

***TakeAFly***

Progetto corso Basi di Dati

A.A 2019/2020

Nome gruppo: TakeAFly SpA

Gatto Michele 875446

Zuliani Riccardo 875532

Sommario

[**Introduzione** 3](#_Toc49779368)

[**1. Funzionalità di TakeAFly** 4](#_Toc49779369)

[1.1 Ricerca di un volo ed acquisto 4](#_Toc49779370)

[1.2 Login e Registrazione 6](#_Toc49779371)

[1.3 Gestione delle Prenotazioni 7](#_Toc49779372)

[1.4 Gestione dell’Account 8](#_Toc49779373)

[1.5 Dashboard 9](#_Toc49779374)

[1.6 Statistiche 11](#_Toc49779375)

[**2. La Base di Dati** 13](#_Toc49779376)

[**3. Query Interessanti** 16](#_Toc49779377)

[**4. Scelte progettuali** 17](#_Toc49779378)

[**5. Considerazioni Finali** 19](#_Toc49779379)

[**6. Come far partire TakeAFly** 19](#_Toc49779380)

# Introduzione

TakeAFly è un’applicazione web, sviluppata principalmente in Python utilizzando il framework Flask ed SQLAlchemy per interfacciarsi con il DBMS, la scelta di quest’ultimo è ricaduta su MySQL per la semplicità di installazione e di utilizzo oltre alla grande quantità di funzionalità supportate.

La parte front-end dell’applicazione è sviluppata in HTML con l’ausilio di linguaggi quali Javascript e Jquery necessari per il passaggio di dati ed altre funzionalità che verranno descritte successivamente. Per la parte estetica è stato utilizzato il framework Bootstrap che ha permesso di ottenere un risultato minimale senza troppi sforzi.

Per poter scrivere dei template con dei placeholder che permettono di scrivere codice simile a Python dove poi vengono passati dei dati per eseguire il rendering finale della pagina, è stato utilizzato il motore Jinja.

In questo documento verranno descritte le varie funzionalità dell’applicazione sia per quanto riguarda l’amministratore che per i clienti, facendo riferimento allo schema relazionale della base di dati, alle query utilizzate e alle varie scelte progettuali che sono state fatte nel corso dello sviluppo.

La traccia selezionata è quella riguardante la compagnia aerea:

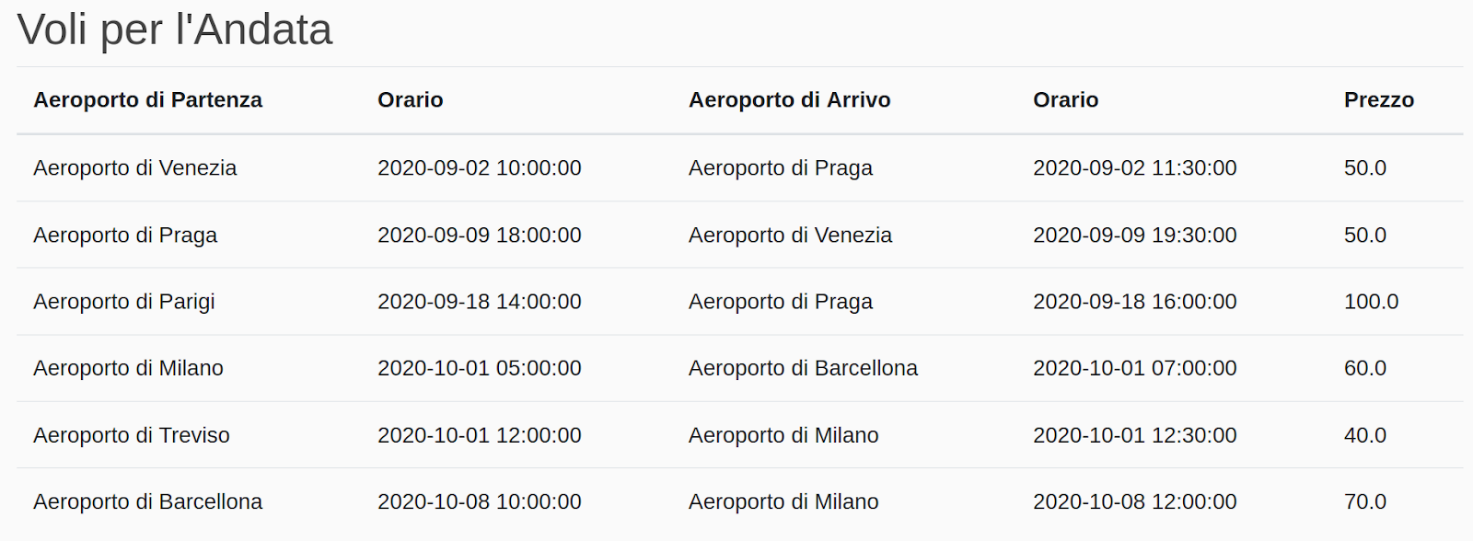
*“Gli utenti possono avere un ruolo fra Cliente ed Operatore. I Clienti possono cercare voli in base alle loro necessità di viaggio, prenotarli e selezionare un posto a sedere, mentre gli Operatori hanno il compito di amministrare l’elenco dei voli disponibili. Gli Operatori hanno anche interesse a collezionare statistiche a fini commerciali.”*

