

# *Progetto Basi di Dati 2025*

***Tema:*** Piattaforma per la  
gestione di viaggi aerei

Gruppo SOTEKAM:  
AWAMBO lionel (882999)  
ONNA MERCY  
Gianmarco Gandini (898066)

## INDICI

INTRODUZIONE .....	3
1- STRUTTURA DEL DOCUMENTO .....	4
2. Funzionalità Principali .....	4
3 -PROGETTAZIONE DELLA BASI DI DATI .....	5
<i>Progettazione logica</i> .....	8
4- QUERRY PRINCIPALI .....	10
7. Contributo al Progetto .....	20
CONCLUSIONE .....	21

# INTRODUZIONE

Il progetto implementa un sistema di gestione voli aerei completo, che permette agli utenti di cercare, prenotare e gestire voli, mentre le compagnie aeree possono gestire i propri voli e le prenotazioni. Il sistema è stato sviluppato utilizzando tecnologie moderne e seguendo le best practices di sviluppo software visti durante il corso .

# **1- STRUTTURA DEL DOCUMENTO**

## **2. Funzionalità Principali**

### **2.1-Gestione Voli**

- Ricerca Avanzata: Implementazione di un sistema di ricerca multi-criterio che permette di filtrare voli per:
  - Aeroporto di partenza/arrivo
  - Data
  - Compagnia aerea
  - Classe (Economy, Business, First)
  - Prezzo
- Visualizzazione Dettagli: Presentazione dettagliata di ogni volo con:
  - Informazioni aeroportuali
  - Orari
  - Disponibilità posti
  - Prezzi per classe
  - Stato volo

### **2.2 Sistema di Prenotazione**

- Prenotazione Multi-classe : Supporto per tre classi di viaggio con:
  - Gestione dinamica dei posti
  - Calcolo automatico dei prezzi
  - Validazione disponibilità
- Gestione Biglietti : Sistema completo per:
  - Creazione biglietti
  - Modifica prenotazioni
  - Cancellazione
  - Storico prenotazioni

## 2.3- Gestione Compagnie

- Dashboard Compagnie: Interfaccia dedicata per:
  - Gestione voli
  - Monitoraggio prenotazioni
  - Statistiche
  - Gestione profilo compagnia

## 2.4 Sistema Utenti

- Autenticazione Multi-ruolo : Supporto per:
  - Utenti standard
  - Compagnie aeree
- Gestione Profili : Funzionalità per:
  - Registrazione
  - Modifica dati (non ancora implementato)
  - Recupero password (non ancora implementato)
  - Gestione prenotazioni

# 3 -PROGETTAZIONE DELLA BASI DI DATI

Per la progettazione concettuale della la basi di dati sono stati identificati come oggetti :

- **Utente**

.

**Attributi: Id\_Utente, Nome**

- **Passeggero**

**(specializzazione di Utente)**

.

**Attributi: Id\_Passeggero, Nome, Cognome**

- **Utente anonimo**

**(specializzazione di Utente)**

**Attributi: tipo\_utente**

• Ricerca

**Attributi: Id\_Ricerca, aeroport\_partenza, aeroport\_arrivo, data\_partenza, data\_arrivo**

• Biglietto

**Attributi: Tipo\_Biglietto, Posto\_Assegnato, Data\_Acquisto, Numeri\_posti\_Eco, Numeri\_posti\_BUS, Numeri\_posti\_First**

