

Gruppo Aereo 5

Progetto Aereo



Riva Samuele - matr.829874

Gioele Mella - matr.812802

Stefan Negura - matr.830585

Sommario

Sommario	2
Specifiche dei requisiti	3
Scelte di programmazione	4
Diagramma di Gantt	6
Diagramma casi d'uso	8
Diagramma delle attività	9
Diagramma di sequenza	10
Modello di dominio	11
Diagramma dei Package	13
Diagramma dell'Architettura Software	14
Diagramma di stato	15
DATABASE: Schema ER	16
Design Pattern	17
Tutorial di utilizzo	18
Funzionalità mancanti	23
Guida installazione	24
Conclusioni	24

Specifiche dei requisiti

1. 1. Definizione delle specifiche dei requisiti.

Analizzando la traccia del progetto, abbiamo rilevato i seguenti requisiti:

REQUISITI FUNZIONALI

Il sistema permette di consultare l'orario dei voli e la disponibilità dei posti sui voli.

Il sistema permette di comprare biglietti aerei per lo stesso volo da parte dello stesso cliente.

Il sistema permette di effettuare una prenotazione per un determinato volo.

Il sistema consente agli utenti di registrarsi al programma “Fedeltà”.

Il sistema permette di annullare una prenotazione. Il sistema permette la modifica della data e dell'ora dei biglietti già acquistati.

REQUISITI NON FUNZIONALI

Il sistema consente di effettuare il pagamento dei biglietti aerei tramite carta di credito.

Il sistema notifica l'utente prima della scadenza di una prenotazione.

Il sistema annulla la prenotazione se il cliente non effettua il pagamento del biglietto prenotato entro la data di scadenza.

Per ogni biglietto i clienti “Fedeltà” accumulano dei punti che posso essere spesi comprando biglietti aerei.

I clienti “Fedeltà” possono chiedere l'estratto conto con i punti accumulati.

Il sistema mette a disposizione due tipi di promozioni: per tutti i clienti e per i clienti “Fedeltà”.

Scelte di programmazione

Per sviluppare il nostro progetto abbiamo deciso di adottare le seguenti scelte per quanto riguarda lo sviluppo.

TECNOLOGIE DI SVILUPPO

In fase di progettazione, una delle prime cose che abbiamo deciso è la tecnologia con cui sviluppare il progetto. Infatti, dopo aver scelto la traccia del progetto, abbiamo cominciato a pensare come svilupparlo a grandi linee. La scelta è ricaduta su sviluppare un'applicazione Java composta da un backend ed un frontend, nella quale l'interfaccia utente (sviluppata nel front-end) è fatta in html e jsp e il database alla quale il back-end si collega è un db SQL.

SPRING

Abbiamo deciso di utilizzare alcuni framework Spring per la gestione dell'applicazione. Soprattutto abbiamo utilizzato Spring Web che si occupa del layer di presentazione dei dati specifico per applicazioni Web.

Come per ogni framework Spring, anche per usare questo modulo non è necessario estendere o implementare nessuna interfaccia specifica del framework.



DBMS H2

Il database è gestito da H2 che è un DBMS open-source scritto in linguaggio Java con tecnologia in memory. Esso può essere incapsulato in applicazioni Java ed eseguito in modalità client-server.



INVIO EMAIL

Per l'invio delle email, abbiamo fatto uso dell'API JavaMail. Esso è un package della Sun Microsystems che fornisce classi necessarie alla gestione della posta elettronica in linguaggio Java. Supporta tutti i protocolli di posta (POP, SMTP, IMAP) ma non contiene componenti per l'interfaccia grafica.

STRUMENTI UTILIZZATI:

Per la progettazione e pianificazione abbiamo utilizzato i seguenti programmi:

- Microsoft Project: per sviluppare il diagramma di Gantt iniziale e finale
- IBM Rational Software Architect Designer: per la realizzazione dei vari diagrammi fatti in fase di progettazione

Diagramma di Gantt

Diagramma iniziale

	Modal attività	Nome attività	Durata	Inizio	Fine	Predecessori	Nomi risorse
1		Diagramma Gantt	1 g	mar 11/02/20	mar 11/02/20		Gioele Mella; Samulele Riva; Stefan Negura
2		Brainstorming	1 g	mar 11/02/20	mar 11/02/20		Gioele Mella; Stefan Negura; Samulele Riva
3		Diagramma Casi d'uso	3 g	mer 12/02/20	ven 14/02/20		Gioele Mella; Stefan Negura; Samulele Riva
4		Modello di Dominio	3 g	mer 12/02/20	ven 14/02/20		Gioele Mella; Stefan Negura; Samulele Riva
5		Diagramma dell'Architettura Software	3 g	mer 12/02/20	ven 14/02/20		Gioele Mella; Stefan Negura; Samulele Riva
6		Diagramma delle Classi	3 g	mer 12/02/20	ven 14/02/20		Gioele Mella; Stefan Negura; Samulele Riva
7		Diagramma di Sequenza	3 g	mer 12/02/20	ven 14/02/20		Gioele Mella
8		Diagramma di Stato	3 g	mer 12/02/20	ven 14/02/20		Stefan Negura
9		Diagramma delle Attività	3 g	mer 12/02/20	ven 14/02/20		Samulele Riva
10		Fine Analisi	0 g	lun 17/02/20	lun 17/02/20	3;4;5;6;7;8;9	Gioele Mella; Stefan Negura; Samulele Riva
11		Database	10 g	lun 17/02/20	ven 28/02/20	10	Gioele Mella
12		Back-end	10 g	lun 17/02/20	ven 28/02/20	10	Samulele Riva
13		Front-end	10 g	lun 17/02/20	ven 28/02/20	10	Stefan Negura
14		Test	8 g	lun 24/02/20	mer 04/03/20		Gioele Mella; Samulele Riva; Stefan Negura
15		Fine coding	0 g	gio 05/03/20	gio 05/03/20	11;12;13;14	
16		Fine Documentazione	5 g	lun 02/03/20	ven 06/03/20		Gioele Mella; Samulele Riva; Stefan Negura
17		Consegna Progetto	0 g	lun 09/03/20	lun 09/03/20	16;15	