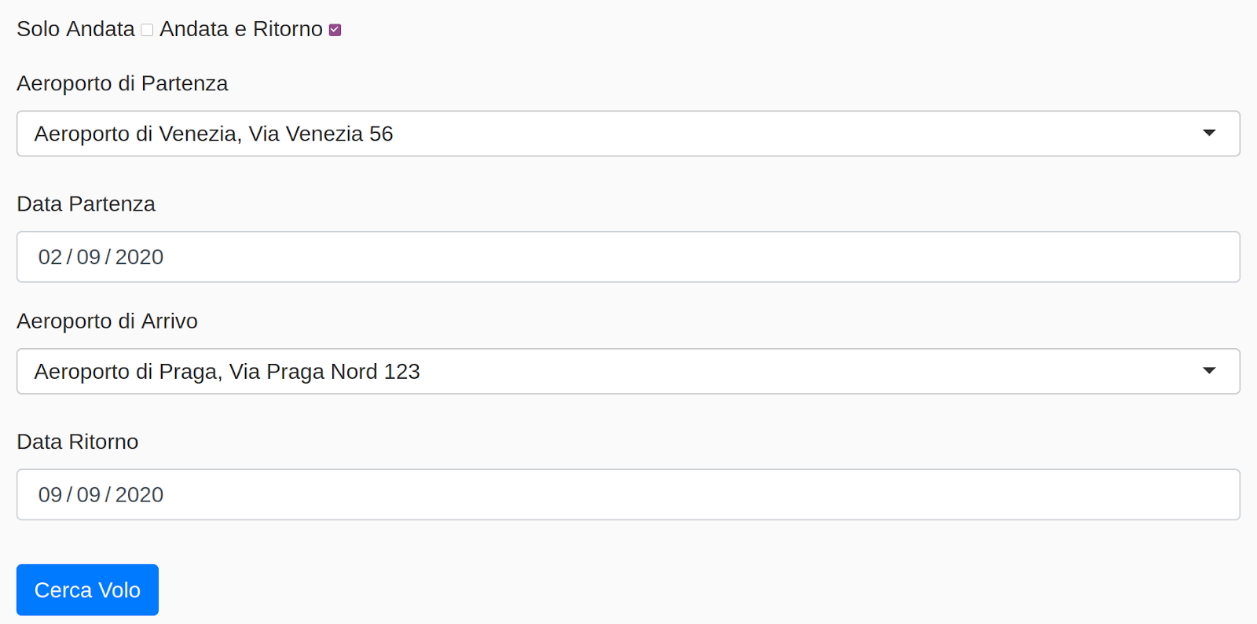
# 1. Funzionalità di TakeAFly

Pensata per gestire una compagnia aerea, permette di aggiungere voli dove i clienti possono successivamente acquistare i biglietti, e visualizzare varie informazioni che possono essere sfruttate per apportare delle modifiche ai dati inseriti.

## 1.1 Ricerca di un volo ed acquisto

Per la prima pagina che appare quando si apre l’applicazione non è necessario eseguire l’autenticazione, ed è possibile vedere tutti i voli disponibili con le relative informazioni,



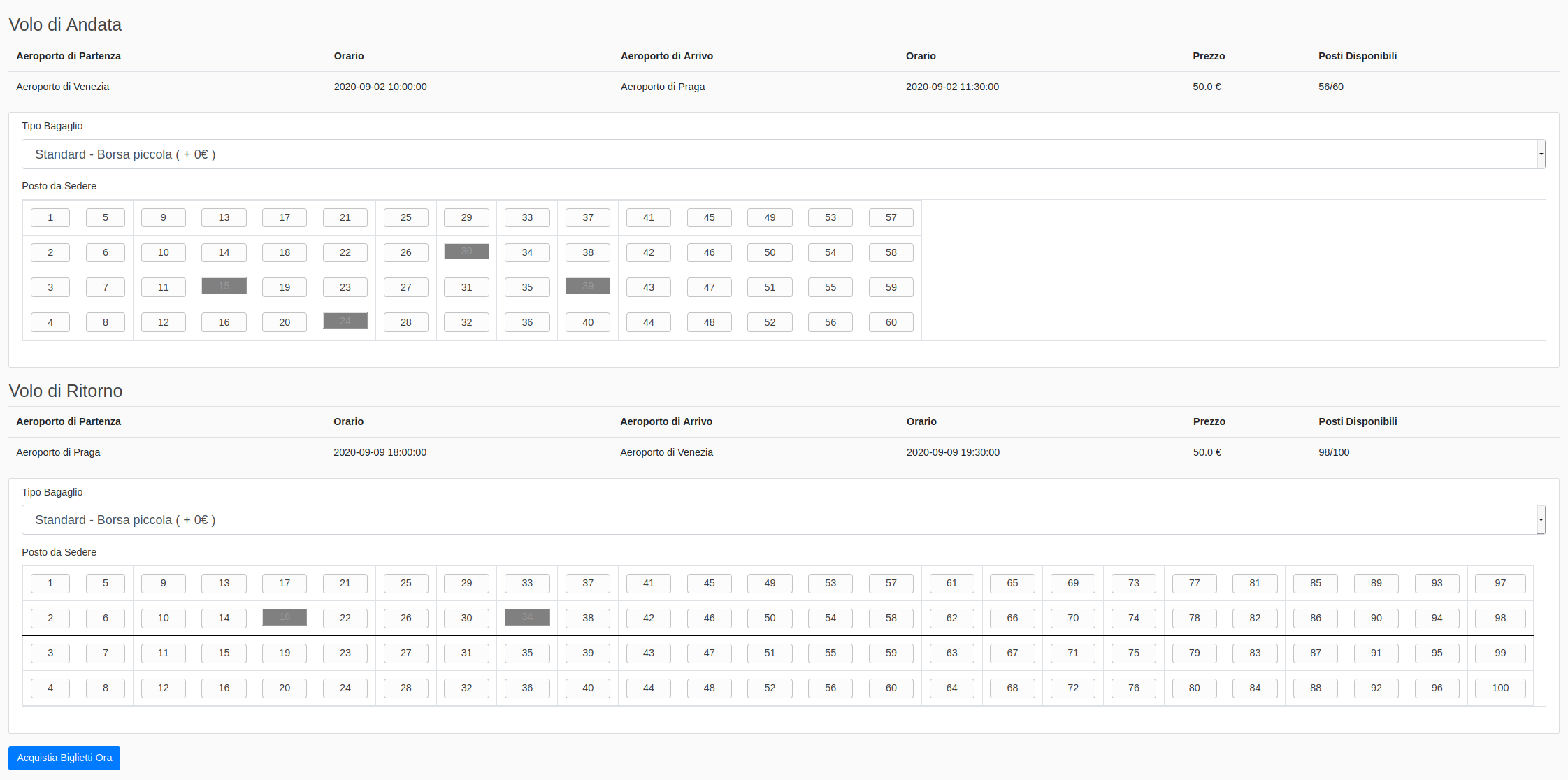
qui il cliente, tramite il form, può filtrare le opzioni scegliendo l'aeroporto da dove partire, la destinazione e la data di partenza, inoltre è possibile selezionare se acquistare un biglietto andata e ritorno, in questo caso è necessario specificare anche la data nel quale si desidera tornare

Dopo aver eseguito la ricerca della tratta a sua scelta, si potrà visualizzare tutti i voli disponibili per la tratta desiderata, voli suddivisi nel caso che l’utente abbia scelto di acquistare sia il biglietto per l’andata che quello per il ritorno.

Continuando l’interazione si dovrà selezionare la rispettiva riga rappresentante il volo che si vuole scegliere e cliccare il tasto “Continua”.



A questo punto potremmo visualizzare le informazioni del volo con in aggiunta la possibilità di poter selezionare il tipo di bagaglio da portarsi a bordo (ogni bagaglio ha un relativo sovrapprezzo da sommare al costo del volo) e di poter scegliere il posto da sedere cliccando sulla relativa mappa del volo il posto desiderato.



Successivamente dopo aver cliccato “Acquista Biglietto Ora” comparirà il modal con tutte le informazioni sulla nostra prenotazione del volo di sola andata oppure di andata e ritorno e quindi anche il costo totale del biglietto.



Se non avremo scelto un posto da sedere il sistema ce lo segnalerà mostrandoci a schermo un messaggio di errore e non ci permetterà concludere l’acquisto del biglietto.

