità dell'algoritmo sarà  $O(|V| + |E|)$ , dove  $|V|$  è il numero di vertici ed  $|E|$  il numero di archi.

-Algoritmi su grafi, cammino di minima distanza tra due vertici (Dijkstra).

Per calcolare la distanza minima in un dato anno tra due vertici, bisogna prima verificarne la connettività.

Dopo ciò, si può applicare l'algoritmo di Dijkstra, che permette di trovare il cammino di minimo peso tra due vertici in un grafo.

Soprattutto in questo caso, è piuttosto intricato dal punto di vista dell'implementazione costruire l'algoritmo con stringhe come vertici del grafo. E' anche qui necessario l'utilizzo di HashMap per aggiornare il percorso e la distanza migliore fino ad un certo punto dell'algoritmo.

L'implementazione con la migliore complessità teorica utilizza una min-heap (coda prioritaria di default in java) per tener traccia degli archi da processare prima degli altri ed ha una complessità finale di  $O(|E| \log(|V|))$ .

#### DESCRIZIONE PRELIMINARE DELLE FUNZIONALITA' PREVISTE PER L'APPLICAZIONE SOFTWARE:

L'applicazione sarà dotata di un'interfaccia grafica, nella quale l'utente dovrà selezionare un anno di riferimento dall'apposito box in alto. Una volta effettuato ciò, sarà possibile inserire una distanza massima in una casella di testo apposita e utilizzare il tasto apposito per trovare l'insieme delle tratte "migliori".

Il tasto "carica voli" permetterà di inserire le città nei box "origine" e "destinazione", che permetteranno a loro volta di verificare la connettività e il cammino minimo in termini di distanza totale.

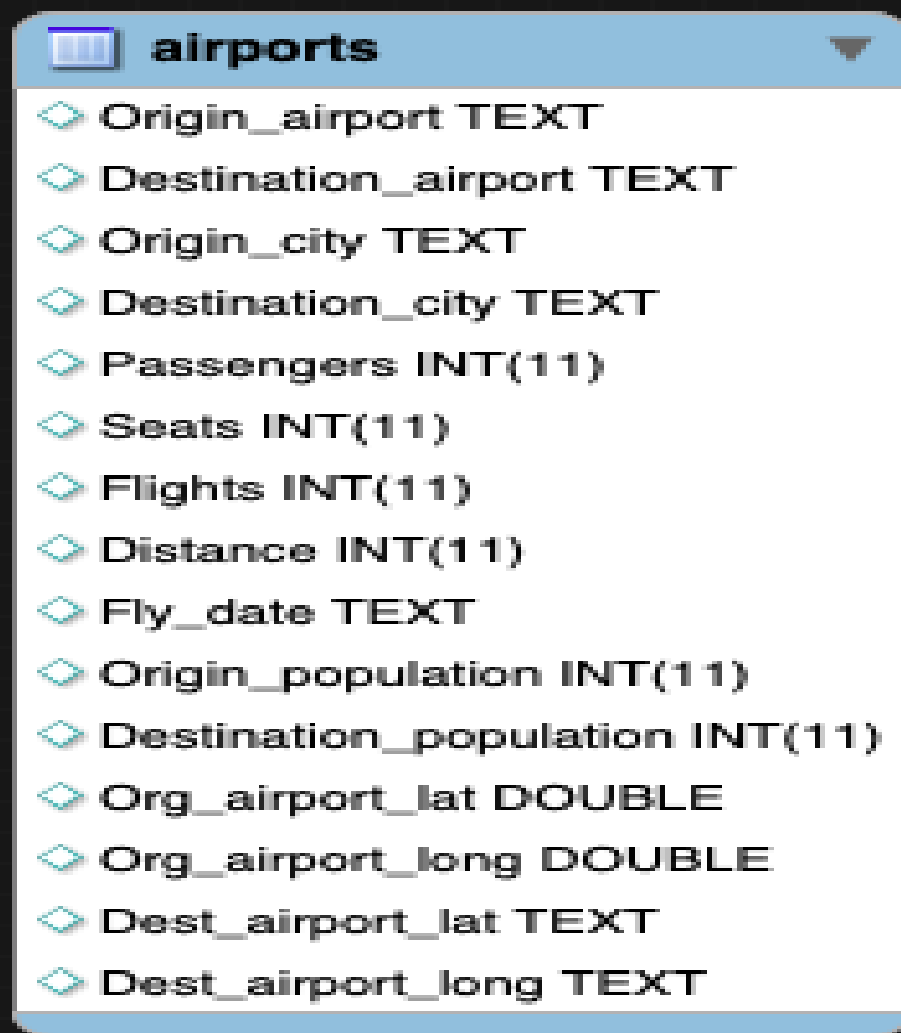
I risultati ottenuti saranno riportati in un box nella parte inferiore dell'applicazione grafica.