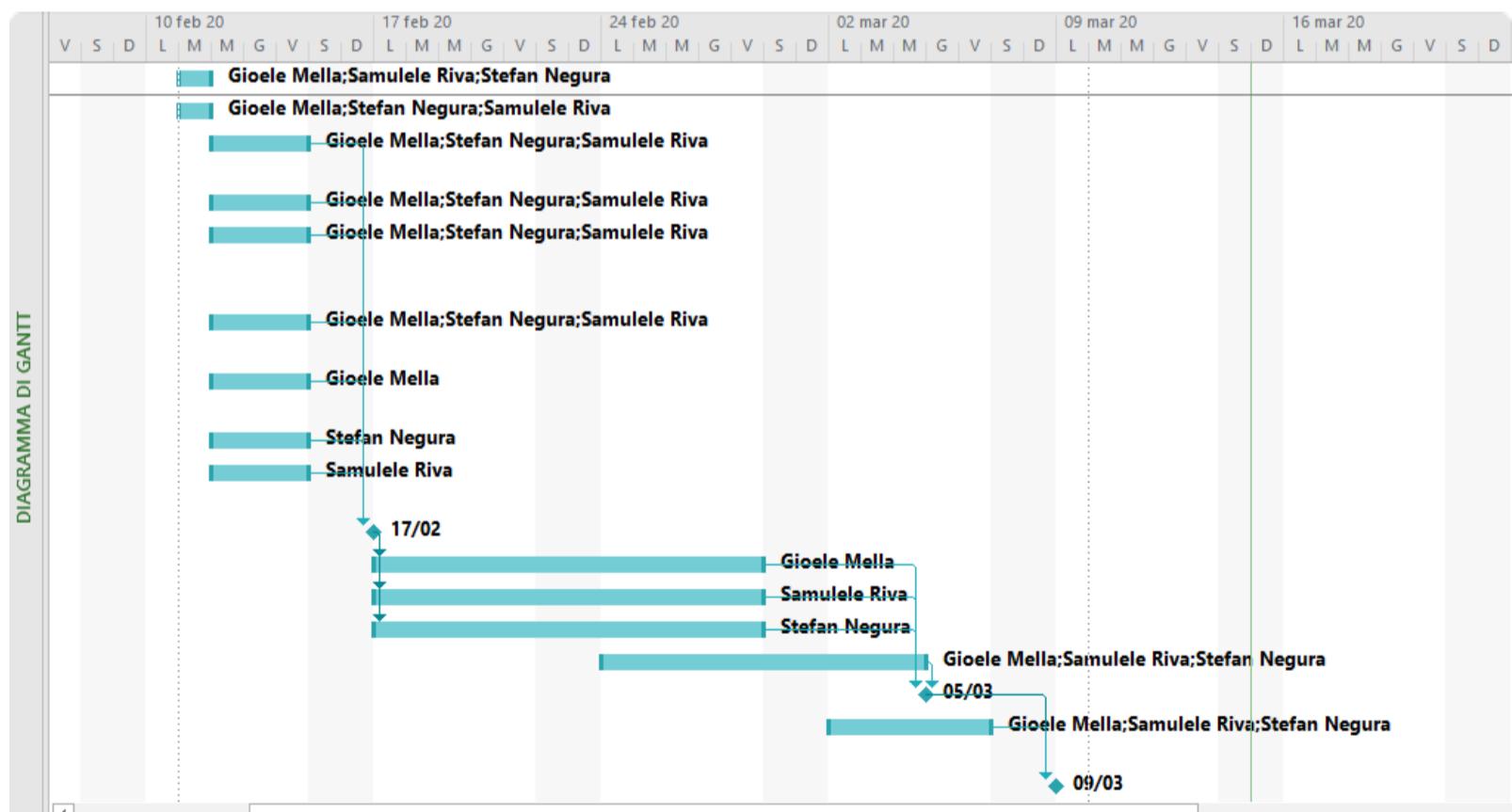
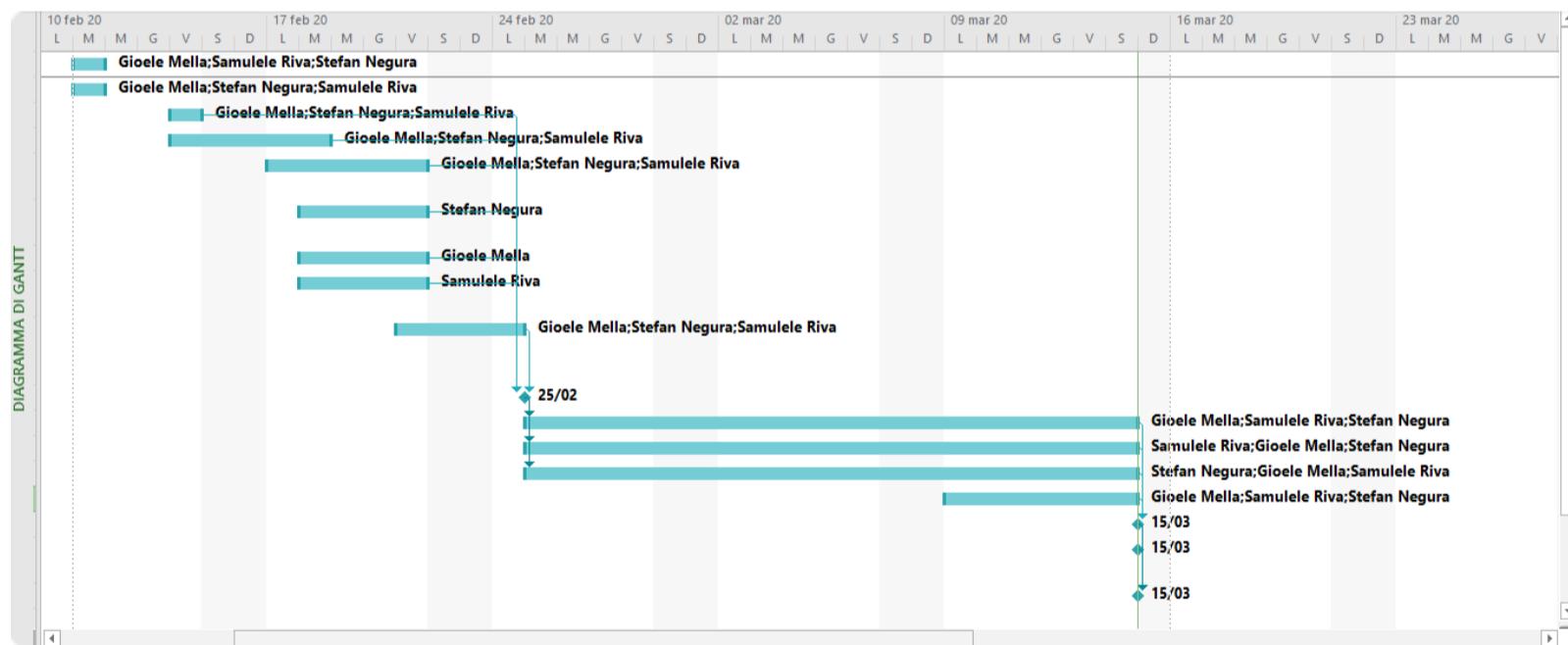


