



Università
Ca' Foscari
Venezia

TakeAFly

Progetto corso Basi di Dati
A.A 2019/2020

Nome gruppo: TakeAFly SpA

Gatto Michele 875446

Zuliani Riccardo 875532

Sommario

Introduzione	3
1. Funzionalità di TakeAFly	4
1.1 Ricerca di un volo ed acquisto	4
1.2 Login e Registrazione.....	6
1.3 Gestione delle Prenotazioni.....	7
1.4 Gestione dell'Account.....	8
1.5 Dashboard.....	8
1.6 Statistiche	11
2. La Base di Dati	13
3. Query Interessanti.....	16
4. Scelte progettuali	17
5. Considerazioni Finali.....	19
6. Come far partire TakeAFly	19

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 ma comunque apprezzabile.

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,

Voli per l'Andata				
Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Prezzo
Aeroporto di Venezia	2020-09-02 10:00:00	Aeroporto di Praga	2020-09-02 11:30:00	50.0
Aeroporto di Praga	2020-09-09 18:00:00	Aeroporto di Venezia	2020-09-09 19:30:00	50.0
Aeroporto di Parigi	2020-09-18 14:00:00	Aeroporto di Praga	2020-09-18 16:00:00	100.0
Aeroporto di Milano	2020-10-01 05:00:00	Aeroporto di Barcellona	2020-10-01 07:00:00	60.0
Aeroporto di Treviso	2020-10-01 12:00:00	Aeroporto di Milano	2020-10-01 12:30:00	40.0
Aeroporto di Barcellona	2020-10-08 10:00:00	Aeroporto di Milano	2020-10-08 12:00:00	70.0

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 i biglietti per andata e ritorno, in questo caso è necessario specificare anche la data nel quale si desidera tornare.

Solo Andata ☐ Andata e Ritorno ☒

Aeroporto di Partenza

Aeroporto di Venezia, Via Venezia 56

Data Partenza

02 / 09 / 2020

Aeroporto di Arrivo

Aeroporto di Praga, Via Praga Nord 123

Data Ritorno

09 / 09 / 2020

Cerca Volo

Dopo aver eseguito la ricerca della tratta scelta, si potranno visualizzare tutti i voli disponibili, i quali saranno suddivisi per andata e per ritorno nel caso che l'utente abbia scelto di acquistare i due biglietti.

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

Voli per l'Andata

Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Prezzo
Aeroporto di Venezia	2020-09-02 10:00:00	Aeroporto di Praga	2020-09-02 11:30:00	50.0

Voli per il Ritorno

Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Prezzo
Aeroporto di Praga	2020-09-09 18:00:00	Aeroporto di Venezia	2020-09-09 19:30:00	50.0

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.

Volo di Andata

Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Prezzo	Posti Disponibili
Aeroporto di Venezia	2020-09-02 10:00:00	Aeroporto di Praga	2020-09-02 11:30:00	50.0 €	56/60

Tipo Bagaglio

Standard - Borsa piccola (+ 0€)

Posto da Sedere

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60

Volo di Ritorno

Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Prezzo	Posti Disponibili
Aeroporto di Praga	2020-09-09 18:00:00	Aeroporto di Venezia	2020-09-09 19:30:00	50.0 €	98/100

Tipo Bagaglio

Standard - Borsa piccola (+ 0€)

Posto da Sedere

1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61	65	69	73	77	81	85	89	93	97
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62	66	70	74	78	82	86	90	94	98
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63	67	71	75	79	83	87	91	95	99
4	8	12	16	20	24	28	32	36	40	44	48	52	56	60	64	68	72	76	80	84	88	92	96	100

