Documentazione Tecnica

Progetto Gestione Aereoporto Napoli

Autori:

Andrea Marquez, Mirko Maiello, Rocco Molinari

Data:
25 maggio 2025

Indice

1 Introduzione

Questo documento descrive in dettaglio il progetto software per il Sistema di Gestione del complesso aeroportuale di Napoli, sviluppato con linguaggio Java utilizzando il framework Swing per l'interfaccia grafica. Il sistema è stato progettato seguendo il pattern architetturale MVC (Model-View-Controller), per garantire una chiara separazione tra la logica della gestione dei dati e la visualizzazione.

2 Descrizione Generale del Progetto

Il progetto si propone di fornire un sistema integrato per la gestione degli accessi e delle registrazioni degli utenti, dei voli (in partenza e in arrivo), dei bagagli ed eventuali segnalazioni associate. I protagonisti del progetto si suddividono in: utente generico e amministratore. L'interazione con il sistema avviene tramite una serie di interfacce grafiche (GUI), sviluppate nel package `gui`, che si interfacciano con la logica di controllo situata nel package `controller`.

3 Package `gui`

Il package `gui` contiene tutte le classi dedicate alla gestione dell'interfaccia grafica tramite Swing. Le classi principali sono:

3.1 Classe `DashBoardUser`

Descrizione:
Questa classe implementa l'interfaccia grafica dedicata all'utente generico. Presenta un pannello principale che mostra un messaggio di benvenuto personalizzato, un menu a tendina (combo box) con le azioni disponibili (es. visualizza voli, prenotazione, ricerca prenotazioni, segnalazioni smarrimenti) e un pulsante per il logout. L'interazione con la GUI viene inoltrata al `Controller` tramite eventi, mantenendo separata la logica di controllo dalla visualizzazione.

Attributi principali:

- `JPanel homePage`: pannello principale contenente l'intera dashboard utente.
- `JPanel welcomePanel, welcomeText`: pannelli dedicati al messaggio di benvenuto.
- `JButton logoutButton`: pulsante per effettuare il logout o chiudere l'applicazione.
- `JLabel welcomeTextLabel`: etichetta che mostra l'ID o username dell'utente loggato.
- `JComboBox comboBox1`: menu a tendina con le azioni selezionabili dall'utente.
- `JLabel outputLabel`: etichetta nascosta usata per mostrare messaggi o risultati dinamici.
- `String username`: username dell'utente per personalizzare l'interfaccia.

- `JDialog choiceDialog`: dialog modale per finestre secondarie, come conferme o scelte particolari.
- `Controller controller`: riferimento al controller per delegare la logica delle azioni utente.

Metodi principali:

- `DashBoardUser(String username, Controller controller)`: costruttore che inizializza i componenti grafici, imposta la lista delle azioni nella combo box e associa i listener agli eventi generati dall'utente.
- `actionPerformed(ActionEvent e)`: gestisce gli eventi legati al pulsante di logout.
- `itemStateChanged(ItemEvent e)`: intercetta la selezione nella combo box e passa la scelta al controller tramite il metodo `selectedItem(String, DashBoardUser)`.
- Getter e setter per i componenti principali, utili per interagire con l'interfaccia da altre classi o dal controller.

3.2 Classe `Login`

Descrizione:
La classe `Login` implementa la schermata di accesso all'applicazione. Permette all'utente di inserire username o email e password per effettuare il login, oppure di accedere alla schermata di registrazione per nuovi utenti. La classe si occupa di gestire l'interfaccia grafica (GUI) e di inoltrare le richieste al `Controller` secondo il pattern MVC (Model-View-Controller).

Attributi principali:

- `JTextField login`: campo di testo per l'inserimento dell'username o email.
- `JPasswordField passwordField1`: campo per l'inserimento della password con caratteri nascosti.
- `JButton INVIOButton`: pulsante per confermare il login.
- `JButton REGISTRATIButton`: pulsante per accedere alla schermata di registrazione.
- `JPanel mainPanel`: pannello principale che contiene tutta l'interfaccia della schermata di login.
- `JPanel loginPanel, textPanel`: pannelli secondari per organizzare la grafica.
- `Controller controller`: riferimento al controller che gestisce la logica di login e registrazione.

Metodi principali:

- `Login(Controller controller)`: costruttore che inizializza la GUI, associa i listener ai pulsanti e configura il frame.
- `actionPerformed(ActionEvent e)`: gestisce gli eventi dei pulsanti; invia i dati di login al controller o apre la schermata di registrazione.
- Metodi di supporto per la gestione della GUI e delle interazioni utente.

Note:

- La classe si appoggia al controller per l'effettiva verifica delle credenziali e la gestione del flusso applicativo.
- I campi di testo vengono letti e validati al momento della pressione del pulsante INVIO.
- L'apertura della schermata di registrazione è delegata al controller.