Diagramma finale

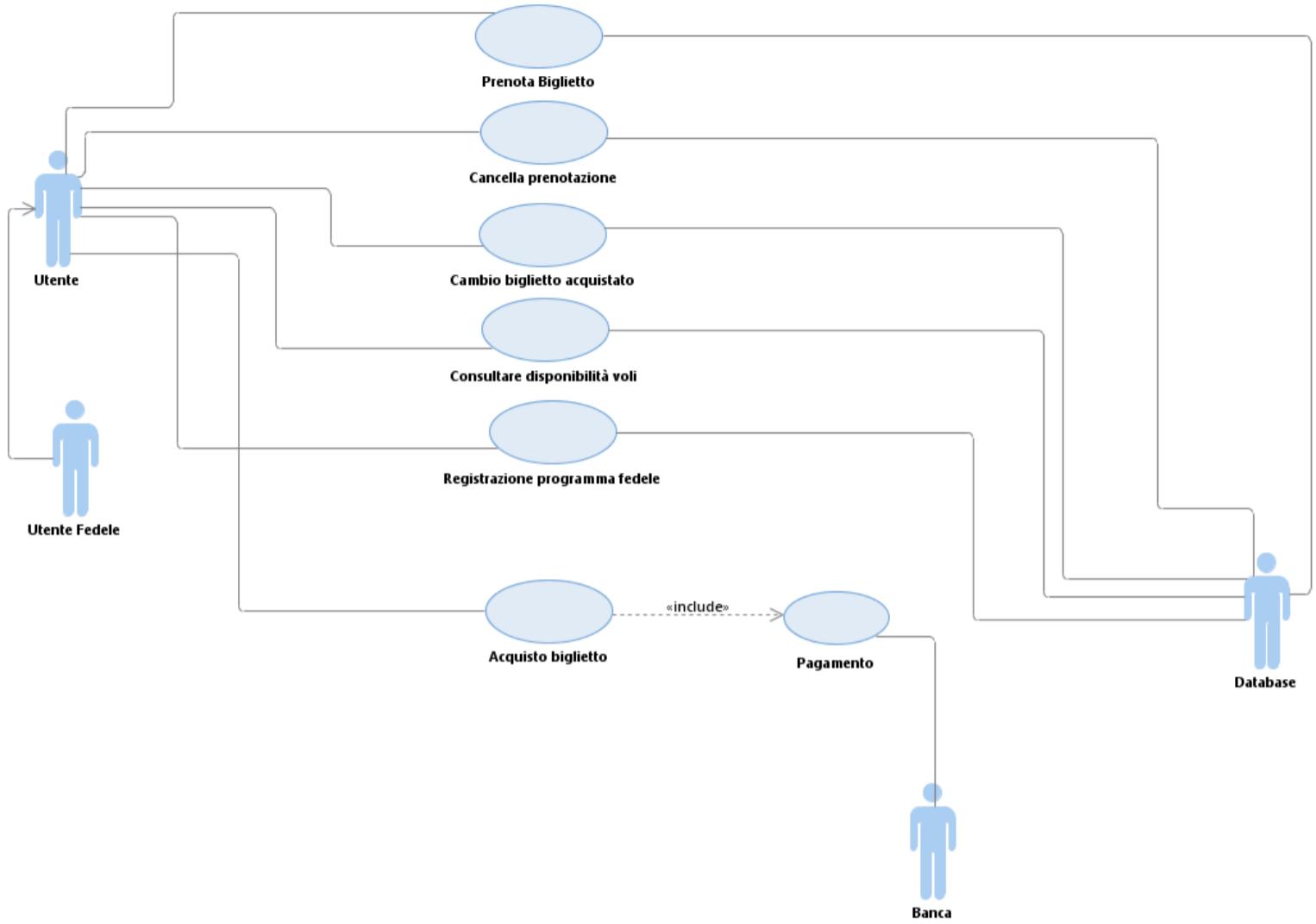
		Modal attività	Nome attività	Durata	Inizio	Fine	Predecessori	Nomi risorse
1		Diagramma Gantt	Diagramma Gantt	1 g	mar 11/02/20	mar 11/02/20		Gioele Mella; Samulele Riva; Stefan Negura
2		Brainstorming	Brainstorming	1 g	mar 11/02/20	mar 11/02/20		Gioele Mella; Stefan Negura; Samulele Riva
3		Diagramma Casi d'uso	Diagramma Casi d'uso	1 g	ven 14/02/20	ven 14/02/20		Gioele Mella; Stefan Negura; Samulele Riva
4		Modello di Dominio	Modello di Dominio	3 g	ven 14/02/20	mar 18/02/20		Gioele Mella; Stefan Negura; Samulele Riva
5		Diagramma delle Classi	Diagramma delle Classi	5 g	lun 17/02/20	ven 21/02/20		Gioele Mella; Stefan Negura; Samulele Riva
6		Diagramma di Sequenza	Diagramma di Sequenza	4 g	mar 18/02/20	ven 21/02/20		Stefan Negura
7		Diagramma di Stato	Diagramma di Stato	4 g	mar 18/02/20	ven 21/02/20		Gioele Mella
8		Diagramma delle Attività	Diagramma delle Attività	4 g	mar 18/02/20	ven 21/02/20		Samulele Riva
9		Diagramma dell'Architettura Software	Diagramma dell'Architettura Software	2 g	ven 21/02/20	lun 24/02/20		Gioele Mella; Stefan Negura; Samulele Riva
10		Fine Analisi	Fine Analisi	0 g	mar 25/02/20	mar 25/02/20	3;4;9;5;6;7;8	
11		Database	Database	15 g	mar 25/02/20	sab 14/03/20	10	Gioele Mella; Samulele Riva; Stefan Negura
12		Back-end	Back-end	15 g	mar 25/02/20	sab 14/03/20	10	Samulele Riva; Gioele Mella; Stefan Negura
13		Front-end	Front-end	15 g	mar 25/02/20	sab 14/03/20	10	Stefan Negura; Gioele Mella; Samulele Riva
14		Test	Test	6 g	lun 09/03/20	sab 14/03/20		Gioele Mella; Samulele Riva; Stefan Negura
15		Fine coding	Fine coding	0 g	dom 15/03/20	dom 15/03/20	11;12;13;14	
16		Fine Documentazione	Fine Documentazione	0 g	dom 15/03/20	dom 15/03/20		
17		Consegna Progetto	Consegna Progetto	0 g	dom 15/03/20	dom 15/03/20	16;15	



La nostra pianificazione ha avuto diverse complicanze a causa dei molti disagi che ha portato l'arrivo del Covid-19 in Italia e soprattutto in Lombardia.