## ***II- DESCRIZIONE DETTAGLIATA DEL PROBLEMA AFFRONTATO***

Un'azienda che fornisce servizi di trasporto mediante aerei ha sicuramente interesse nell'avere riscontro quantitativo dell'efficienza delle tratte che copre in un certo periodo temporale. Chiaramente cercare quali di esse sono maggiormente efficienti, cioè trasportano in media un elevato numero di passeggeri rispetto ad altre. L'indagine può però fornire dati fuorvianti se non vengono imposti i corretti vincoli quantitativi, ad esempio si potrebbero preferire tratte che trasportano un elevato numero di passeggeri ma a distanze molto più elevate. Quindi, il vincolo della distanza totale del sottoinsieme da ricercare, come menzionato in precedenza, può essere sicuramente un utile parametro, seppur chiaramente artificiale.

Un'altra esigenza plausibile è la verifica di percorsi tra due città, ma ciò deve essere effettuato chiaramente in maniera efficiente. In un vasto data-set, certi algoritmi inefficienti potrebbero impiegare molto tempo a verificare semplici informazioni. Utilizzando appositi algoritmi, un'azienda di voli può verificare se due città richiedono una connessione, oppure se si può operare in maniera attiva sul collegamento tra due città, possibilmente quando il numero di scali intermedi è piuttosto sconveniente.

### ***III- DESCRIZIONE DEL DATA-SET UTILIZZATO PER L'ANALISI***



airports	
◇	Origin_airport TEXT
◇	Destination_airport TEXT
◇	Origin_city TEXT
◇	Destination_city TEXT
◇	Passengers INT(11)
◇	Seats INT(11)
◇	Flights INT(11)
◇	Distance INT(11)
◇	Fly_date TEXT
◇	Origin_population INT(11)
◇	Destination_population INT(11)
◇	Org_airport_lat DOUBLE
◇	Org_airport_long DOUBLE
◇	Dest_airport_lat TEXT
◇	Dest_airport_long TEXT

Come accennato in fase di proposta, le informazioni che verranno attivamente utilizzate sono origine,destinazione,passeggeri,distanza,data.

Dalla data andrà estrapolato l'anno,il quale è l'intervallo temporale di riferimento per il software.



#### ***IV- DESCRIZIONE DI ALTO LIVELLO DELLE STRUTTURE DATI E DEGLI ALGORITMI UTILIZZATI***

Come accennato il fase di proposta,il problema principale del software viene risolto mediante algoritmo ricorsivo di ricerca esaustiva dello spazio delle soluzioni. Esso è rassimilabile a livello teorico al problema dello zaino,in cui la decisione da prendere per ogni elemento è se includerlo nell'insieme che costituirà la soluzione ottima o meno. Si viene a costituire un albero binario decisionale che rappresenta lo spazio complessivo di dimensione esponenziale delle soluzioni.

```
int ans1 = input.get(n - 1).getAvg_passengers()
           + recursiveSearch(distance - input.get(n - 1).getDistance(), input, n - 1, v1);
int ans2 = recursiveSearch(distance, input, n - 1, v2);
```

Come si può notare,con la variabile “ans1” si prova ad inserire una tratta nella soluzione,con “ans” ciò non viene fatto.