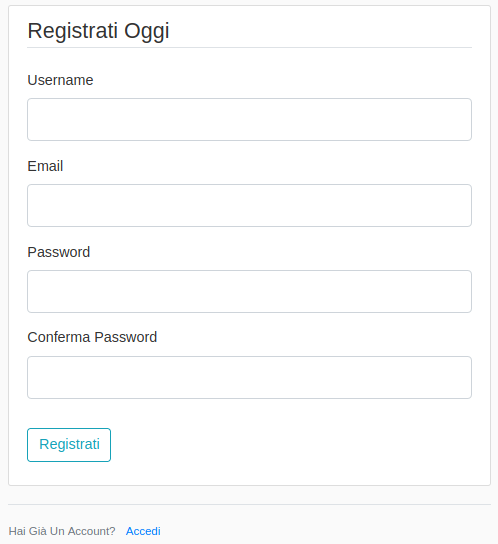
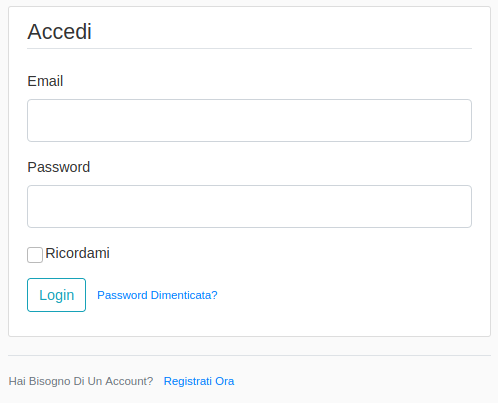
Mentre se ciò è stato fatto correttamente potremmo acquistare il biglietto solo ed esclusivamente dopo aver fatto l’accesso oppure la registrazione al sistema.

In conclusione cliccando “Conferma acquisto” si sarà reindirizzati in “user\_fly” pagina come si può ben capire dal nome suddetta alla visualizzazione dei voli acquistati dall'utente e si sarà notificati da un messaggio a schermo dell’invio di una mail con le informazioni per effettuare l’imbarco

## 1.2 Login e Registrazione

Come detto in precedenza per acquistare un biglietto è necessario essere registrati ed aver effettuato il login al sistema.

Sia il Login che la Registrazione sono forniti attraverso due semplici from in cui dopo aver inserito le credenziali ci restituirà un messaggio di successo se tutto è andato a gonfie vele mentre un messaggio di errore se abbiamo sbagliato a scrivere qualcosa oppure ci siamo dimenticati la password di accesso.



Appunto per questo è presente una sezione a parte proprio per resettare le proprie credenziali, questo è possibile dopo aver fornito al sistema la mail con cui ci siamo registrati e dopo aver atteso qualche secondo riceveremo una notifica mail dall’applicazione che ci consiglierà di cliccare al sito web indicato il quale ci permetterà sempre attraverso un form di impostare la nostra nuova password.

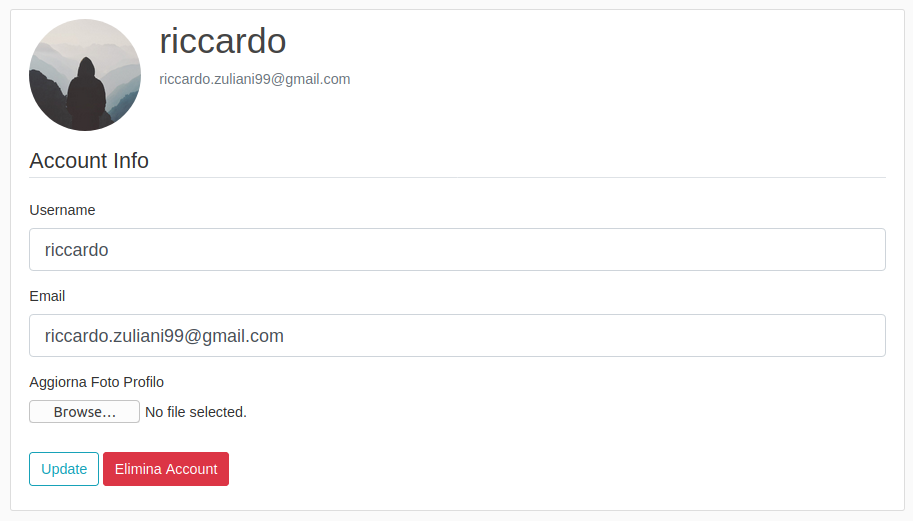
## 1.3 Gestione delle Prenotazioni

Accessibile attraverso la barra superiore è la sezione “I miei voli”, la quale ci mostra tutti i nostri voli che abbiamo eseguito e devono ancora avvenire cioè che abbiamo prenotato.



In questa sezione oltra a poter vedere delle informazioni generali per i voli è possibile inserire una recensione univoca ad un volo effettuato e eliminare prenotazioni di voli che abbiamo scelto di non effettuare.

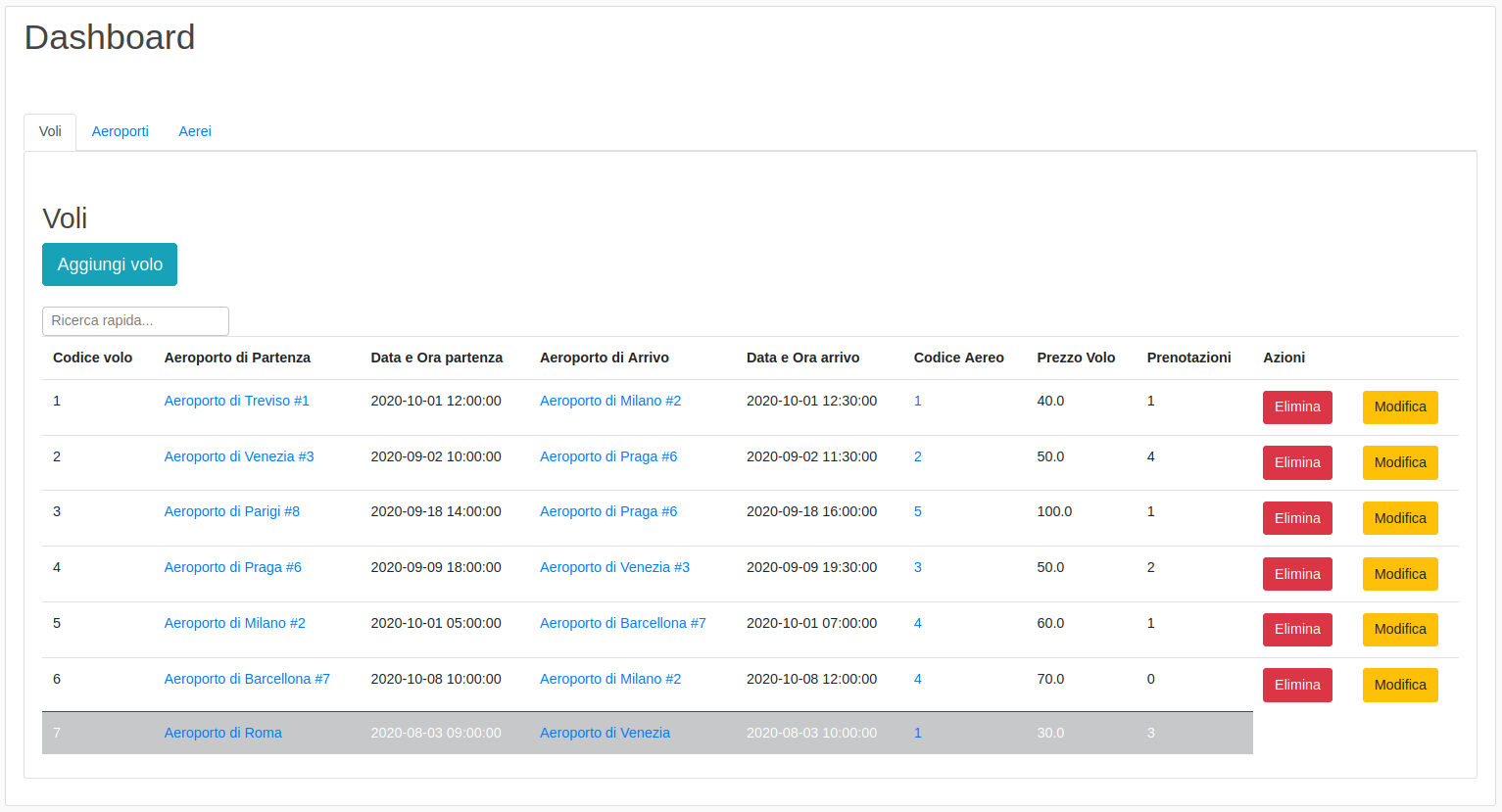
Sia l’eliminazione che l'inserimento delle prenotazioni sono gestite da dei modal in cui nel caso della recensione sarà possibile inserire una valutazione da 1 fino a 5 con relativo commento, mentre nel caso dell’eliminazione ci sarà mostrato un messaggio di conferma.