La chiusura dell'università e i noti decreti fatti dal Governo per tutelare la salute dei cittadini non ci hanno, per gran parte del periodo a disposizione, permesso di incontrarci. Siamo quindi stati costretti a lavorare sempre da casa, comunicando solo attraverso le videochiamate. Questo, per un gruppo poco esperto come noi, ci ha creato diverse difficoltà e allungato le tempistiche.

Diagramma casi d'uso



Nel diagramma dei Casi d'Uso vengono rappresentate le interazioni degli attori che interagiscono con il sistema.

Nel nostro diagramma troviamo l'utente e l'utente registrato che interagiscono eseguendo le seguenti operazioni:

- Prenota biglietto
- Cancella prenotazione
- Cambia biglietto acquistato
- Consultare disponibilità voli
- Registrazione programma fedeltà
- Acquisto biglietto

Il sistema esterno Banca interagisce col sistema al momento del pagamento(evento simulato).

Diagramma delle attività

Nel diagramma delle attività riportato rappresentiamo i passi necessari per procedere al pagamento dei biglietti.

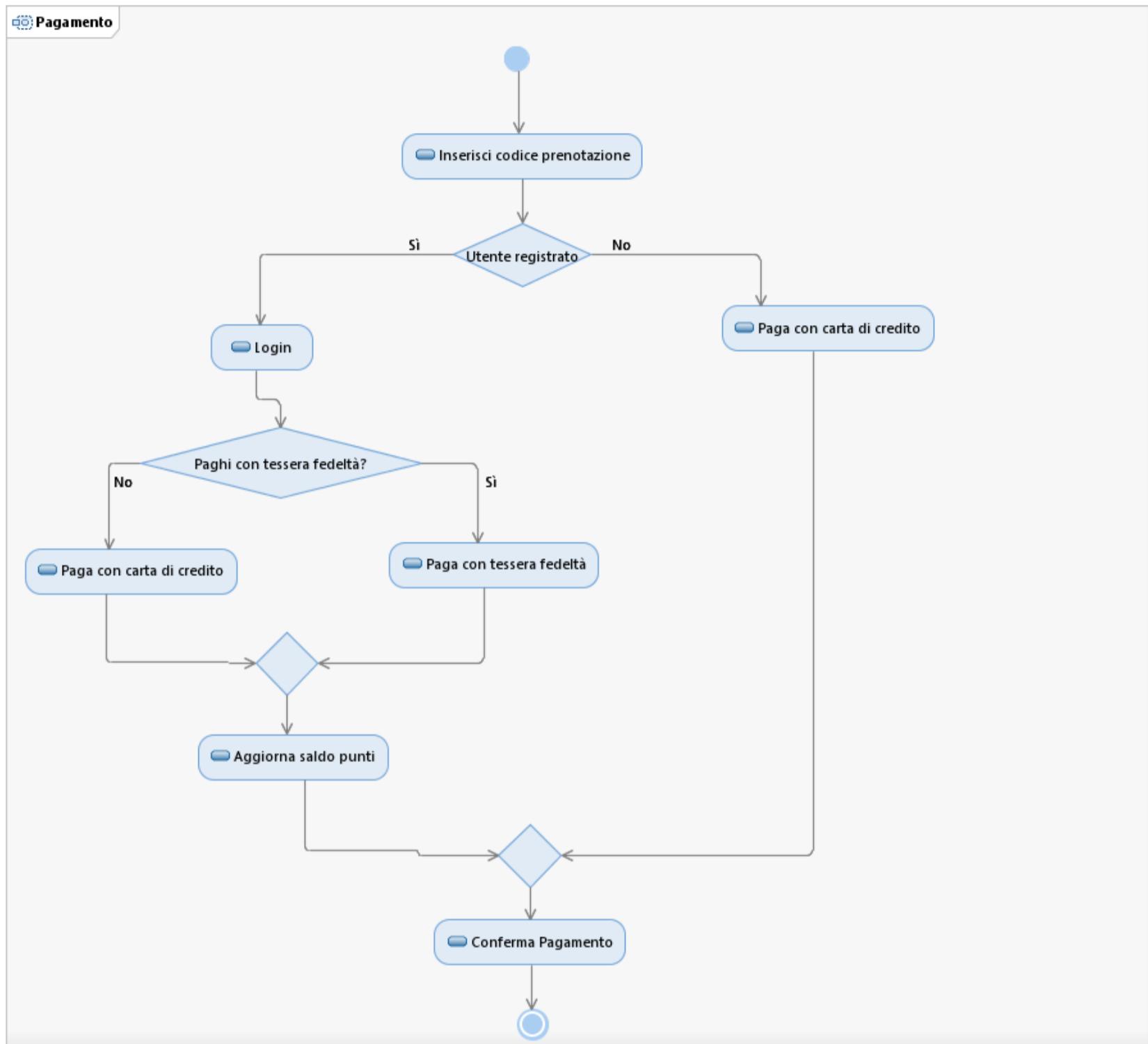
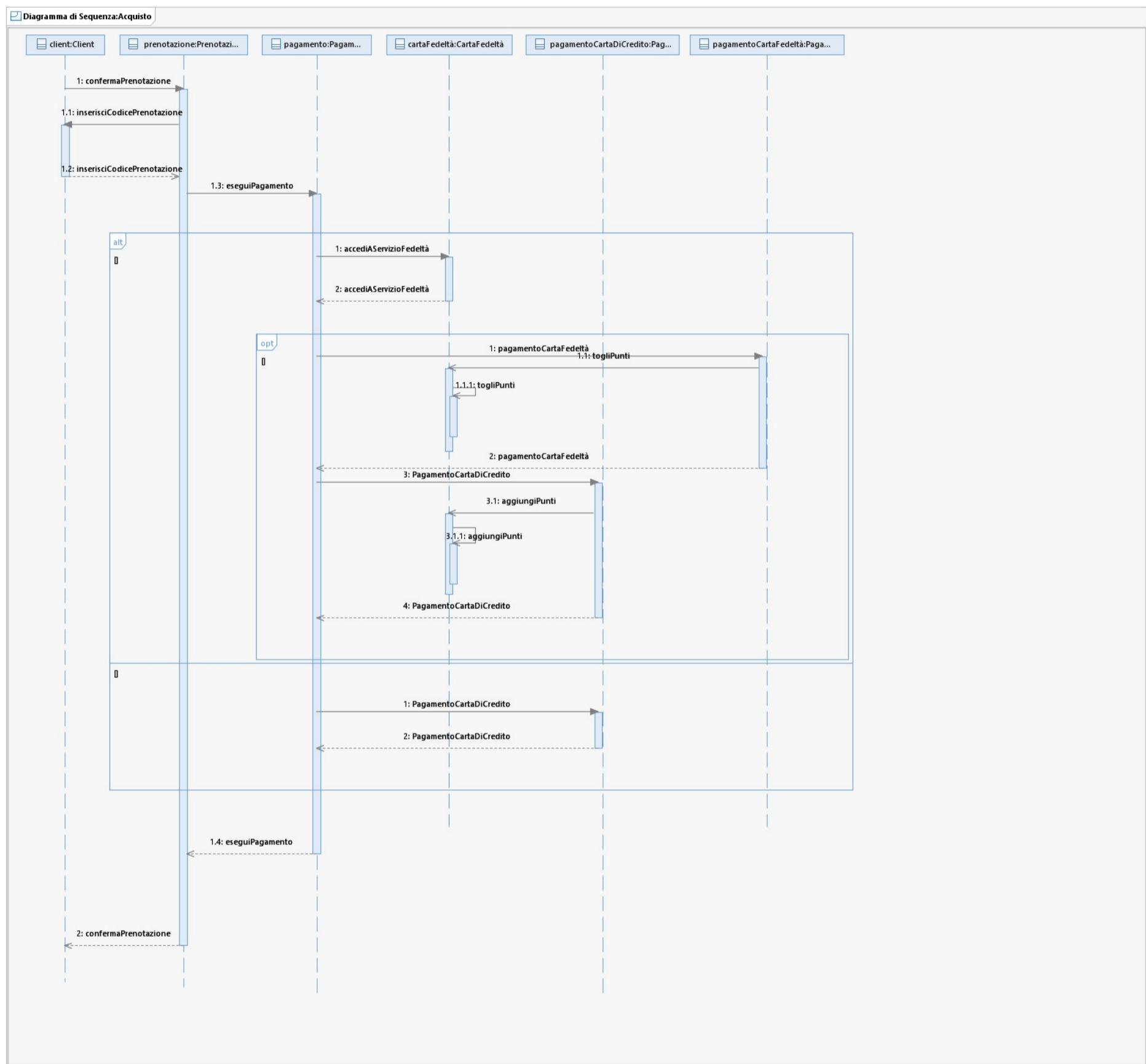