Acquista Biglietti Ora

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

Resoconto Biglietti

Volo di Andata

Codice Volo: 2

Aeroporto di Partenza: Aeroporto di Venezia

Orario: 2020-09-02 10:00:00

Aeroporto di Arrivo: Aeroporto di Praga

Orario: 2020-09-02 11:30:00

Numero posto da sedere: 34

Totale: 50€ + 20€ = 70€

Volo di Ritorno

Codice Volo: 4

Aeroporto di Partenza: Aeroporto di Praga

Orario: 2020-09-09 18:00:00

Aeroporto di Arrivo: Aeroporto di Venezia

Orario: 2020-09-09 19:30:00

Numero posto da sedere: 47

Totale: 50€ + 40€ = 90€

Prezzo Totale Biglietti: 160€

Chiudi

Conferma Acquisto

Resoconto Biglietti

Volo di Andata

Codice Volo: 2

Aeroporto di Partenza: Aeroporto di Venezia

Orario: 2020-09-02 10:00:00

Aeroporto di Arrivo: Aeroporto di Praga

Orario: 2020-09-02 11:30:00

Scegliere un posto nella mappa

Totale: 50€ + 20€ = 70€

Volo di Ritorno

Codice Volo: 4

Aeroporto di Partenza: Aeroporto di Praga

Orario: 2020-09-09 18:00:00

Aeroporto di Arrivo: Aeroporto di Venezia

Orario: 2020-09-09 19:30:00

Scegliere un posto nella mappa

Totale: 50€ + 40€ = 90€

Prezzo Totale Biglietti: 160€

Chiudi

Conferma Acquisto

Se non si è 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 si potrà 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 form 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.

Registrati Oggi

Username

Email

Password

Conferma Password

Registrati

Hai Già Un Account? [Accedi](#)

Accedi

Email

Password

☐ Ricordami

Login

[Password Dimenticata?](#)

Hai Bisogno Di Un Account? [Registrati Ora](#)

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 si è registrati e dopo aver atteso qualche secondo si riceverà una notifica mail dall'applicazione che consiglierà di cliccare al sito web indicato il quale permetterà sempre attraverso un form di impostare la nostra nuova password.

1.3 Gestione delle Prenotazioni

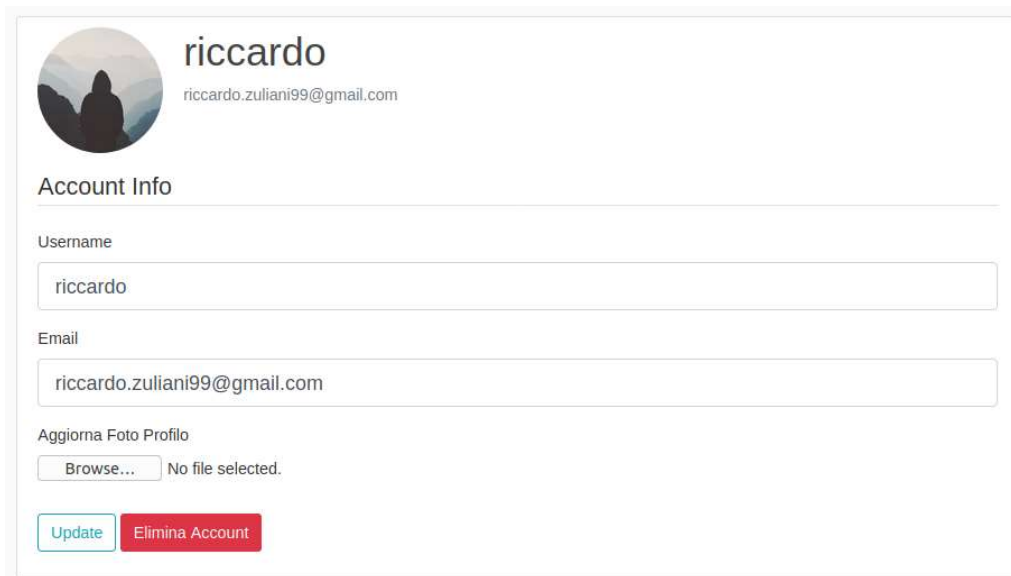
Accessibile attraverso la barra superiore è la sezione “I miei voli”, la quale mostra tutti i voli che l'utente ha eseguito e che ha prenotato.