****1.4 Gestione dell’Account

La pagina di gestione del proprio account si presenta in questo modo, e come si può ben notare è possibile modificare Username, Email e anche l’immagine del profilo che al momento dell’iscrizione è impostata a quella di default ed In aggiunta è anche possibile eliminare il proprio account.

## 1.5 Dashboard

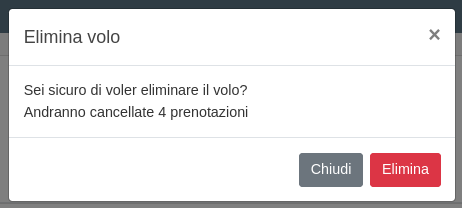
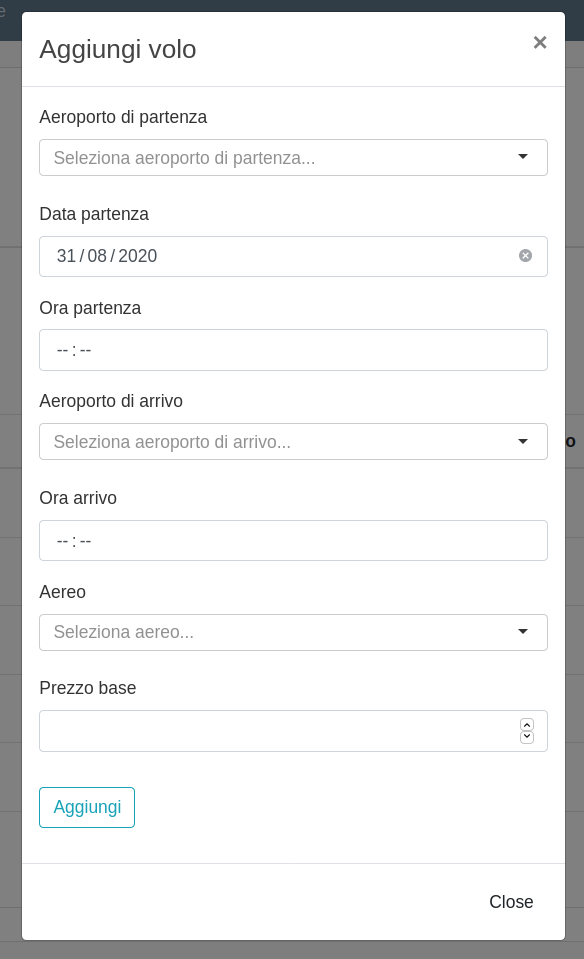
Una volta fatto l’accesso con l’account di amministratore è possibile accedere alla pagina Dashboard tramite il link in alto a sinistra, appena caricata verrà visualizzata una barra di navigazione che permette di visualizzare rispettivamente i voli, gli aeroporti e gli aerei a disposizione della compagnia in quel momento.

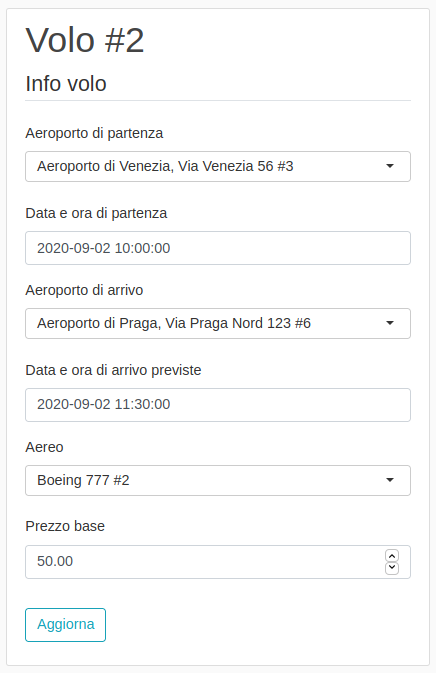


Premendo il pulsante aggiungi volo verrà aperto un modal che darà la possibilità di inserire un nuovo volo scegliendo l'aeroporto di partenza e arrivo tramite un menu a tendina che si espande mostrando il nome, l’indirizzo e l’id dell’aeroporto, per selezionarne uno è possibile cliccare con il mouse oppure cercarlo direttamente dalla barra di ricerca; è necessario inserire anche data e ora di partenza, e l’ora di arrivo, (non essendo possibile che un volo duri più di 24 ore, se viene inserita un’ora di arrivo minore rispetto a quella di partenza come data di arrivo viene automaticamente inserita quella del giorno successivo), ed infine si deve selezionare l’aereo e il prezzo base del volo. Una volta premuto il tasto “Aggiungi” se tutti i campi rispettano i vincoli il volo verrà inserito.

Successivamente come si può vedere dallo screenshot della dashboard è possibile eliminare e modificare un qualsiasi volo ad eccezione di quelli che sono già stati effettuati.

Per quanto riguarda l'eliminazione è gestita da un modal che ci avvisa se siamo sicuri di voler eliminare quel dato volo perchè andremo anche ad eliminare anche le relative prenotazioni.

Mentre per la modifica si aprirà una nuova pagina nella quale nei relativi form saranno presenti i dati attuali del volo che noi potremo modificare a nostra scelta.