Siccome è fondamentale risalire all'insieme delle tratte selezionate,è necessario memorizzare quali di esse sono selezionate in ogni chiamata ricorsiva,che viene effettuato con l'array “Visited” nell'algoritmo:

```
int v1[] = new int[visited.length];
System.arraycopy(visited, 0, v1, 0, v1.length);
int v2[] = new int[visited.length];
System.arraycopy(visited, 0, v2, 0, v2.length);
v1[n - 1] = 1;
```

Un altro problema algoritmico è la verifica della connettività tra due vertici del grafo pesato ed orientato. Ciò viene effettuato tramite visita in profondità, che è un algoritmo ricorsivo di ricerca su un grafo con complessità di caso peggiore  $O(|V| + |E|)$ . Come accennato, viene utilizzata una HashMap come lista di adiacenza e un'altra anche per verificare i nodi visitati nelle varie chiamate ricorsive.

```
private void depthFirstSearch(String vertex, Map<String, Boolean> vis) {  
    vis.put(vertex, true);  
    if (this.graph.getNeighbours(vertex).isEmpty()) {  
        return;  
    }  
    for (Edge e : this.graph.getNeighbours(vertex)) {  
        if (!vis.containsKey(e.getDestination())) {  
            depthFirstSearch(e.getDestination(), vis);  
        }  
    }  
}
```

Infine, viene implementato l'algoritmo di Dijkstra per trovare il cammino di minima distanza tra due nodi.

Questo noto algoritmo ha diverse implementazioni, di cui la migliore (perlomeno con strutture dati presenti nella libreria Java) ha una complessità di  $O(|E| \log(|V|))$  utilizzando una min-heap (coda prioritaria in Java).

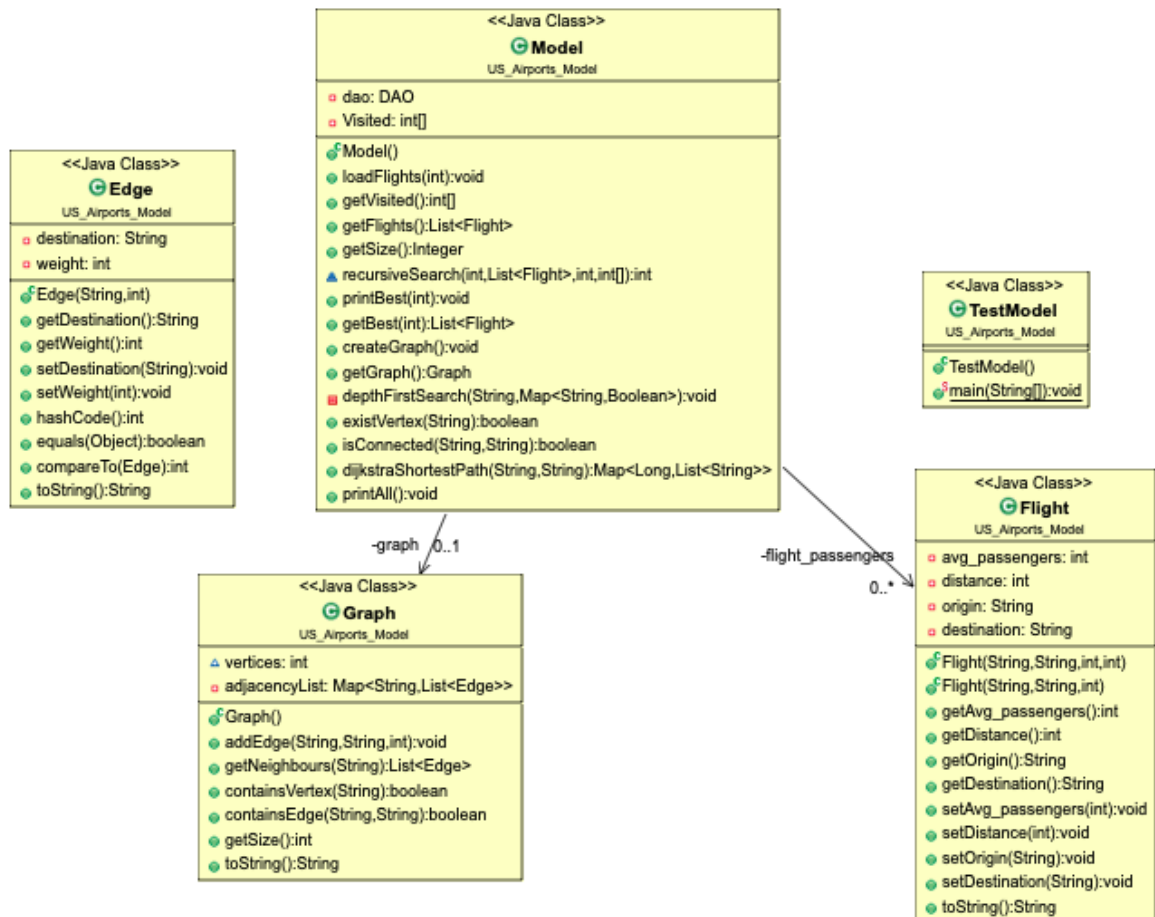
Per utilizzare la coda prioritaria con gli archi è necessario implementare un comparator nella loro classe.