• Compagnia aerea

**Attributi: Id\_Compagnia, Nome**

• Aero

**Attributi: ID\_Aero, Modello, Numero\_posti**

• Volo

**Attributi: Id\_volo, Data\_Partenza, Data\_Arrivo**

• Tratta

**Attributi: Id\_Tratt, Durata\_Tratta, Statistica**

• Aeroporto

**Attributi: Id\_Aeroporto, Nome\_città, Nome\_Aeroporto**

• Sosta

**Attributi: Id\_Sosta, Durata\_Sosta**



## *Progettazione logica*

```
UTENTE(  
  Id_Utente STRING PRIMARY KEY,  
  Nome STRING  
)  
PASSEGGERO(  
  Id_Passeggero STRING PRIMARY KEY, -- FK verso UTENTE(Id_Utente)  
  Nome STRING,  
  Cognome STRING,  
  FOREIGN KEY (Id_Passeggero) REFERENCES UTENTE(Id_Utente)  
)  
UTENTE_ANONIMO(  
  Id_Utente STRING PRIMARY KEY, -- FK verso UTENTE(Id_Utente)  
  tipo_utente STRING,  
  FOREIGN KEY (Id_Utente) REFERENCES UTENTE(Id_Utente)  
)  
RICERCA(  
  Id_Ricerca STRING PRIMARY KEY,  
  aeroport_partenza STRING,  
  aeroport_arrivo STRING,  
  data_partenza DATE,  
  data_arrivo DATE,  
  Id_Utente STRING,  
  FOREIGN KEY (Id_Utente) REFERENCES UTENTE(Id_Utente))  
COMPAGNIA_AEREA(  
  Id_Compagnia STRING PRIMARY KEY,  
  Nome STRING  
)  
AEREO(  
  Id_Aero STRING PRIMARY KEY,  
  Modello STRING,  
  Numero_posti INT,  
  Id_Compagnia STRING,  
  FOREIGN KEY (Id_Compagnia) REFERENCES COMPAGNIA_AEREA(Id_Compagnia)  
)  
AEROPORTO(  
  Id_Aeroporto STRING PRIMARY KEY,  
  Nome_città STRING,  
  Nome_Aeroporto STRING  
)  
TRATTA(  

```



```

    Id_Tratt STRING PRIMARY KEY,
    Durata_Tratta INT,
    Statistica STRING
)
COLLEGA(
    Id_Tratt STRING,
    Partenza_A STRING,
    Arrivo_A STRING,
    PRIMARY KEY (Id_Tratt, Partenza_A, Arrivo_A),
    FOREIGN KEY (Id_Tratt) REFERENCES TRATTA(Id_Tratt),
    FOREIGN KEY (Partenza_A) REFERENCES AEROPORTO(Id_Aeroporto),
    FOREIGN KEY (Arrivo_A) REFERENCES AEROPORTO(Id_Aeroporto)
)
SOSTA(
    Id_Sosta STRING PRIMARY KEY,
    Durata_Sosta STRING
)
VOLO(
    Id_Volo STRING PRIMARY KEY,
    Data_Partenza DATE,
    Data_Arrivo DATE,
    Id_Compagnia STRING,
    Id_Aero STRING,
    Id_Tratt STRING,
    Id_Sosta STRING, -- può essere NULL
    FOREIGN KEY (Id_Compagnia) REFERENCES COMPAGNIA_AEREA(Id_Compagnia),
    FOREIGN KEY (Id_Aero) REFERENCES AEREO(Id_Aero),
    FOREIGN KEY (Id_Tratt) REFERENCES TRATTA(Id_Tratt),
    FOREIGN KEY (Id_Sosta) REFERENCES SOSTA(Id_Sosta)
)
BIGLIETTO(
    Id_Biglietto STRING PRIMARY KEY,
    Tipo_Biglietto STRING CHECK (Tipo_Biglietto IN ('ECO', 'BUS', 'FIRST')),
    Posto_Assegnato STRING,
    Data_Acquisto DATE,
    Numeri_posti_Eco INT,
    Numeri_posti_BUS INT,
    Numeri_posti_First INT,
    Id_Volo STRING,
    Id_Passeggero STRING,
    FOREIGN KEY (Id_Volo) REFERENCES VOLO(Id_Volo),
    FOREIGN KEY (Id_Passeggero) REFERENCES PASSEGGERO(Id_Passeggero)
)