Sempre nella navbar della dashboard è possibile entrare nella sezione Aeroporti dove è consentito aggiungere un nuovo aeroporto tramite modal a comparsa dove inserendo nome e l’indirizzo nel form. E’ possibile eliminare un aeroporto cliccando elimina infatti comparirà un modal di conferma eliminazione che ci avviserà che eliminando il relativo aeroporto elimineremo i suddetti voli e prenotazioni collegate ad esso. Ed in conclusione è anche possibile modificare i campi degli aeroporti cliccando nel bottone Modifica.

Azioni uguali sono possibili effettuare anche per aerei con l’unica differenza che durante l’inserimento di un nuovo aereo il numero di posti da sedere deve essere un multiplo di 4.

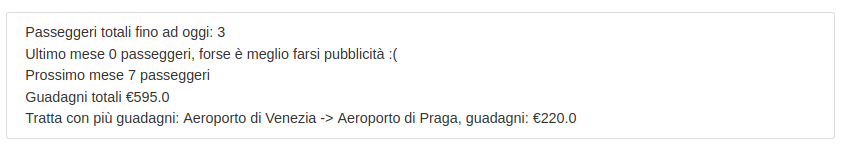
Peculiarità della dashboard è che è possibile effettuare una ricerca veloce di qualsiasi campo presente nella tabella attraverso un picciolo form di ricerca posizionato sotto il bottone di aggiunta, oltre a ciò cliccando sul nome di un aeroporto oppure di un volo saremo reindirizzati alla sua pagina di modifica

.

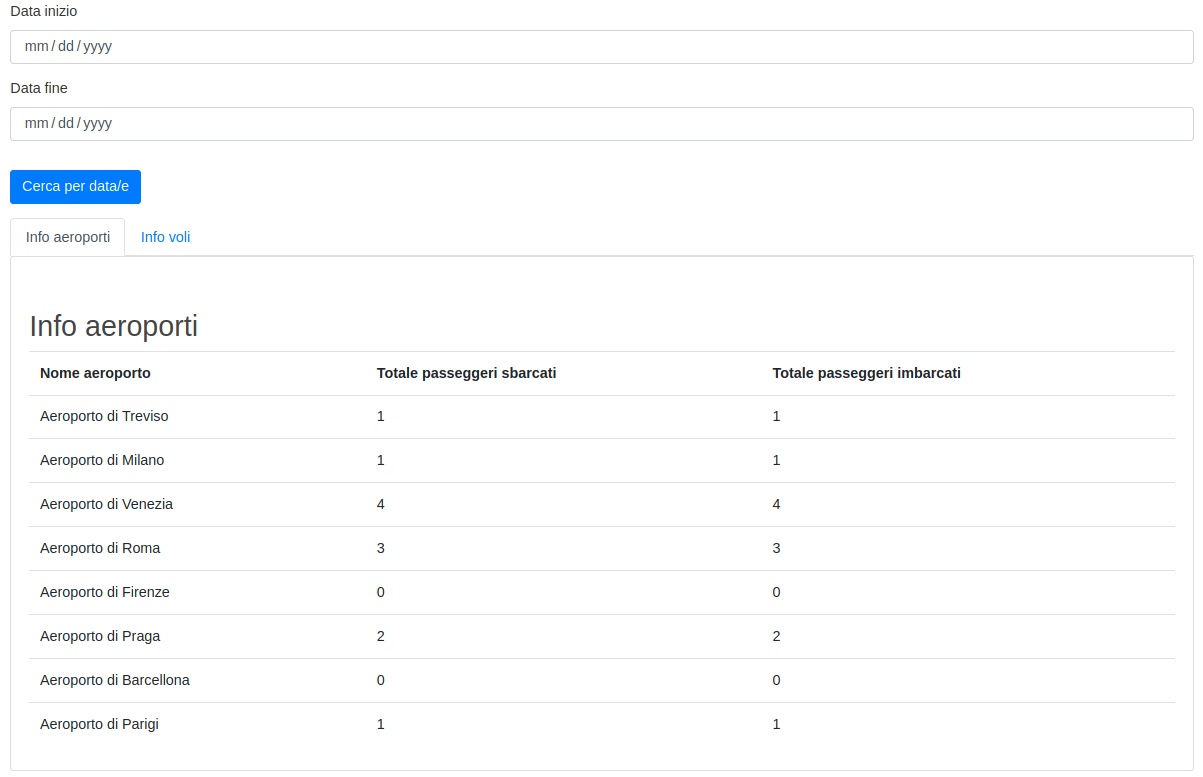
## 1.6 Statistiche

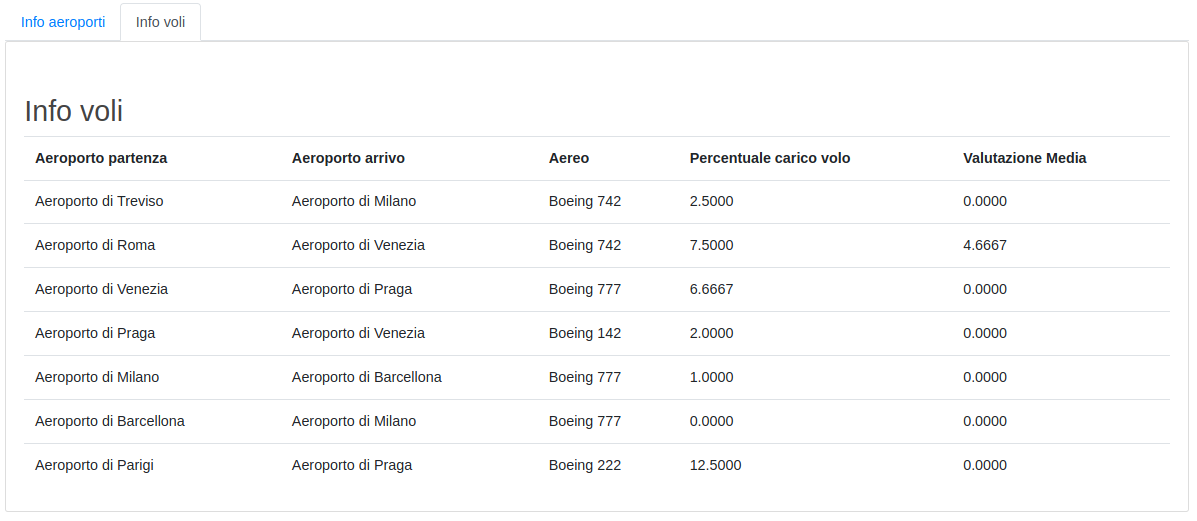
Una seconda funzionalità disponibile solo ed esclusivamente all’amministratore del sistema è la possibilità di visualizzare delle statistiche relative all'andamento attuale e anche generale della compagnia aerea TakeAFlay.

In primo luogo abbiamo delle statistiche generali utili per fornire una visione generale e approssimativa sui guadagni e sull’utilizzo della piattaforma da parte dei clienti.



