Scelte progettuali del sistema informativo per la gestione dell’aeroporto di Napoli

Per lo sviluppo del sistema informativo dell’aeroporto di Napoli, abbiamo progettato un'architettura a più livelli, organizzata in pacchetti logici: model, controller, dao, ImplementazioneDAO, gui, db.

# 📦 Pacchetto model

Contiene le classi fondamentali per rappresentare il dominio applicativo:  
  
- Utente: classe base con credenziali comuni (login, password, email, id).  
- Amministratore e UtenteGenerico: sottoclassi di Utente, modellano due ruoli distinti.  
- Passeggero: rappresenta il soggetto fisico che viaggia, associato a una prenotazione.  
- Volo: contiene le informazioni relative a un singolo volo (origine, destinazione, data, orari, stato, ecc.).  
- Prenotazione: associa un passeggero a un volo, con dettagli su posto, classe, bagagli, stato.  
- Bagaglio: associato a una prenotazione, contiene il codice e lo stato.  
- VoloInArrivo e VoloInPartenza: sottoclassi specializzate di Volo.  
- Le classi StatoDelVolo, StatoPrenotazione, StatoDelBagaglio, ClasseVolo sono gestite come enumerazioni.

## 🔗 Relazioni significative

- Un Amministratore gestisce più voli (1..\*).  
- Un UtenteGenerico può avere molte prenotazioni (0..\*).  
- Ogni Prenotazione è associata a un Passeggero (1:1).  
- Ogni Prenotazione è associata a un Volo (1:1).  
- Una Prenotazione può avere più Bagagli (0..\*).  
- I Voli in partenza hanno un gate assegnato.

# 📦 Pacchetto controller

La classe Controller è il mediatore principale tra la GUI e il livello di accesso ai dati. Contiene 25 metodi pubblici, tra cui:  
  
- Login/logout  
- Ricerca voli  
- Prenotazione, modifica e check-in  
- Monitoraggio bagagli  
- Inserimento e aggiornamento voli  
- Assegnazione gate

# 📦 Pacchetto gui

Comprende tutte le interfacce grafiche, organizzate per funzione:  
  
- HomePageUtenteGenerico, HomePageAmministratore  
- PrenotaVolo, ModificaPrenotazione, CheckIn  
- InserisciVolo, AggiornaVolo, AssegnaGate  
- CercaPasseggero, CercaPrenotazione, MonitoraBagaglio  
- VisualizzaBagagliSmarriti, LoginPage, SignIn  
  
Ogni GUI è collegata al Controller e gestisce azioni utente tramite eventi ActionListener.

# 📦 Pacchetti dao e ImplementazioneDAO

Per separare la logica applicativa dall'accesso ai dati, è stato seguito il pattern DAO (Data Access Object).

## Interface DAO

Definiscono le operazioni CRUD (Create, Read, Update, Delete) per entità come:  
Volo, Prenotazione, UtenteGenerico, Amministratore, Passeggero, Bagaglio.

## Implementazioni

Nel pacchetto ImplementazioneDAO ogni interfaccia ha una classe concreta che implementa l'accesso al database tramite JDBC. Ogni classe implementativa apre e chiude autonomamente la connessione (Connection) per ogni operazione, seguendo un approccio modulare e sicuro.  
  
Esempi:  
- ImplementazioneVoloDAO gestisce la creazione e modifica dei voli.  
- ImplementazionePrenotazioneDAO consente check-in, recupero prenotazioni, aggiornamenti.  
- ImplementazioneBagaglioDAO consente tracciamento e gestione smarrimenti.

# 📦 Pacchetto db

Contiene la classe ConnessioneDatabase, centralizzata per gestire le credenziali di connessione JDBC. In questo progetto è stato scelto un approccio manuale con connessioni gestite nelle DAO, per semplicità e flessibilità.

# 👥 Team di sviluppo

- Alessandro Esposito  
- Stefano Luongo  
- Gianmarco Minei