```

## 4- QUERY PRINCIPALI

```
-- QUERY PRINCIPALI DELL'APPLICAZIONE
-- =====

-- 1. Ricerca voli diretti
-- -----

-- Parametri: aeroporto_partenza, aeroporto_arrivo, data, passeggeri, classe
SELECT
    v.id,
    v.numero_volo,
    ca.nome_compagnia,
    a1.città AS città_partenza,
    a2.città AS città_arrivo,
    v.data_partenza,
    v.data_arrivo,
```

```

CASE
    WHEN 'economy' = 'economy' THEN v.posti_economy
    WHEN 'economy' = 'business' THEN v.posti_business
    WHEN 'economy' = 'first' THEN v.posti_first
END AS posti_disponibili,
CASE
    WHEN 'economy' = 'economy' THEN v.prezzo_economy
    WHEN 'economy' = 'business' THEN v.prezzo_business
    WHEN 'economy' = 'first' THEN v.prezzo_first
END AS prezzo
FROM volo v
JOIN compagnia_aerea ca ON v.airline_id = ca.id
JOIN aeroporto a1 ON v.departure_airport_id = a1.id
JOIN aeroporto a2 ON v.arrival_airport_id = a2.id
WHERE a1.codice_iata = 'FCO' -- parametro aeroporto_partenza
AND a2.codice_iata = 'LHR' -- parametro aeroporto_arrivo
AND DATE(v.data_partenza) = '2024-04-15' -- parametro data
AND (
    CASE
        WHEN 'economy' = 'economy' THEN v.posti_economy
        WHEN 'economy' = 'business' THEN v.posti_business
        WHEN 'economy' = 'first' THEN v.posti_first
    END
    ) >= 1 -- parametro passeggeri
ORDER BY
CASE
    WHEN 'prezzo' = 'prezzo' THEN
        CASE
            WHEN 'economy' = 'economy' THEN v.prezzo_economy
            WHEN 'economy' = 'business' THEN v.prezzo_business
            WHEN 'economy' = 'first' THEN v.prezzo_first
        END
    WHEN 'prezzo' = 'tempo' THEN
        EXTRACT(EPOCH FROM (v.data_arrivo - v.data_partenza))
END;

```

```

-- 2. Ricerca voli con scalo
-- -----
WITH voli_diretti AS (
    SELECT
        v.id,
        v.numero_volo,
        ca.nome_compagnia,
        a1.città AS città_partenza,

```

```

        a2.città AS città_arrivo,
        v.data_partenza,
        v.data_arrivo,
        v.prezzo_economy,
        v.prezzo_business,
        v.prezzo_first
    FROM volo v
    JOIN compagnia_aerea ca ON v.airline_id = ca.id
    JOIN aeroporto a1 ON v.departure_airport_id = a1.id
    JOIN aeroporto a2 ON v.arrival_airport_id = a2.id
)
SELECT
    v1.id AS volo1_id,
    v1.numero_volo AS volo1_numero,
    v1.nome_compagnia AS compagnia1,
    v1.città_partenza,
    v1.città_arrivo AS città_scalo,
    v2.id AS volo2_id,
    v2.numero_volo AS volo2_numero,
    v2.nome_compagnia AS compagnia2,
    v2.città_arrivo AS città_arrivo,
    v1.data_partenza,
    v1.data_arrivo AS scalo_arrivo,
    v2.data_partenza AS scalo_partenza,
    v2.data_arrivo,
    v1.prezzo_economy + v2.prezzo_economy AS prezzo_totale_economy,
    v1.prezzo_business + v2.prezzo_business AS prezzo_totale_business,
    v1.prezzo_first + v2.prezzo_first AS prezzo_totale_first
FROM voli_diretti v1
JOIN voli_diretti v2 ON v1.città_arrivo = v2.città_partenza
WHERE v1.città_partenza = 'Roma' -- parametro città_partenza
    AND v2.città_arrivo = 'Madrid' -- parametro città_arrivo
    AND DATE(v1.data_partenza) = '2024-04-15' -- parametro data
    AND v2.data_partenza - v1.data_arrivo >= INTERVAL '2 hours' -- minimo 2 ore di scalo
ORDER BY
    CASE
        WHEN 'prezzo' = 'prezzo' THEN v1.prezzo_economy + v2.prezzo_economy
        WHEN 'prezzo' = 'tempo' THEN
            EXTRACT(EPOCH FROM (v2.data_arrivo - v1.data_partenza))
    END;

```

```

-- 3. Statistiche compagnia aerea
-- -----
SELECT

```

```

    ca.nome_compagnia,
    COUNT(DISTINCT v.id) AS numero_voli,
    COUNT(b.id) AS numero_passeggeri,
    SUM(b.prezzo) AS guadagno_totale,
    AVG(b.prezzo) AS prezzo_medio,
    COUNT(DISTINCT CASE WHEN b.classe = 'economy' THEN b.id END) AS passeggeri_economy,
    COUNT(DISTINCT CASE WHEN b.classe = 'business' THEN b.id END) AS passeggeri_business,
    COUNT(DISTINCT CASE WHEN b.classe = 'first' THEN b.id END) AS passeggeri_first
FROM compagnia_aerea ca
LEFT JOIN volo v ON ca.id = v.airline_id
LEFT JOIN biglietto b ON v.id = b.flight_id
WHERE ca.id = 1 -- parametro compagnia_id
      AND v.data_partenza >= '2024-04-01' -- parametro data_inizio
      AND v.data_partenza <= '2024-04-30' -- parametro data_fine
GROUP BY ca.id, ca.nome_compagnia;