Oltre a ciò è possibile visualizzare la lista degli aeroporti con il numero totale di passeggeri in arrivo e in partenza da ogni aeroporti e cambiando sezione tramite una navbar e possibile visualizzare anche la lista dei voli con la relativa percentuale di carico dell’aereo e valutazione media (inseribile solo dopo l’atterraggio del volo).





In queste due funzionalità è possibile inserire la data di inizio e di fine analisi tramite un classico form di inserimento date, è possibile anche inserire una solo e il sistema computerà dalla data di inizio fino alla fine del database se abbiamo inserito solo la prima data dell'analisi mentre dall’inizio del data base fino alla data di fine se abbiamo inserito la seconda data.

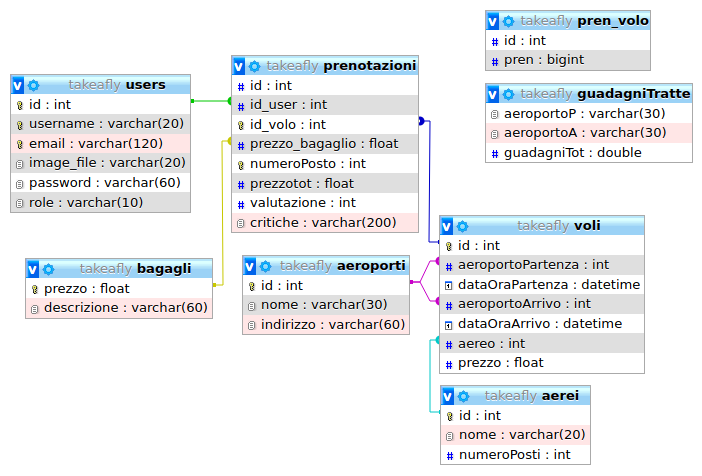
# 2. La Base di Dati

* Rappresentazione della base di dati attraverso un modello a oggetti:

Immagine che contiene screenshot

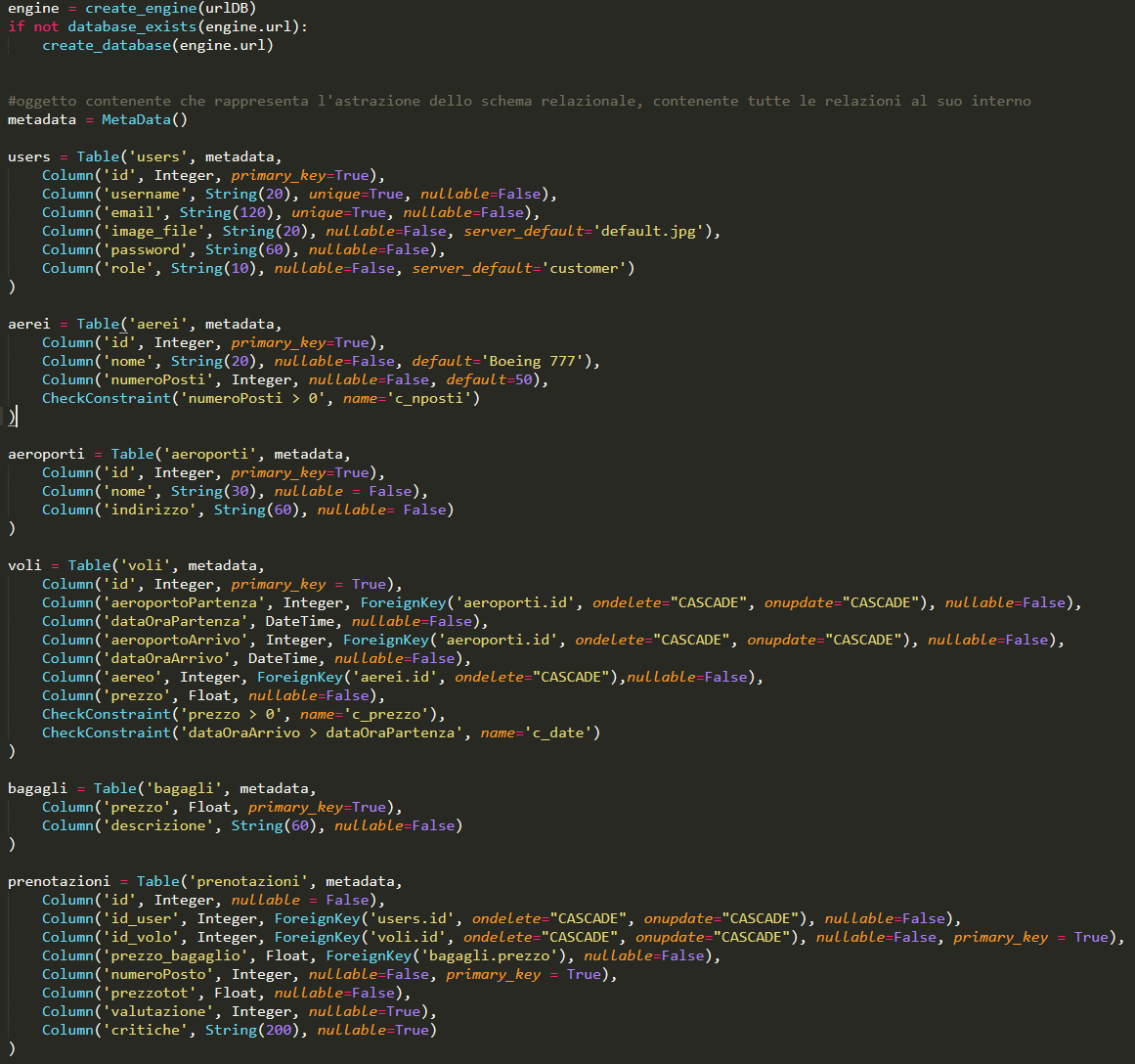
Descrizione generata automaticamente

* Rappresentazione della base di dati attraverso lo schema relazionale:



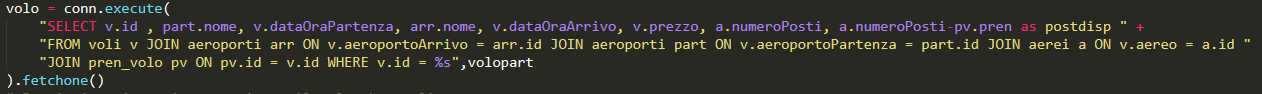
Per la definizione del DataBase abbiamo sfruttato la libreria SQLAlchemy, all’interno del file table.py inizialmente creiamo l’engine importando del pakage file \_\_init\_\_.py l’urlDB e se il database non esiste lo creiamo, continuando creiamo l’oggetto contenente che rappresenta l'astrazione dello schema relazionale, contenente tutte le relazioni al suo interno cioè il metadata e successivamente definiamo le seguenti tabelle:

* Users: tabella che raggruppa sia i clienti che gli amministratori infatti abbiamo optato per una selezione unica visto che le due categorie differenziavano solo ed esclusivamente dal campo “role”. Essa è identificata univocamente dal suo id, ha un username, ha una mail, un’immagine di profilo, una password ed il ruolo che di default è impostato a “customer”.
* Bagagli: contiene esclusivamente i tre tipi di bagagli che sono è possibile portare a bordo durante il volo, ogni bagaglio è identificato univocamente dal proprio prezzo e ha una piccola descrizione aggiuntiva.
* Aerei: ogni aereo è identificato dal proprio id ha un proprio nome che di default è impostato a “Boeing 777” ed ha un numeroPosti che deve essere maggiore di zero e multiplo di 4
* Aeroporti: gli aeroporti sono identificati univocamente dal loro id hanno un nome e un indirizzo.
* Voli: i voli sono identificati univocamente dal loro id, hanno tre chiavi esterne infatti ogni volo ha un aeroporto di partenza ed una di arrivo e ogni volo è effettuato da un aereo, in aggiunta hanno un orario di partenza e di arrivo e un prezzo base.
* Prenotazioni: è la tabella principale infatti è addetta alla memorizzazione di tutte le prenotazioni effettuate dai clienti della compagnia, è identificata univocamente da numeroPosto e dall’id del volo il quale è anche una chiave esterna per voli, ogni prenotazione si riferisce ad un cliente, ha un prezzo aggiuntivo riferito al tipo di bagaglio scelto, un prezzo totale che è dato dalla somma tra il prezzo standard e quello del bagaglio e piò avere una valutazione con annesse critiche.

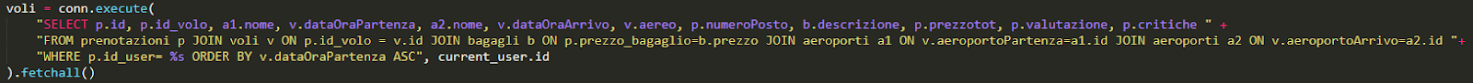


# 3. Query Interessanti

* Nel file route.py di fly per avere tutte le informazioni necessarie del volo che vogliamo acquistare devo andare a selezionare i campi necessari dalla tabella voli aeroporti ed aerei, inoltre grazie all’uso di una vista posso calcolarmi il numero di posti disponibili per quel volo. Sempre grazie alla vista restituisco i posti occupati, dato che mi sarà utile successivamente per creare la lista dei posti disponibili usata per creare la mappa.

https://lh3.googleusercontent.com/sg6O_8qFxCnwYgEErTiLGiLBZExtrt1gtA7_tbnM1Qrnafsuiy86hRvJzFS4oE9whA3aEfbYMHRUO6iFHiavaVx1hSmdHQHpTKreuPWSG6Y86OSwGGE0ICzcjAjTYiWViYrXdMk5

* Nel file route.py di users nella funzione in cui si vanno a calcolare tutti i voli che ha eseguito l’utente andiamo a fare una selezione dei campi necessari nelle tabelle voli bagagli ed aeroporti.

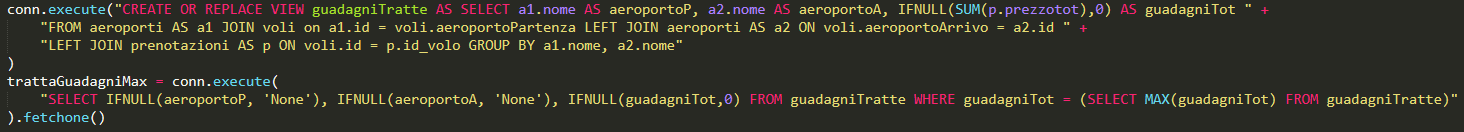


* Nel file route.py di statistics abbiamo le query più interessanti che utilizziamo per avere una visione generale di come sta andando la nostra compagnia aerea.

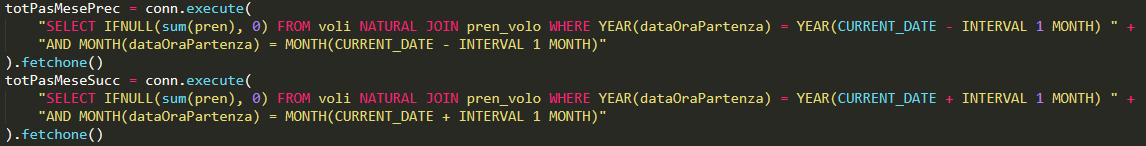
Come prima query abbiamo i guadagni totali assoluti, cioè calcolati su tutta la tabella prenotazioni;

https://lh6.googleusercontent.com/KIhFkMPCUtwoN3jjaue6zD3PkXuzg5Tz7Ukds717trFGs-z5ItaBUl_7qvITn7Kv6fkGanLUr-0r1EGOXNSV4kiyW_fcozNjQM4FNeCsv1GDCyloo-ClAKm7_ALfIQMVPRHMUvp-

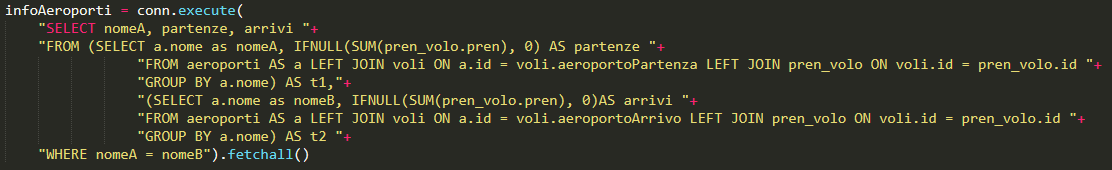
In questa query andiamo innanzi tutto a crearci una vista di appoggio che ci calcola la somma di tutti i prezzi dei biglietti acquistati per ogni tratta e successivamente restituiamo la tratta che ha più guadagni in totale



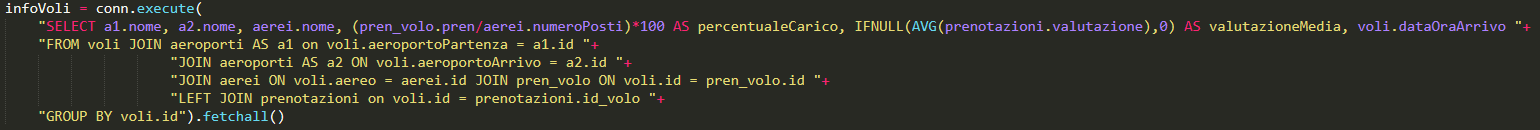
Con queste due query invece andiamo a calcolare il numero di passeggeri totali che abbiamo avuto nel mese precedente e che avremo attualmente nel mese successivo;



In infoAeroporti andiamo a salvare la lista dei aeroporti con relativo nome, totale di passeggeri in arrivo e in partenza da quel relativo aeroporto;



E per concludere in infoVoli andiamo a salvare i voli con relativa percentuale di posto occupati finora e con la relativa valutazione media assegnata dai clienti che hanno effettuato quel volo.