3.3 Classe `DashBoardAdmin`

Descrizione:
Questa classe implementa l'interfaccia grafica dedicata all'amministratore del sistema aeroportuale. Presenta un pannello principale con un messaggio di benvenuto personalizzato contenente l'ID dell'admin, un menu a tendina con le azioni amministrative disponibili (come visualizzazione voli, inserimento e aggiornamento voli, modifica gate, gestione bagagli e visualizzazione segnalazioni) e un pulsante per il logout che chiude l'applicazione. Come per la dashboard utente, le azioni vengono inoltrate al `Controller` per mantenere la separazione tra logica e interfaccia.

Attributi principali:

- `JPanel dashboardAdminPage`: pannello principale che contiene tutta la dashboard dell'amministratore.
- `JLabel adminLabel`: etichetta per mostrare informazioni sull'admin, come l'username.
- `JPanel homePage, welcomePanel, welcomeText`: pannelli per organizzare i contenuti principali e il messaggio di benvenuto.
- `JLabel welcomeTextLabel`: etichetta che mostra il messaggio di benvenuto personalizzato con l'ID dell'admin.
- `JButton LOGOUTUSER`: pulsante per effettuare il logout (attualmente chiude l'applicazione).
- `JComboBox comboBox1`: menu a tendina con le azioni selezionabili dall'amministratore.
- `JLabel outputLabel`: etichetta usata per mostrare messaggi di feedback o output dopo azioni amministrative.
- `JDialog choiceDialog`: dialog modale per finestre di conferma o scelte particolari.
- `String username`: username o ID dell'amministratore, usato per personalizzare l'interfaccia.
- `Controller controller`: riferimento al controller per delegare la logica delle azioni.

Metodi principali:

- `DashBoardAdmin(String username, Controller controller)`: costruttore che inizializza i componenti grafici, imposta la lista delle azioni nella combo box, associa i listener agli eventi generati dall'admin e mostra l'username in alto.
- `getDashboardAdminPage()`: restituisce il pannello principale della dashboard per l'inserimento nel frame.
- Getter e setter per i componenti GUI, utili per l'interazione con altre classi o il controller.

4 Package `controller`

Il package `controller` implementa la logica di gestione delle operazioni richieste dall'interfaccia, coordinando dunque l'interazione tra model e GUI.

4.1 Classe `Controller`

La classe `Controller` gestisce la logica applicativa tra il modello dati e le interfacce grafiche (GUI), implementando il pattern MVC (Model-View-Controller).

Attributi principali

- `Utente user`: riferimento all'utente attualmente autenticato.
- `Amministratore admin`: riferimento all'amministratore autenticato (se presente).
- `Utente_Generico utente`: riferimento all'utente generico autenticato.
- `JFrame framePrincipale`: frame principale dell'applicazione che contiene le diverse interfacce grafiche.

Funzionalità principali

- **Gestione autenticazione**: semplificata con un metodo di login che distingue tra utenti normali e amministratori basandosi su credenziali fittizie (username e password "admin").
- **Interazione con la GUI**: costruisce dinamicamente pannelli Swing per le dashboard degli utenti e degli amministratori, in base all'azione selezionata tramite menu a tendina.
- **Supporto per utenti normali**: permette visualizzazione voli, prenotazione, ricerca prenotazioni e segnalazione bagagli smarriti attraverso finestre di dialogo interattive.
- **Supporto per amministratori**: consente operazioni avanzate quali inserimento, aggiornamento voli, modifica gate, aggiornamento stato bagagli e visualizzazione segnalazioni di smarrimento.
- **Interfaccia di registrazione**: fornisce una finestra modale per la registrazione di nuovi utenti con controlli di validazione di base sui campi obbligatori.
- **Componenti GUI helper**: metodi di utilità per creare campi di testo e pulsanti con azioni associate, migliorando l'aspetto e l'usabilità delle interfacce.

Note implementative

- L'autenticazione e la registrazione sono simulate e non prevedono collegamenti a database o sistemi persistenti.
- Le azioni eseguite tramite i pulsanti mostrano semplici messaggi di conferma senza effettuare modifiche reali ai dati.

- Il design è orientato ad una rapida dimostrazione delle funzionalità, lasciando spazio a futuri miglioramenti come l'integrazione con un backend.

Metodi principali

- `login(String login, String password)`: identifica il tipo di utente e crea l'istanza corrispondente.
- `selectedItem(String item, DashBoardUser d)`: genera e mostra l'interfaccia per l'azione selezionata dall'utente normale.
- `selectedItem(String item, DashBoardAdmin d)`: genera e mostra l'interfaccia per l'azione selezionata dall'amministratore.
- `interfacciaRegistrazione(Login l)`: crea la finestra di registrazione per nuovi utenti.
- Metodi helper privati per la creazione di componenti Swing personalizzati (`creaCampo` e `creaBottoneConAzione`).

5 Conclusioni

La struttura attuale del progetto rappresenta ancora uno scheletro funzionale, ma è ben ottimizzata per garantire una futura espandibilità e manutenzione. Pur essendo in una fase iniziale di sviluppo, l'architettura supporta chiaramente la separazione dei ruoli, la gestione di dati aeroportuali e le operazioni sia per utenti che per amministratori. Questo approccio modulare e orientato al pattern MVC facilita l'integrazione di nuove funzionalità, migliorando la robustezza e la scalabilità del sistema nel tempo.