I miei voli										
Codice Prenotazione	Codice volo	Aeroporto di Partenza	Orario	Aeroporto di Arrivo	Orario	Codice Aereo	Numero Posto da Sedere	Bagaglio	Prezzo Totale	Azioni
1	7	Aeroporto di Roma	2020-08-03 09:00:00	Aeroporto di Venezia	2020-08-03 10:00:00	1	15	Standard - Borsa piccola (+ 0€)	15.0	Aggiungi Recensione
2	2	Aeroporto di Venezia	2020-09-02 10:00:00	Aeroporto di Praga	2020-09-02 11:30:00	2	30	Plus - Bagaglio a mano da 10 Kg e borsa piccola (+ 20€)	70.0	Elimina

In questa sezione oltre 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 si ha scelto di non effettuare.

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

1.4 Gestione dell'Account



The screenshot shows a user profile for 'riccardo' with the email 'riccardo.zuliani99@gmail.com'. The profile picture is a default silhouette of a person against a landscape. Below the profile information, there is a section titled 'Account Info' with two input fields: 'Username' containing 'riccardo' and 'Email' containing 'riccardo.zuliani99@gmail.com'. Below these fields is a section titled 'Aggiorna Foto Profilo' with a 'Browse...' button and the text 'No file selected.'. At the bottom of the form are two buttons: 'Update' (light blue) and 'Elimina Account' (red).

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, in aggiunta a ciò è anche possibile eliminare il proprio account, andando però ad eliminare anche le suddette prenotazioni effettuate.

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.

Dashboard

Voli

Aeroporti

Aerei

Voli

Aggiungi volo

Ricerca rapida...

Codice volo	Aeroporto di Partenza	Data e Ora partenza	Aeroporto di Arrivo	Data e Ora arrivo	Codice Aereo	Prezzo Volo	Prenotazioni	Azioni
1	Aeroporto di Treviso #1	2020-10-01 12:00:00	Aeroporto di Milano #2	2020-10-01 12:30:00	1	40.0	1	<div>Elimina</div> <div>Modifica</div>
2	Aeroporto di Venezia #3	2020-09-02 10:00:00	Aeroporto di Praga #6	2020-09-02 11:30:00	2	50.0	4	<div>Elimina</div> <div>Modifica</div>
3	Aeroporto di Parigi #8	2020-09-18 14:00:00	Aeroporto di Praga #6	2020-09-18 16:00:00	5	100.0	1	<div>Elimina</div> <div>Modifica</div>
4	Aeroporto di Praga #6	2020-09-09 18:00:00	Aeroporto di Venezia #3	2020-09-09 19:30:00	3	50.0	2	<div>Elimina</div> <div>Modifica</div>
5	Aeroporto di Milano #2	2020-10-01 05:00:00	Aeroporto di Barcellona #7	2020-10-01 07:00:00	4	60.0	1	<div>Elimina</div> <div>Modifica</div>
6	Aeroporto di Barcellona #7	2020-10-08 10:00:00	Aeroporto di Milano #2	2020-10-08 12:00:00	4	70.0	0	<div>Elimina</div> <div>Modifica</div>
7	Aeroporto di Roma	2020-08-03 09:00:00	Aeroporto di Venezia	2020-08-03 10:00:00	1	30.0	3	

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 avvisa se si è sicuri di voler eliminare quel dato volo perchè si andrà 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 si potrà modificare a propria scelta.

The image displays three screenshots of a flight management system interface:

- Aggiungi volo (Add flight) modal:** A form for adding a new flight. It includes fields for 'Aeroporto di partenza' (Departure airport), 'Data partenza' (Departure date), 'Ora partenza' (Departure time), 'Aeroporto di arrivo' (Arrival airport), 'Ora arrivo' (Arrival time), 'Aereo' (Aircraft), and 'Prezzo base' (Base price). A green 'Aggiungi' button is at the bottom left, and a 'Close' button is at the bottom right.
- Elimina volo (Delete flight) modal:** A confirmation modal asking 'Sei sicuro di voler eliminare il volo?' (Are you sure you want to delete the flight?). It states 'Andranno cancellate 4 prenotazioni' (4 bookings will be cancelled). It has a grey 'Chiudi' (Close) button and a red 'Elimina' (Delete) button.
- Volo #2 (Flight #2) details page:** A page showing the details of a specific flight. It includes sections for 'Info volo' (Flight info) and 'Aereo' (Aircraft). The details shown are: 'Aeroporto di partenza' (Aeroporto di Venezia, Via Venezia 56 #3), 'Data e ora di partenza' (2020-09-02 10:00:00), 'Aeroporto di arrivo' (Aeroporto di Praga, Via Praga Nord 123 #6), 'Data e ora di arrivo previste' (2020-09-02 11:30:00), 'Aereo' (Boeing 777 #2), and 'Prezzo base' (50.00). A green 'Aggiorna' (Update) button is at the bottom.