Per il resto, la criticità maggiore, come accennato, è rappresentata dal parsing dell'input, che è composto da stringhe.

La soluzione è l'uso di HashMap, che permettono comunque di ottenere complessità  $O(1)$  nell'accesso e modifica di dati, mediante hashing.

```
PriorityQueue<Edge> pq = new PriorityQueue<Edge>();
long INF = 1000000000;
Map<String, Long> distanza = new HashMap<String, Long>();
Map<String, String> precedente = new HashMap<String, String>();
for (Flight f : this.flight_passengers) {
    distanza.put(f.getOrigin(), INF);
    distanza.put(f.getDestination(), INF);
}
Edge inizio = new Edge(origin, 0);
pq.add(inizio);
distanza.put(origin, (long) 0);
while (!pq.isEmpty()) {
    Edge curr = pq.peek();
    String nodo = pq.peek().getDestination();
    long peso = pq.peek().getWeight();
    pq.poll();
    if (peso != distanza.get(nodo)) {
        continue;
    }
    for (Edge e : this.graph.getNeighbours(nodo)) {
        if (distanza.get(e.getDestination()) > peso + e.getWeight()) {
            distanza.put(e.getDestination(), peso + e.getWeight());
            pq.add(new Edge(e.getDestination(), (int) peso + e.getWeight()));
            precedente.put(e.getDestination(), nodo);
        }
    }
}
}
```

## V- DIAGRAMMA DELLE CLASSI



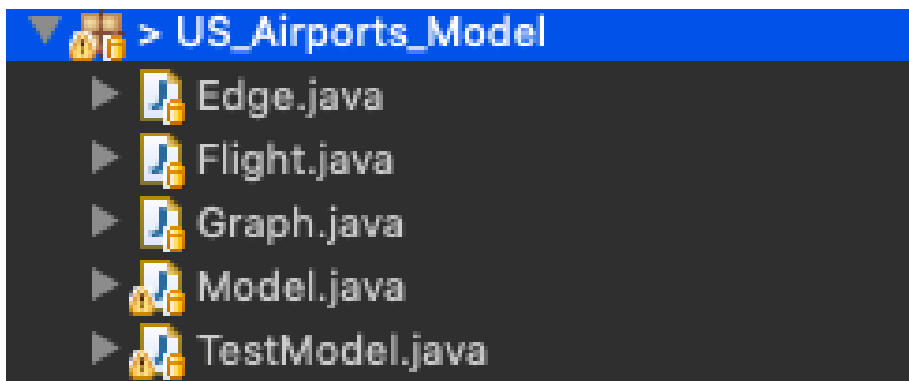
Come si può notare nel diagramma, vengono utilizzate 5 classi (4 importanti ed una di testing).