```

```

-- 4. Prenotazioni utente
-- -----
SELECT
    p.id AS prenotazione_id,
    p.data_prenotazione,
    p.stato,
    p.prezzo_totale,
    json_agg(json_build_object(
        'volo', v.numero_volo,
        'compagnia', ca.nome_compagnia,
        'partenza', a1.città,
        'arrivo', a2.città,
        'data_partenza', v.data_partenza,
        'data_arrivo', v.data_arrivo,
        'classe', b.classe,
        'posto', b.numero_posto,
        'prezzo', b.prezzo,
        'bagaglio_extra', b.bagaglio_extra,
        'servizi_extra', b.servizi_extra
    )) AS dettagli_biglietti
FROM prenotazione p
JOIN biglietto b ON p.id = b.booking_id
JOIN volo v ON b.flight_id = v.id
JOIN compagnia_aerea ca ON v.airline_id = ca.id
JOIN aeroporto a1 ON v.departure_airport_id = a1.id
JOIN aeroporto a2 ON v.arrival_airport_id = a2.id
WHERE p.user_id = 4 -- parametro user_id
GROUP BY p.id, p.data_prenotazione, p.stato, p.prezzo_totale

```

```
ORDER BY p.data_prenotazione DESC;
```

```
-- 5. Verifica disponibilità posti
-- -----
SELECT
    v.id,
    v.numero_volo,
    v.posti_economy - COUNT(CASE WHEN b.classe = 'economy' THEN 1 END) AS
posti_economy_disponibili,
    v.posti_business - COUNT(CASE WHEN b.classe = 'business' THEN 1 END) AS
posti_business_disponibili,
    v.posti_first - COUNT(CASE WHEN b.classe = 'first' THEN 1 END) AS posti_first_disponibili
FROM volo v
LEFT JOIN biglietto b ON v.id = b.flight_id
WHERE v.id = 1 -- parametro volo_id
GROUP BY v.id, v.numero_volo, v.posti_economy, v.posti_business, v.posti_first;
```

```
-- QUERY PRINCIPALI PER IL SISTEMA DI GESTIONE VOLI AEREI
-- Questo file contiene le query SQL più importanti utilizzate nel sistema
```

```
-- 1. RICERCA VOLI
-- Questa query permette di cercare i voli disponibili con vari filtri:
-- - Città di partenza e arrivo
-- - Data del volo
-- - Compagnia aerea (opzionale)
-- Restituisce tutti i dettagli del volo inclusi prezzi e posti disponibili
SELECT
    f.id,
    f.numero_volo,
    c.nome as compagnia,
    a1.citta as partenza,
    a2.citta as arrivo,
    f.data_partenza,
    f.data_arrivo,
    f.posti_economy_disponibili,
    f.posti_business_disponibili,
    f.posti_first_disponibili,
    f.prezzo_economy,
    f.prezzo_business,
    f.prezzo_first,
    f.stato
FROM voli f
JOIN compagnie c ON f.compagnia_id = c.id
JOIN aeroporti a1 ON f.aeroporto_partenza_id = a1.id
```

```

JOIN aeroporti a2 ON f.aeroporto_arrivo_id = a2.id
WHERE
    a1.citta = :citta_partenza
    AND a2.citta = :citta_arrivo
    AND f.data_partenza >= :data_partenza
    AND f.data_partenza < :data_partenza + INTERVAL '1 day'
    AND (:compagnia_id IS NULL OR f.compagnia_id = :compagnia_id)
    AND f.stato = 'CONFERMATO'
ORDER BY f.data_partenza;

```

```

-- 2. VERIFICA DISPONIBILITÀ POSTI
-- Controlla quanti posti sono disponibili per una specifica classe (Economy, Business, First)
-- Utile durante il processo di prenotazione per verificare immediatamente la disponibilità
SELECT
    f.id,
    f.numero_volo,
    CASE
        WHEN :classe = 'ECONOMY' THEN f.posti_economy_disponibili
        WHEN :classe = 'BUSINESS' THEN f.posti_business_disponibili
        WHEN :classe = 'FIRST' THEN f.posti_first_disponibili
    END as posti_disponibili
FROM voli f
WHERE f.id = :volo_id;