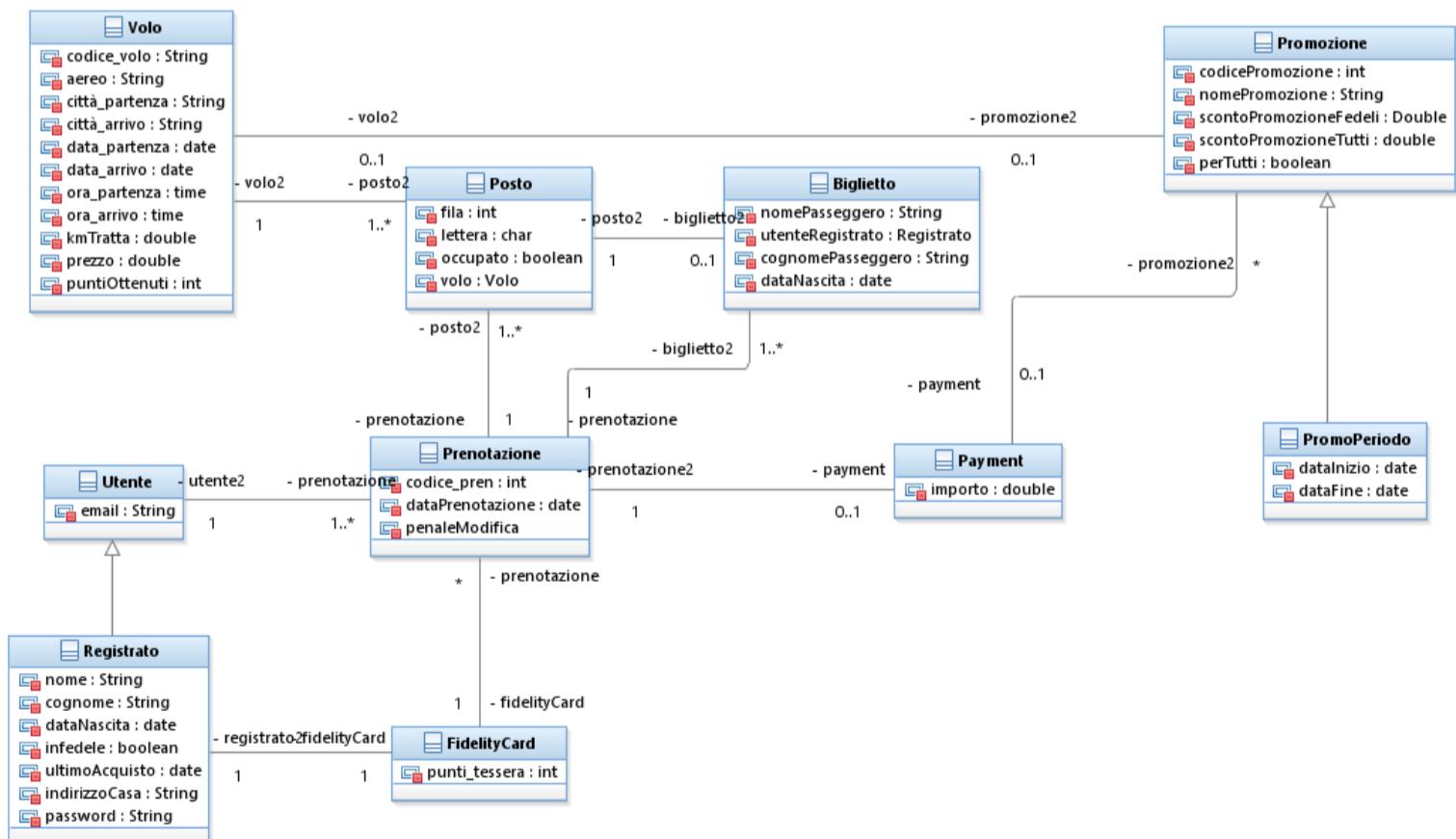