Le classi **Edge** e **Graph** consentono di utilizzare il grafo, con relativi archi e vertici.

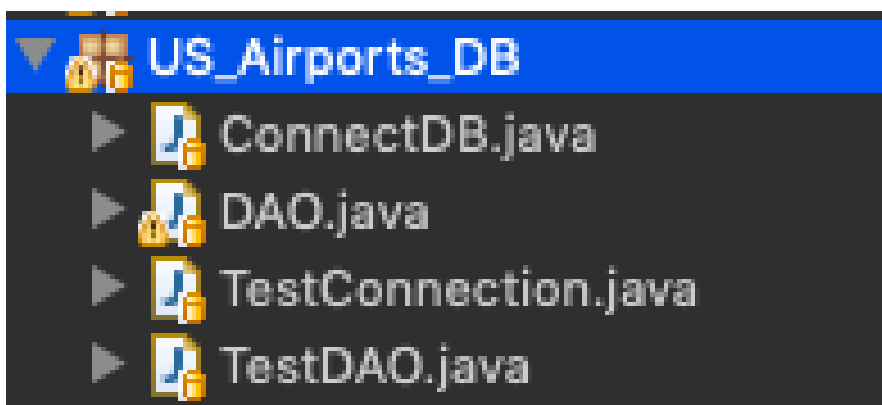
La classe **Flight** contiene le informazioni sui voli e relativi metodi.

Il Model contiene la logica principale ed algoritmi del software ed è il suo core.

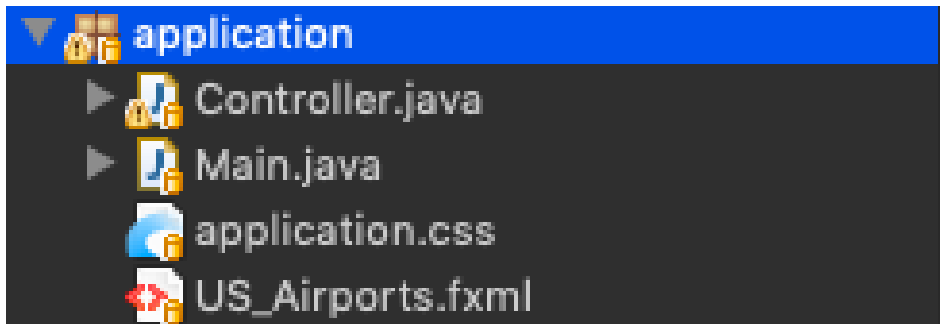
Queste classi sono raggruppate nel seguente package:



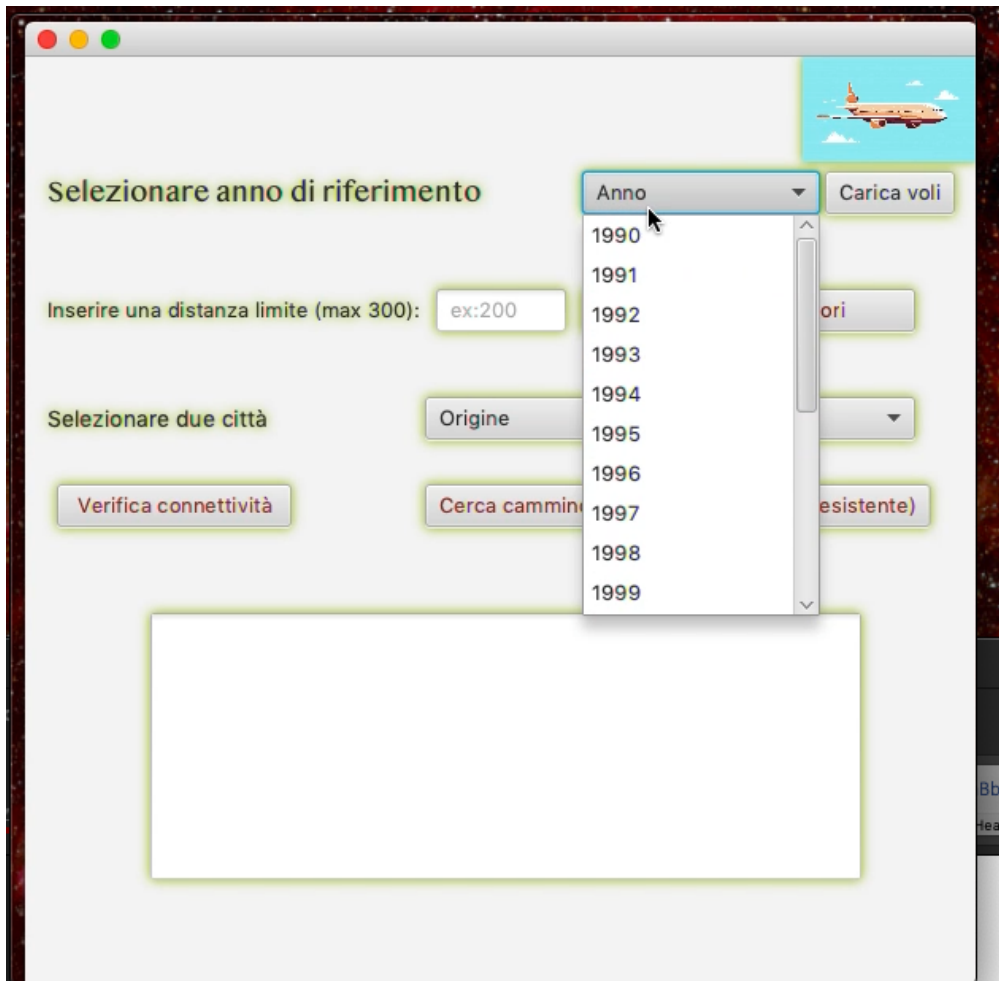
Il package che consente la connessione al data-set e l'estrazione delle informazioni da esso contiene la classe DAO e ConnectDB, oltre a classi ausiliarie di testing:



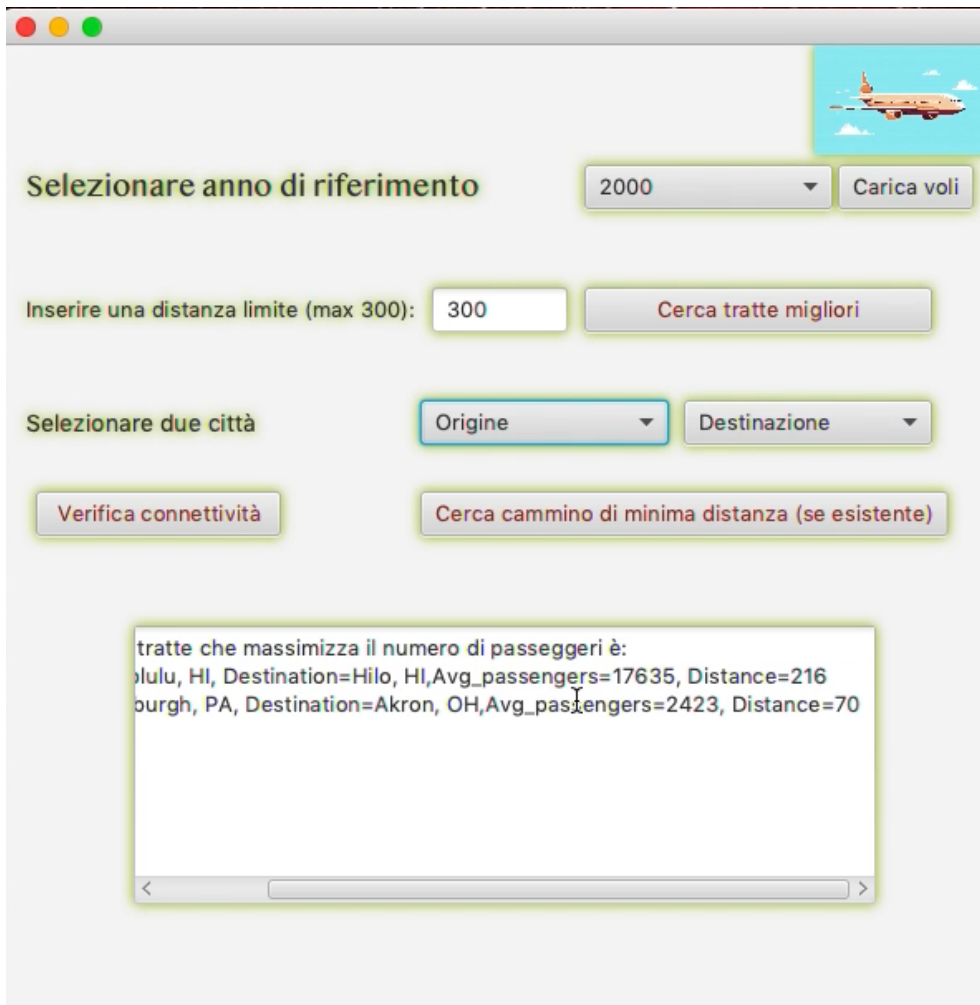
Infine, il package `application` contiene le classi `main` e `controller` (che implementa le interazioni con l'applicazione) :