Sempre nella navbar della dashboard è possibile entrare nella sezione Aeroporti dove è consentito aggiungere un nuovo aeroporto tramite modal a comparsa inserendo nome ed indirizzo nel form. E' possibile eliminare un aeroporto cliccando elimina infatti comparirà un modal di conferma eliminazione che avviserà che eliminando il relativo aeroporto si andrà ad eliminare i suddetti voli e prenotazioni collegate ad esso. 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 il fatto di poter effettuare una ricerca veloce di qualsiasi campo presente nella tabella attraverso un piccolo form di ricerca posizionato sotto il bottone di aggiunta, oltre a ciò cliccando sul nome di un aeroporto oppure di un aereo si sarà 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 TakeAFly.

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

Passeggeri totali fino ad oggi: 3
Ultimo mese 0 passeggeri, forse è meglio farsi pubblicità :(
Prossimo mese 7 passeggeri
Guadagni totali €595.0
Tratta con più guadagni: Aeroporto di Venezia -> Aeroporto di Praga, guadagni: €220.0

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).

Data inizio

mm / dd / yyyy

Data fine

mm / dd / yyyy

Cerca per data/e

Info aeroporti

Info voli

Info aeroporti

Nome aeroporto	Totale passeggeri sbarcati	Totale passeggeri imbarcati
Aeroporto di Treviso	1	1
Aeroporto di Milano	1	1
Aeroporto di Venezia	4	4
Aeroporto di Roma	3	3
Aeroporto di Firenze	0	0
Aeroporto di Praga	2	2
Aeroporto di Barcellona	0	0
Aeroporto di Parigi	1	1

