Documentazione del class diagram per la prima consegna del progetto di Object Orientation.

Componenti del gruppo:

Paduano Alessio N86005142

Scafaro Antonio N86005219

Scalice Luigi N86005197

Sommario

Identificazione delle Classi

Dopo una prima lettura della traccia assegnata abbiamo individuato l'esistenza di 5 classi e 3 classi di enumerazione.

Di seguito verranno elencate le classi con una breve descrizione di cosa rappresentano e il modo in cui sono state individuate.

Classi:

- **Utente**: Astrazione di chiunque andrà ad utilizzare l'applicativo, senza specificarne il suo ruolo all'interno.
  Questa classe è stata individuata tramite la frase *"Il sistema può essere utilizzato da utenti autenticati"*
- **Volo**: Astrazione del concetto di volo all'interno di un sistema aeroportuale.
  Questa classe è stata individuata attraverso numerosi riferimenti a dei voli nel testo.
- **Prenotazione**: ciò che l'utente effettua quando deve effettuare un viaggio.
  Questa classe è stata individuata mediante la frase *"Gli utenti generici possono effettuare prenotazioni per i voli programmati."*
- **Bagaglio**: ciò che l'ipotetico utente-passeggero porta durante il volo.
  Questa classe ci è stata specificamente richiesta dalla traccia attraverso la frase *"Un'altra funzione importante è il monitoraggio dei bagagli"*
- **Gate**: I luoghi fisici da cui partono i voli.
  Questa classe ci è stata specificamente richiesta dalla traccia attraverso la frase *"Il sistema gestisce anche i gate di imbarco"*

Identificazione degli Attributi

In riferimento alle classi individuate prima, la traccia ci suggerisce delle loro caratteristiche, che noi abbiamo assegnato come attributi.

Di seguito, la lista di ogni attributo individuato per ogni classe:

- **Utente**: abbiamo scelto come attributi login, password. La traccia ci dice *"utenti autenticati tramite una login e una password"*, quindi ci è sembrato opportuno inserirli. Dopodiché abbiamo individuato anche nome, cognome e numero_Documento, data una frase riferita alle prenotazioni *"Ogni prenotazione è legata a un volo e contiene informazioni come i dati del passeggero"*, che ci ha fatto ipotizzare la presenza di questi attributi, ai fini del miglioramento del sistema.
- **Volo**: codice_Univoco, compagnia_Aerea, aeroporto_Partenza, aeroporto_Arrivo, data_Partenza (che abbiamo suddiviso in giorno, mese ed anno per semplicità) stato_Volo (che indica lo stato del volo, attraverso una classe di enumerazione: esso può essere Programmato, Decollato, In Ritardo, Atterrato o Cancellato) e ritardo (che indica il ritardo in tempo). Tutti questi attributi sono stati specificati nella traccia, noi non abbiamo giunto nulla in più.
- **Prenotazione**: per la prenotazione abbiamo aggiunto nome_Passeggero, cognome_Passeggero e numero_Documento_Passeggero (rimanendo coerenti con la frase specificata prima per Utente, abbiamo specificato che fossero quelli i dati essenziali, riportandoli anche qui). Dopodiché abbiamo aggiunto soltanto attributi specificati nella traccia, ovvero stato_Prenotazione (tramite una classe di enumerazione: la prenotazione può essere Confermata, in attesa o cancellata), numero_Biglietto (essenziale ai fini del sistema) e posto_Assegnato.
- **Bagaglio**: la traccia ci ha specificato le uniche informazioni essenziali per un bagaglio, ovvero codice_Bagaglio (un codice univoco identificativo) e stato_Bagaglio (tramite una classe di enumerazione, con stati possibili: Ritiro_Disponibile, Caricato_Aereo)
- **Gate**: un solo attributo, ovvero numero_Gate, specificato dalla traccia.

Identificazione delle Generalizzazioni

Mediante una seconda lettura della traccia, è saltata all'occhio la presenza di ulteriori classi derivanti da classi già esistenti, in modo da poter dare una visione ed una gestione più completa del sistema, aggiungendo degli attributi più specifici per ognuna delle sottoclassi.

Di seguito elencate le generalizzazioni individuate, con allegata spiegazione:

- **Utente**: classe divisa in Utente_Generico ed Amministratore.
  Utente_Generico: è colui che si iscrive alla piattaforma e vuole prenotare dei voli, gestendo le proprie prenotazioni.
  Gli attributi nome, cognome e numero_Documento sono stati spostati in questa sottoclasse.
  Amministratore: è un utente di alto rango all'interno del sistema, se vogliamo, un impiegato dell'aeroporto. Il testo definisce le sue mansioni di gestione, modifica ed aggiornamento delle funzioni aeroportuali.
  Non sono stati aggiunti ulteriori attributi.
  Frase di riferimento per entrambe le sotto-classi:
  "Gli utenti sono suddivisi in due ruoli: utenti generici, che possono prenotare voli, e amministratori del sistema, che gestiscono l'inserimento e l'aggiornamento dei voli."
- **Volo**: classe divisa in Volo_Partenza e Volo_Arrivo.
  La traccia ci ha specificato una differenza tra volo in partenza e volo in arrivo, quindi abbiamo creato questa generalizzazione.
  Volo_Partenza: si distingue da un generico volo dato che ad esso viene assegnato un gate d'imbarco ed è per lui specificato l'aeroporto di partenza, ovvero quello di Napoli.
  Quindi in questa sottoclasse è stato assegnato un valore di default all'attributo aeroporto_Partenza, cioè "Napoli".
  Abbiamo inoltre aggiunto l'attributo gate_Partenza.
  Volo_Arrivo: si distingue da un generico volo dato che l'aeroporto di partenza non è quello di Napoli, ma lo è quello di arrivo.
  Abbiamo quindi assegnato un valore di default all'attributo aeroporto_Arrivo, ovvero "Napoli".
  Frase di riferimento per entrambe le sotto-classi:
  "Il sistema gestisce i voli in arrivo e quelli in partenza"

Individuazione Associazioni

Leggendo la traccia abbiamo notato un legame tra alcune classi, segnandole così come delle associazioni.