```

```

-- 3. STATISTICHE COMPAGNIE AEREE
-- Fornisce un riepilogo completo delle performance di ogni compagnia aerea:
-- - Numero totale di voli operati
-- - Numero totale di posti venduti
-- - Ricavi totali generati
-- Ordinato per ricavi in ordine decrescente
SELECT
    c.nome as compagnia,
    COUNT(f.id) as totale_voli,
    SUM(
        (f.posti_economy_totali - f.posti_economy_disponibili) +
        (f.posti_business_totali - f.posti_business_disponibili) +
        (f.posti_first_totali - f.posti_first_disponibili)
    ) as posti_venduti,
    SUM(
        (f.posti_economy_totali - f.posti_economy_disponibili) * f.prezzo_economy +
        (f.posti_business_totali - f.posti_business_disponibili) * f.prezzo_business +
        (f.posti_first_totali - f.posti_first_disponibili) * f.prezzo_first
    ) as ricavi_totali
FROM compagnie c

```

```
LEFT JOIN voli f ON c.id = f.compagnia_id
GROUP BY c.id, c.nome
ORDER BY ricavi_totali DESC;
```

#### -- 4. PRENOTAZIONI UTENTE

-- Mostra la cronologia completa delle prenotazioni di un utente specifico  
-- Include dettagli del volo, compagnia, prezzi e stato della prenotazione

```
SELECT
    b.id as biglietto_id,
    f.numero_volo,
    c.nome as compagnia,
    a1.citta as partenza,
    a2.citta as arrivo,
    f.data_partenza,
    f.data_arrivo,
    b.classe,
    b.prezzo_pagato,
    b.stato
FROM biglietti b
JOIN voli f ON b.volo_id = f.id
JOIN compagnie c ON f.compagnia_id = c.id
JOIN aeroporti a1 ON f.aeroporto_partenza_id = a1.id
JOIN aeroporti a2 ON f.aeroporto_arrivo_id = a2.id
WHERE b.utente_id = :utente_id
ORDER BY f.data_partenza DESC;
```

#### -- 5. ROTTE PIÙ POPOLARI

-- Identifica le rotte più richieste basandosi sul numero di prenotazioni  
-- Include anche il prezzo medio per ogni rotta  
-- Limitato alle top 10 rotte più popolari

```
SELECT
    a1.citta as partenza,
    a2.citta as arrivo,
    COUNT(b.id) as numero_prenotazioni,
    AVG(
        CASE
            WHEN b.classe = 'ECONOMY' THEN f.prezzo_economy
            WHEN b.classe = 'BUSINESS' THEN f.prezzo_business
            WHEN b.classe = 'FIRST' THEN f.prezzo_first
        END
    ) as prezzo_medio
FROM biglietti b
JOIN voli f ON b.volo_id = f.id
JOIN aeroporti a1 ON f.aeroporto_partenza_id = a1.id
```



```

JOIN aeroporti a2 ON f.aeroporto_arrivo_id = a2.id
WHERE b.stato = 'CONFERMATO'
GROUP BY a1.citta, a2.citta
ORDER BY numero_prenotazioni DESC
LIMIT 10;

```

```

-- 6. ANALISI OCCUPAZIONE VOLI
-- Calcola la percentuale di occupazione per ogni volo
-- Utile per analisi di performance e ottimizzazione delle rotte
SELECT
    f.numero_volo,
    f.data_partenza,
    c.nome as compagnia,
    a1.citta as partenza,
    a2.citta as arrivo,
    ROUND(
        (
            (f.posti_economy_totali - f.posti_economy_disponibili) +
            (f.posti_business_totali - f.posti_business_disponibili) +
            (f.posti_first_totali - f.posti_first_disponibili)
        )::numeric /
        (
            f.posti_economy_totali +
            f.posti_business_totali +
            f.posti_first_totali
        )::numeric * 100,
        2
    ) as percentuale_occupazione
FROM voli f
JOIN compagnie c ON f.compagnia_id = c.id
JOIN aeroporti a1 ON f.aeroporto_partenza_id = a1.id
JOIN aeroporti a2 ON f.aeroporto_arrivo_id = a2.id
WHERE f.data_partenza >= CURRENT_DATE
ORDER BY f.data_partenza;