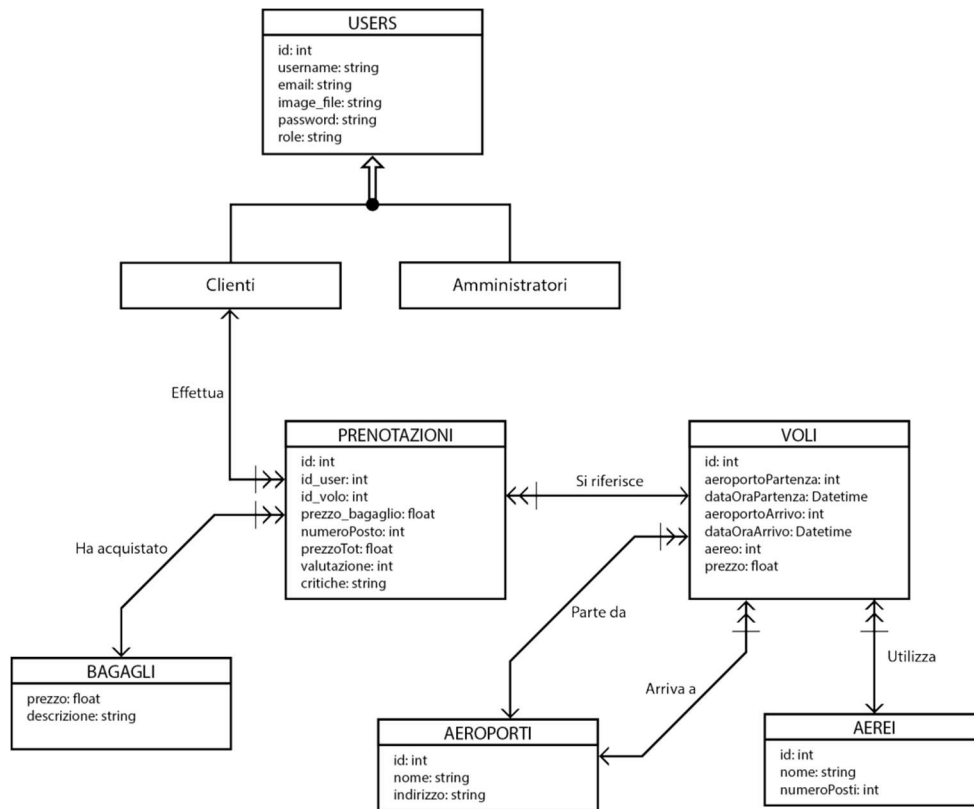
Info voli

Aeroporto partenza	Aeroporto arrivo	Aereo	Percentuale carico volo	Valutazione Media
Aeroporto di Treviso	Aeroporto di Milano	Boeing 742	2.5000	0.0000
Aeroporto di Roma	Aeroporto di Venezia	Boeing 742	7.5000	4.6667
Aeroporto di Venezia	Aeroporto di Praga	Boeing 777	6.6667	0.0000
Aeroporto di Praga	Aeroporto di Venezia	Boeing 142	2.0000	0.0000
Aeroporto di Milano	Aeroporto di Barcellona	Boeing 777	1.0000	0.0000
Aeroporto di Barcellona	Aeroporto di Milano	Boeing 777	0.0000	0.0000
Aeroporto di Parigi	Aeroporto di Praga	Boeing 222	12.5000	0.0000

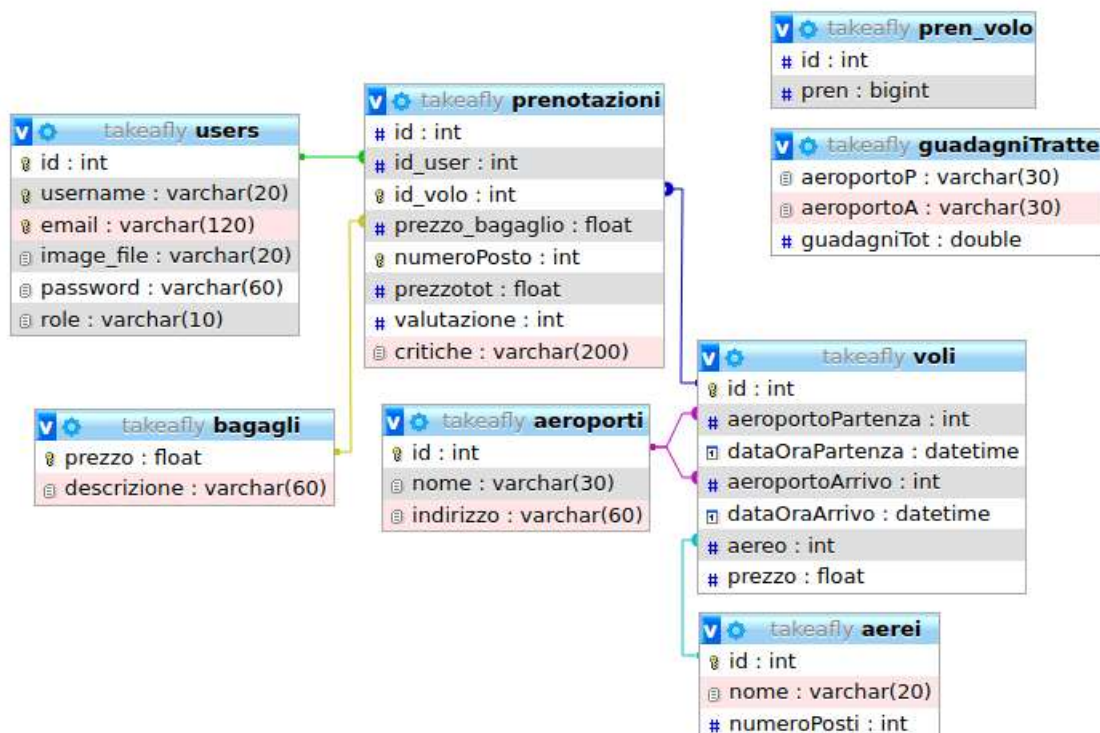
In queste due funzionalità è possibile inserire la data di inizio e di fine analisi tramite un classico form di inserimento date, è possibile anche inserire una solo e il sistema computerà dalla data di inizio in poi se abbiamo inserito solo la prima data dell'analisi mentre dalla data di fine e precedenti se abbiamo inserito la seconda data.