Di seguito, la lista delle associazioni:

- **Amministratore** ↔ **Volo**: La traccia afferma *"amministratori del sistema, che gestiscono l'inserimento e l'aggiornamento dei voli."* Portandoci a creare l'associazione Gestione tra le due classi.
  Un **Amministratore** deve gestire uno o più **Voli**.
  Un **Volo** deve essere gestito da uno ed un solo **Amministratore**.
- **Utente_Generico** ↔ **Prenotazione**: La traccia afferma *"utenti generici, che possono prenotare voli"* portandoci quindi a creare l'associazione Richiesta tra le due classi.
  Un **Utente_Generico** deve richiedere una o più **Prenotazioni**.
  Una **Prenotazione** deve essere richiesta da uno ed un solo **Utente_Generico**.
- **Prenotazione** ↔ **Bagaglio**: La traccia afferma *"Ogni bagaglio è associato al volo del passeggero"* che potrebbe far pensare ad un'associazione tra Bagaglio e Volo; tuttavia, si parla del volo del passeggero, che è quindi un insieme di informazioni legate al passeggero: ossia la prenotazione. Abbiamo quindi creato l'associazione Assegnazione.
  Ad una **Prenotazione** possono essere assegnati zero o più **Bagagli**.
  Un **Bagaglio** deve essere assegnato ad una ed una sola **Prenotazione**.
- **Gate** ↔ **Volo_Partenza**: La traccia afferma *"Il sistema gestisce anche i gate di imbarco (identificati da un numero), assegnandoli ai voli in partenza."* Portandoci quindi a creare l'associazione Partenza.
  Da un **Gate** devono partire uno o più **Voli_Partenza**.
  Un **Volo_Partenza** deve partire da uno ed un solo **Gate**.

Identificazione Operazioni

Infine, la traccia ha indicato la presenza di alcune operazioni che ogni singola classe deve effettuare. Ogni operazione ha un nome abbastanza autoesplicativo per ciò che sarà la sua funzione, in caso esso non lo sia, è spiegata la sua eventuale funzione.

Di seguito, la lista delle operazioni per ogni singola classe:

- **Utente**: La frase *"Il sistema consente agli utenti di visualizzare aggiornamenti sui voli"* ci ha portati a inserire il metodo visualizza_Voli(), in quanto ogni utente, amministratore o meno, può visualizzare i tutti i voli.
- **Utente_Generico**: abbiamo inserito i metodi prenota_Volo() , tramite la frase *"che possono prenotare voli"*, cerca_PrenotazioneNome(nome_Passeggero), cerca_PrenotazioneVolo(codice_Volo), modifica_Prenotazione(Prenotazione), queste ultime tre operazioni sono state individuate tramite la frase *"Gli utenti possono cercare e modificare le proprie prenotazioni in base al nome del passeggero o al numero del volo"*. segnala_Smarrimento(codice_Bagaglio), individuato grazie alla frase *"Se un bagaglio viene smarrito, l'utente può segnalarlo direttamente nel sistema."* ed infine visualizza_Bagagli(codice_Volo), aggiunto tramite la frase *"Gli utenti possono visualizzare lo stato aggiornato del proprio bagaglio tramite l'interfaccia del sistema."*
- **Amministratore**: Abbiamo individuato numerose operazioni per questa classe, ossia inserisci_Volo(), aggiorna_Volo(codice_Volo), queste ultime due operazioni sono state individuate mediante la frase *"Gli amministratori del sistema hanno la possibilità di inserire nuovi voli e aggiornare le informazioni sui voli esistenti"*, assegna_Gate(numero_Gate, codice_Volo), individuata mediante la frase *"Gli amministratori possono modificare l'assegnazione dei gate ",* visualizza_Passeggeri(), visualizza_BagagliTutti(), queste ultime due operazioni sono state individuate mediante la frase *"Infine, il sistema permette di eseguire ricerche rapide per trovare voli, passeggeri e bagagli in base a diversi criteri."* (per visualizzare i voli c'è già il metodo ereditato dalla classe Utente) aggiorna_Bagaglio(codice_Bagaglio), visualizza_Smarrimenti(), queste due ultime operazioni sono state individuate tramite la frase *"lo stato del bagaglio viene gestito manualmente dagli amministratori, che aggiornano il sistema indicando se è stato caricato sull'aereo o è disponibile per il ritiro".*
- **Volo**: Non ha operazioni.
- **Volo_Partenza**: Non ha operazioni.
- **Volo_Arrivo**: Non ha operazioni.
- **Prenotazione**: Ha come unica operazione check_In(), individuata tramite la frase *"Ogni bagaglio è associato al volo del passeggero e viene registrato nel sistema durante l'operazione di check-in"* , questo metodo il cui nome non fa pensare subito ad un'operazione specifica, servirà a legare un bagaglio ad una prenotazione.
- **Bagaglio**: Non ha operazioni.
- **Gate**: Non ha operazioni.