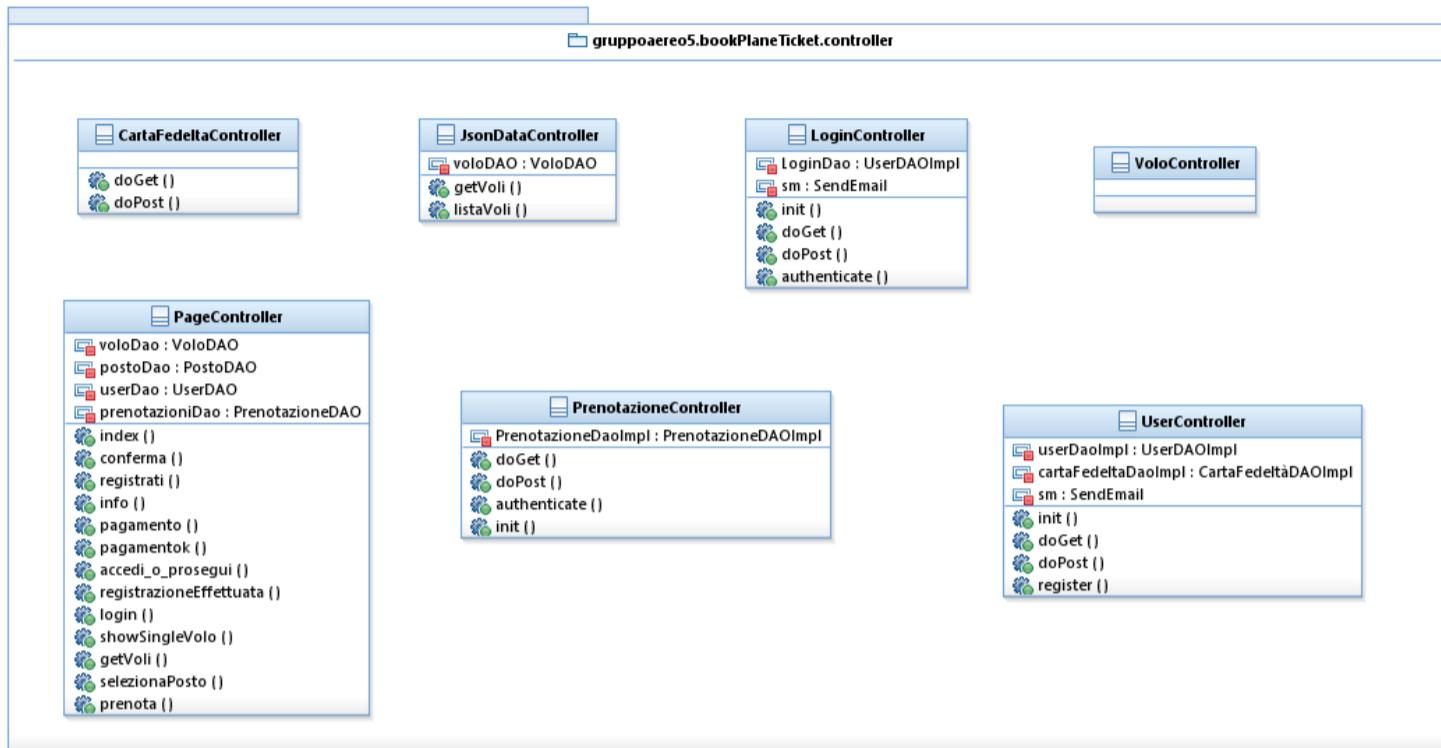
Diagramma di sequenza