2. La Base di Dati

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



- 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 package file __init__.py l'urlDB e se il database non esiste lo creiamo, continuando istanziamo l'oggetto 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 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 ed 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 un di arrivo ed 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 può avere una valutazione con annesse critiche.


```

engine = create_engine(urlDB)
if not database_exists(engine.url):
    create_database(engine.url)

#oggetto contenente che rappresenta l'astrazione dello schema relazionale, contenente tutte le relazioni al suo interno
metadata = MetaData()

users = Table('users', metadata,
    Column('id', Integer, primary_key=True),
    Column('username', String(20), unique=True, nullable=False),
    Column('email', String(120), unique=True, nullable=False),
    Column('image_file', String(20), nullable=False, server_default='default.jpg'),
    Column('password', String(60), nullable=False),
    Column('role', String(10), nullable=False, server_default='customer')
)

aerei = Table('aerei', metadata,
    Column('id', Integer, primary_key=True),
    Column('nome', String(20), nullable=False, default='Boeing 777'),
    Column('numeroPosti', Integer, nullable=False, default=50),
    CheckConstraint('numeroPosti > 0', name='c_nposti')
)

aeroporti = Table('aeroporti', metadata,
    Column('id', Integer, primary_key=True),
    Column('nome', String(30), nullable=False),
    Column('indirizzo', String(60), nullable=False)
)

voli = Table('voli', metadata,
    Column('id', Integer, primary_key=True),
    Column('aeroportoPartenza', Integer, ForeignKey('aeroporti.id', ondelete="CASCADE", onupdate="CASCADE"), nullable=False),
    Column('dataOraPartenza', DateTime, nullable=False),
    Column('aeroportoArrivo', Integer, ForeignKey('aeroporti.id', ondelete="CASCADE", onupdate="CASCADE"), nullable=False),
    Column('dataOraArrivo', DateTime, nullable=False),
    Column('aereo', Integer, ForeignKey('aerei.id', ondelete="CASCADE", nullable=False),
    Column('prezzo', Float, nullable=False),
    CheckConstraint('prezzo > 0', name='c_prezzo'),
    CheckConstraint('dataOraArrivo > dataOraPartenza', name='c_date')
)

bagagli = Table('bagagli', metadata,
    Column('prezzo', Float, primary_key=True),
    Column('descrizione', String(60), nullable=False)
)

prenotazioni = Table('prenotazioni', metadata,
    Column('id', Integer, nullable=False),
    Column('id_user', Integer, ForeignKey('users.id', ondelete="CASCADE", onupdate="CASCADE"), nullable=False),
    Column('id_volo', Integer, ForeignKey('voli.id', ondelete="CASCADE", onupdate="CASCADE"), nullable=False, primary_key=True),
    Column('prezzo_bagaglio', Float, ForeignKey('bagagli.prezzo', nullable=False),
    Column('numeroPosto', Integer, nullable=False, primary_key=True),
    Column('prezzotot', Float, nullable=False),
    Column('valutazione', Integer, nullable=True),
    Column('critiche', String(200), nullable=True)
)

#Mette un indice per la colonna id nella tabella prenotazioni
Index('idpren_index', prenotazioni.c.id)

#Mette un indice per la colonna aeroportoPartenza e aeroportoArrivo nella tabella voli
Index('aeroportiPartArr_index', voli.c.aeroportoPartenza, voli.c.aeroportoArrivo)

#Definizione trigger per l'aumento del prezzo quando un volo supera il 50% della capienza
aumento = DDL(
    "CREATE DEFINER='admin'@'localhost' TRIGGER `aumento` "
    "AFTER INSERT ON `prenotazioni` "
    "FOR EACH ROW "
    "BEGIN "
    "    IF (SELECT (pv.pren*100)/a.numeroPosti FROM voli v JOIN pren_volo pv ON (v.id = pv.id AND v.id = NEW.id_volo) JOIN aerei a on v.aereo = a.id) > 50 "
    "    THEN UPDATE voli v SET v.prezzo = v.prezzo + 0.5 WHERE NEW.id_volo = v.id; "
    "END IF; "
    "END"
)

event.listen(
    prenotazioni,
    'after_create',
    aumento.execute_if(dialect='mysql')
)

#Definizione trigger che incrementa l'ora di partenza di un volo quando viene inserito con una data di partenza uguale ad un altro volo per lo stesso aeroporto
controllo_voli = DDL(
    "CREATE DEFINER='admin'@'localhost' TRIGGER controllo_voli "
    "BEFORE INSERT ON voli FOR EACH ROW "
    "BEGIN "
    "    IF (SELECT COUNT(*) FROM voli v1 WHERE v1.id != NEW.id AND v1.aeroportoPartenza = NEW.aeroportoPartenza AND v1.dataOraPartenza = NEW.dataOraPartenza) >= 1 THEN "
    "    SET NEW.dataOraPartenza = DATE_ADD(NEW.dataOraPartenza, INTERVAL 1 HOUR), NEW.dataOraArrivo = DATE_ADD(NEW.dataOraArrivo, INTERVAL 1 HOUR); "
    "END IF; "
    "END"
)

event.listen(
    voli,
    'after_create',
    controllo_voli.execute_if(dialect='mysql')
)

#Crea tutti gli elementi appena definiti compresi vincoli e chiavi esterne
metadata.create_all(engine)