## VI- VIDEATE DIMOSTRATIVE DEL SOFTWARE



Dall'interfaccia dell'applicazione, si possono selezionare i periodi temporali desiderati, step iniziale dell'utilizzo.



The screenshot shows a flight search application window. At the top right, there is a small icon of an airplane. The interface includes several input fields and buttons:

- Selezione anno di riferimento:** A dropdown menu set to "2000" and a button labeled "Carica voli".
- Inserire una distanza limite (max 300):** A text input field containing "300" and a button labeled "Cerca tratte migliori".
- Selezione due città:** Two dropdown menus labeled "Origine" and "Destinazione".
- Verifica connettività:** A button.
- Cerca cammino di minima distanza (se esistente):** A button.

Below these controls is a text area displaying the following text:

```
tratte che massimizza il numero di passeggeri è:  
olulu, HI, Destination=Hilo, HI,Avg_passengers=17635, Distance=216  
burgh, PA, Destination=Akron, OH,Avg_passengers=2423, Distance=70
```

Una volta effettuato ciò, inserendo un valore di distanza limite e premendo il tasto “Cerca tratte migliori”, viene fornito l’insieme delle tratte con le caratteristiche discusse precedentemente.



Selezionare anno di riferimento

2000 Carica voli

Inserire una distanza limite (max 300): 305 Cerca tratte migliori

Selezionare due città

Seattle, WA Reno, NV

Verifica connettività Cerca cammino di minima distanza (se esistente)

Le due città selezionate sono connesse nell'anno di riferimento

Selezionare anno di riferimento

2000 Carica voli

Inserire una distanza limite (max 300): 305 Cerca tratte migliori

Selezionare due città

Seattle, WA Reno, NV

Verifica connettività Cerca cammino di minima distanza (se esistente)