# 

# 4. Scelte progettuali

* **Bootstrap**:

famosa raccolta di strumenti per la creazione di siti web utilizzata per rendere il sito web più apprezzabile alla vista dell’utente

* **jQuery**:

libreria di JavaScript per applicazioni web utilizzata per rendere il sito dinamico in certi particolari e per effettuare lo scambio di informazioni tra parti della stessa pagina

* **Jinja**:

è il motore di template più usato da Python, il quale permette di creare file in vari formati di markup. Esso utilizza due sequenze di {} come delimitatori, la prima {%  %} è utilizzata per inserire parole chiavi come “block content”, “end Block content” oppure i costrutti iterativi cioè l’if e il for each, mentre la seconda sequenza {{}} è utilizzata per espressioni o variabili che vogliamo compiere o semplicemente stampare a schermo. Queste due forme sono state largamente utilizzate proprio per stampare a schermo i dati inviati da Flask oppure per differenziare alcune parti della pagina web per solo i clienti o amministratori.

* **PhpMyAdmin**:

applicazione web la quale ci ha aiutato nella gestione del database MySQL velocizzando operazione di inserimento oppure di modifica per debuggare o verificare le funzionalità di TakeAFly.

* **Flask** **WTForms**:

è una libreria flessibile per la convalida e il rendering di moduli per lo sviluppo web Python. Può funzionare con qualsiasi framework web e motore di template tu scelga. Supporta la convalida dei dati, la protezione CSRF, l'internazionalizzazione (I18N) e altro ancora. I moduli forniscono l'API di livello più alto in WTForms. Contengono le definizioni dei campi, la convalida dei delegati, l'inserimento di input, gli errori aggregati e in generale funzionano come il collante che tiene tutto insieme. ([https://wtforms.readthedocs.io/en/2.3.x/#](https://wtforms.readthedocs.io/en/2.3.x/))

In aggiunta grazie a questa libreria siamo stati in grado di inserire dei validatori personalizzati per i from disponibili, in questo modo si ha una sicurezza maggiore che l’utente inserisca un valore corretto

* **FLask** **Mail**:

estensione che permette l’invio di messaggi tramite interfaccia SMTP. Essa richiede ovviamente un indirizzo email che abbia però disattivata la funzione del controllo a due passi e permetta l’accesso ad applicazioni poco sicure.

* **Falsk** **Blueprint**:

Blueprint è un modo di organizzare l’applicazione Flask per renderla suddivisa in gruppi di funzionalità, infatti ad ogni gruppo è assegnata una cartella nella quale sarà presente un file route.py il quale conterrà tutte le route specifiche per quella funzione, utils.py se sono presenti delle fuznioni ausiliarie utilizzate da alcune route e form.py che sppecifica i form utilizzati in quella parte.

* **Flask**\_**util\_js:**

è una libreria sviluppata *dantezhu* che permette di usare alcune importanti funzionalità di Flask in JavaScript, una fra tutte è url\_for che viene convertita nel seguente codice JavaScript:

var url = flask\_util.url\_for('route', {parametro1: valore1, parametro2: valore2});

* **Transazioni**:

Nel file route.py di fly sono presenti due transazioni con livello di isolamento SERIALIZABLE:

* La prima transazione si riferisce al momento in cui si acquista un biglietto di sola andata, infatti questa operazione deve essere eseguita in mutua esclusione perché non si vuole che due utenti abbiamo prenotato lo stesso posto da sedere per lo stesso volo. Se dovesse avvenire un’eccezione si ha la necessità di ritornare ad uno stato consistente della base di dati perciò si va ad eseguire un’operazione di rollback
* La seconda transazione si riferisce sempre nel caso in cui si acquistano due biglietti infatti volgiamo che questa operazione sia fatta in mutua esclusione sia per l’acquisto del biglietto di andata che per l’acquisto del biglietto di ritorno.

Anche in questo caso nell’eventualità di eccezioni si esegue l’operazione di rollback

Nel file route.py di statistics abbiamo impostato tutte le query con livello di isolamento READ UNCOMMITTED proprio perché si stanno facendo dei calcoli statistici e quindi si ha una soglia di tolleranza.

* **Indici:**

Nel file table.py vengono definiti due indici vengono definiti due indici al fine di accelerare l’esecuzione delle query che utilizzano questi attributi. Il primo indice si riferisce alla tabella prenotazioni, mentre il secondo indice riguarda la tabella volo più precisamente all’id dell’aeroporto di partenza e a quello di arrivo.

* **Trigger**:

abbiamo deciso di inserire due trigger nella nostra applicazione perché trattandosi di una compagnia aerea i casi in cui il dato può essere inserito in malo modo sono molti.

* Il primo trigger è relativo all’aumento del prezzo di un volo se la capienza totale dell’aereo utilizzato è > del 50%, abbiamo voluto fare questo trigger per simulare in maniera un po’ grossolana il processo per cui quando ci sono pochi posti disponibili e quindi la domanda è alta il prezzo sale leggermente.
* Il secondo trigger invece è relativo al controllo durante l'inserimento di un nuovo volo, infatti se quest'ultimo coincide con un altro volo per aeroporto di partenza e per orario di partenza, andremo a spostare di un’ora più avanti il volo appena inserito, in modo da non avere più di un volo che parte nello stesso momento in ogni aeroporto.
* **Vincoli**:
* CheckConstraint(prezzo > 0, name=’c\_prezzo’): controlla che il prezzo inserito dall’amministratore sia maggiore di 0
* CheckConstraint(dataOraArrivo > dataOraPartenza, name=’c-date’):controlla che durante l’inserimento di un nuovo volo da parte dell’amministratore la data di arrivo sia maggiore rispetto al data di partenza dello stesso.
* CheckConstraint(numeroPosti > 0=’c-date’): controlla che il numero di posti da sedere al momento di inserire o aggiornare un aereo sia maggiore di 0.
* **Textual** **SQL**:

abbiamo deciso di adottare textual sql invece del Expression Language perchè è più semplice testare le query all’interno di PhpMyAdmin infatti è sufficiente fare un classico copia e incolla per verificare la correttezza o meno di essa.

# 5. Considerazioni Finali

In conclusione possiamo dire di essere soddisfatti del lavoro fatto, ad inizio progetto certo non avremmo mai scommesso di arrivare a questo punto con tutte le implementazioni fatta, certamente c’è stata d’aiuto la documentazione fornita nel sito ufficiale di Flask e anche qualche video tutorial su come impostare il progetto ed i form.

# 

# 6. Come far partire TakeAFly

All’interno della cartella aeroporto aprire il file\_\_init\_\_.py e modificare con i propri paramentri di accesso username e password, una volta fatto ciò, salvate il file recarsi alla cartella generale dove risiede il file run.py, aprire il terminale in quella cartella e semplicemente digitare *python3 run.py*.

Così facendo solo con questo comando faremo partire l’applicazione già in debug mode senza digitare *export FLASK\_APP=run.py; export FLASK\_ENV=development; flask run.*

E’ possibile che il terminale richieda di installare i pacchetti necessari per far funzionare l’applicazione nell’eventualità che non siano già installati.