```

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 sarà utile successivamente per creare la lista dei posti disponibili usata per la mappa.

```
volos = conn.execute(
    "SELECT v.id, part.nome, v.dataOraPartenza, arr.nome, v.dataOraArrivo, v.prezzo, a.numeroPosti, a.numeroPosti-pv.pren as postdisp " +
    "FROM voli v JOIN aeroporti arr ON v.aeroportoArrivo = arr.id JOIN aeroporti part ON v.aeroportoPartenza = part.id JOIN aerei a ON v.aereo = a.id " +
    "JOIN pren_volo pv ON pv.id = v.id WHERE v.id = %s", volopart
).fetchone()

conn.execute("CREATE OR REPLACE VIEW pren_volo AS SELECT v.id, count(p.id) AS pren FROM voli v LEFT JOIN prenotazioni p ON v.id = p.id_volo GROUP BY v.id")
```

- 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.

```
voli = conn.execute(
    "SELECT p.id, p.id_volo, a1.nome, v.dataOraPartenza, a2.nome, v.dataOraArrivo, v.aereo, p.numeroPosto, b.descrizione, p.prezzotot, p.valutazione, p.critiche " +
    "FROM prenotazioni p JOIN voli v ON p.id_volo = v.id JOIN bagagli b ON p.prezzo_bagaglio=b.prezzo JOIN aeroporti a1 ON v.aeroportoPartenza=a1.id JOIN aeroporti a2 ON v.aeroportoArrivo=a2.id " +
    "WHERE p.id_user= %s ORDER BY v.dataOraPartenza ASC", current_user.id
).fetchall()
```

- 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 arrotondati a due cifre, calcolati su tutta la tabella prenotazioni;

```
guadagniTotali = conn.execute("SELECT ROUND(IFNULL(sum(prenotazioni.prezzotot),0),2) FROM prenotazioni").fetchone()
```