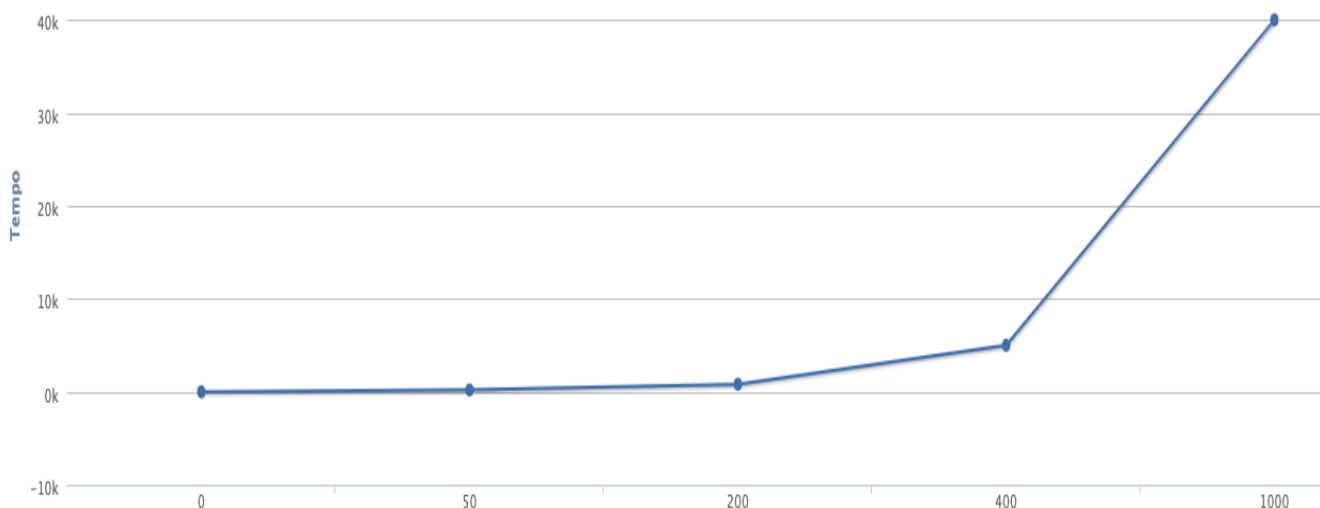
Esiste un cammino minimo di lunghezza 564  
Seattle, WA  
Reno, NV

Premendo il tasto “Carica voli”, è possibile selezionare dagli appositi box una città di origine e destinazione.

Una volta effettuata la selezione, è possibile verificare la connettività di due città ed eventuale cammino minimo con gli appositi tasti.

## ***VII- RISULTATI SPERIMENTALI OTTENUTI***

A livello puramente pratico, ha sicuramente utilità l'analisi sperimentale di un algoritmo con complessità teorica esponenziale. Infatti, nel seguente grafico, notiamo l'evolversi del tempo di esecuzione in proporzione alla distanza massima utilizzata (che chiaramente influenza lo spazio di tratte selezionate).



Come si nota dal grafico, all'aumentare della distanza, la curva che rappresenta approssimativamente il tempo di esecuzione in ms ha una crescita esponenziale, dando riscontro pratico alla complessità teorica dell'algoritmo.

## ***VIII- VALUTAZIONI SUI RISULTATI OTTENUTI***

Cercando di evidenziare un possibile criticità dell'applicazione, sicuramente la complessità esponenziale dell'algoritmo di ricerca potrebbe un limite. Infatti, sul data-set nella sua interezza esso non è eseguibile. Inoltre, il data-set dei voli potrebbe risentire della mancanza di voli negli ultimi 11 anni, periodo dove verosimilmente il numero di voli è più alto. Detto ciò, bisogna evidenziare come la struttura del software, nella sua divisione in classi con diverse funzioni, lo renda facilmente migliorabile ed adattabile a diversi contesti, problemi e data-set. In particolar modo, il data-set fornisce comunque dati che possono sicuramente essere sfruttati in diversi modi, soprattutto se si considera la possibilità di astrarre i voli come grafi e relativi vertici/archi orientati. Inoltre, la struttura dell'interfaccia grafica è molto intuitiva e user-friendly, utilizzando le possibilità offerte da JavaFX.

Per rendere ancora più funzionale a decisioni aziendali il software, sarebbe interessante poter sfruttare un data-set ancora più completo dal punto di vista dei consumatori, che possa sfruttare indici di gradimento di un volo, ad esempio come avviene nei vari siti di booking moderni. Ciò sarebbe utile per avere una metrica quantitativa e qualitativa ancora più funzionale a decisioni che un'azienda di voli deve effettuare.

Quest'opera è distribuita con licenza Creative Commons Attribuzione  
– Non commerciale - Condividi allo stesso modo 4.0 Internazionale.

Copia della licenza consultabile al sito web:

<http://creativecommons.org/licenses/by-nc-sa/4.0/>