```

```

-- 7. RICERCA AEROPORTI
-- Permette di cercare aeroporti per nome città o codice IATA
-- Mostra anche il numero di voli in partenza e arrivo per ogni aeroporto
SELECT DISTINCT
    a.id,
    a.citta,
    a.nazione,
    a.codice_iata,
    COUNT(f.id) as numero_voli_in_partenza,

```

```

        COUNT(f2.id) as numero_voli_in_arrivo
FROM aeroporti a
LEFT JOIN voli f ON a.id = f.aeroporto_partenza_id
LEFT JOIN voli f2 ON a.id = f2.aeroporto_arrivo_id
WHERE
    a.citta ILIKE :cerca_citta || '%'
    OR a.codice_iata ILIKE :cerca_codice || '%'
GROUP BY a.id, a.citta, a.nazione, a.codice_iata
ORDER BY a.citta;

```

```

-- 8. ANALISI CLIENTI
-- Fornisce statistiche dettagliate sui clienti:
-- - Numero totale di prenotazioni per utente
-- - Spesa totale
-- - Data dell'ultimo volo
-- Ordinato per spesa totale in ordine decrescente
SELECT
    u.email,
    COUNT(b.id) as totale_prenotazioni,
    SUM(
        CASE
            WHEN b.classe = 'ECONOMY' THEN f.prezzo_economy
            WHEN b.classe = 'BUSINESS' THEN f.prezzo_business
            WHEN b.classe = 'FIRST' THEN f.prezzo_first
        END
    ) as spesa_totale,
    MAX(f.data_partenza) as ultimo_volo
FROM utenti u
LEFT JOIN biglietti b ON u.id = b.utente_id
LEFT JOIN voli f ON b.volo_id = f.id
WHERE b.stato = 'CONFERMATO'
GROUP BY u.id, u.email
ORDER BY spesa_totale DESC;

```

## Database

### **\*\*Trigger Automatici\*\***

- Calcolo posti totali
- Validazione date voli
- Aggiornamento posti disponibili
- Logging modifiche

### **\*\*Vincoli di Integrità\*\*:**

- Chiavi esterne
- Controlli sui dati

## 6. Informazioni Tecniche

### 6.1 Stack Tecnologico

#### A- Backend:

- Flask (Python)
- SQLAlchemy (ORM)
- PostgreSQL (Database)
- Flask-Login (Autenticazione)
- Flask-WTF (Form)

#### B- Frontend:

- HTML5/CSS3
- JavaScript
- Bootstrap
- jQuery

#### - Sicurezza:

- Flask-Security

# 7. Contributo al Progetto

## 7.1 Design e Sviluppo

- **\*\*Design Database\*\***: [CEDRICK LIONE 882999] - Progettazione schema e relazioni
- **\*\*Backend\*\***: [ONNA MERCY] - Implementazione API e logica business
- **\*\*Frontend\*\***: [Gianmarco Gandini - cedrick lionel ] - Sviluppo interfaccia utente
- **\*\*Testing\*\***: [Gianmarco Gandini cedrick lionel ONNA MERCY] - Test e validazione

## 7.2 Documentazione

- **Documentazione Tecnica\*\***: [AWAMBO CEDRICK 882999] - Stesura documentazione
- **\*\*Manuale Utente\*\***: [AWAMBO CEDRICK 882999]
- **Guide e tutorial**
- **\*\*Commenti Codice\*\***: Tutti i membri del team

# CONCLUSIONE

Questa applicazione web permette la gestione di voli aerei, la ricerca di tratte e l'acquisto di biglietti. Il sistema è sviluppato utilizzando Flask e SQLAlchemy, con PostgreSQL come database , durante lo sviluppo di questa web application abbiamo incontrato tante difficoltà soprattutto al livello del frontend . il nostro progetto non è totalmente funzionale ovvero non si può commercializzare oppure metterlo online al profitto dell'uso pubblico ragione per la quale abbiamo deciso di elencare qualche funzionalità future per poter considerare la nostra web app operativa ; come funzionalità futura abbiamo : INTEGRAZIONE PAGAMENTO , APP MOBILE ,STATISTICHE AVANZATE GESTIONE BAGAGLI .

Il sistema offre una soluzione completa per la gestione di voli aerei, con particolare attenzione a:

- Usabilità
- Sicurezza
- Scalabilità
- Manutenibilità
- Performance