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

```
conn.execute(
    "CREATE OR REPLACE VIEW guadagniTratte AS SELECT a1.nome AS aeroportoP, a2.nome AS aeroportoA, IFNULL(SUM(p.prezzotot),0) AS guadagniTot " +
    "FROM aeroporti AS a1 JOIN voli ON a1.id = voli.aeroportoPartenza LEFT JOIN aeroporti AS a2 ON voli.aeroportoArrivo = a2.id LEFT JOIN prenotazioni AS p ON voli.id = p.id_volo GROUP BY a1.nome, a2.nome "
)
trattaGuadagniMax = conn.execute(
    "SELECT IFNULL(aeroportoP, 'None'), IFNULL(aeroportoA, 'None'), ROUND(IFNULL(guadagniTot,0),2) FROM guadagniTratte WHERE guadagniTot = (SELECT MAX(guadagniTot) FROM guadagniTratte)"
).fetchone()
```

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;

```
totPasMesePrec = conn.execute(
    "SELECT IFNULL(sum(pren), 0) FROM voli NATURAL JOIN pren_volo WHERE voli.dataOraPartenza BETWEEN (CURRENT_DATE() - INTERVAL 1 MONTH) AND CURRENT_DATE()"
).fetchone()
totPasMeseSucc = conn.execute(
    "SELECT IFNULL(sum(pren), 0) FROM voli NATURAL JOIN pren_volo WHERE voli.dataOraPartenza BETWEEN CURRENT_DATE() AND (CURRENT_DATE() + INTERVAL 1 MONTH)"
).fetchone()
```

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

```
infoAeroporti = conn.execute(
    "SELECT nomeA, partenze, arrivi " +
    "FROM (SELECT a.nome as nomeA, IFNULL(SUM(pren_volo.pren), 0) AS partenze " +
    "FROM aeroporti AS a LEFT JOIN voli ON a.id = voli.aeroportoPartenza LEFT JOIN pren_volo ON voli.id = pren_volo.id " +
    "GROUP BY a.nome) AS t1," +
    "(SELECT a.nome as nomeB, IFNULL(SUM(pren_volo.pren), 0) AS arrivi " +
    "FROM aeroporti AS a LEFT JOIN voli ON a.id = voli.aeroportoArrivo LEFT JOIN pren_volo ON voli.id = pren_volo.id " +
    "GROUP BY a.nome) AS t2 " +
    "WHERE nomeA = nomeB").fetchall()
```


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.

```
infoVoli = conn.execute(
    "SELECT a1.nome, a2.nome, aerei.nome, (pren_volo.pren/aerei.numeroPosti)*100 AS percentualeCarico, IFNULL(AVG(prenotazioni.valutazione),0) AS valutazioneMedia, voli.dataOraArrivo "+
    "FROM voli JOIN aeroporti AS a1 ON voli.aeroportoPartenza = a1.id "+
    "JOIN aeroporti AS a2 ON voli.aeroportoArrivo = a2.id "+
    "JOIN aerei ON voli.aereo = aerei.id JOIN pren_volo ON voli.id = pren_volo.id "+
    "LEFT JOIN prenotazioni ON voli.id = prenotazioni.id_volo "+
    "GROUP BY voli.id").fetchall()
```

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 rendering di moduli per lo sviluppo web Python. 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/#>)
In aggiunta grazie a questa libreria siamo stati in grado di inserire dei validatori personalizzati per i form 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.
- **Flask 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 che conterrà tutte le route specifiche per quella funzione,

utils.py se sono presenti delle funzioni ausiliarie utilizzate da alcune route e form.py che specifica 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 abbiano 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 nel caso in cui si acquistano due biglietti infatti vogliamo che questa operazione sia fatta sempre 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 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 la domanda è alta e quindi 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 sia la l'orario di partenza che quello di arrivo, 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 per la semplicità di testare le query, infatti all'interno di PhpMyAdmin 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 fatte, certamente c'è stata d'aiuto la documentazione fornita nel sito ufficiale di Flask e anche qualche video tutorial su come impostare il progetto con i form.

6. Come far partire TakeAFly

All'interno della cartella aeroporto aprire il file `__init__.py` e modificare le credenziali di accesso al database con i propri parametri, 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.