Diagramma di sequenza dell'acquisto di un biglietto.



Modello di dominio

Il modello di dominio rappresenta le varie entità che fanno parte del sistema e le loro relazioni.



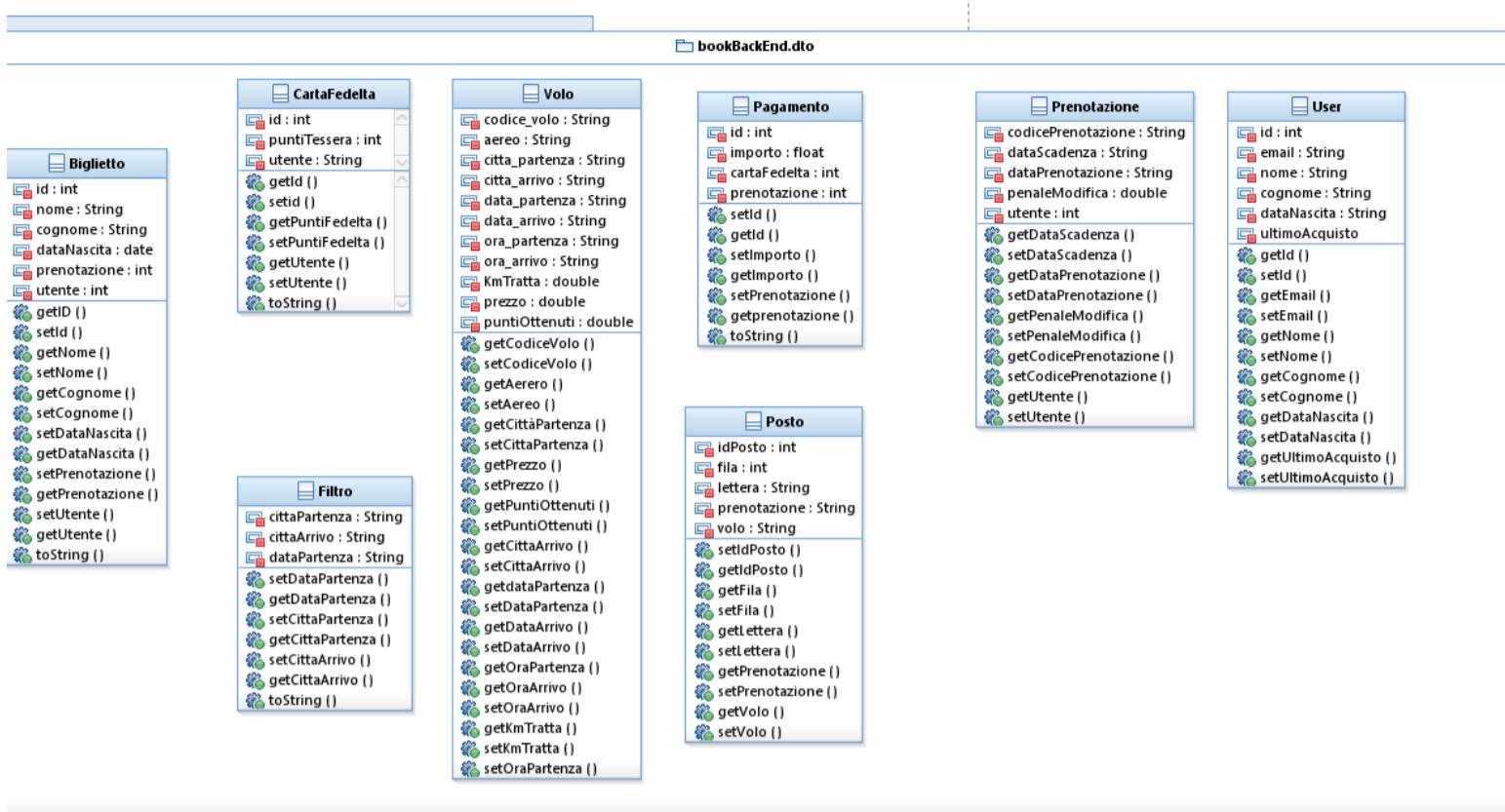


Diagramma dei Package

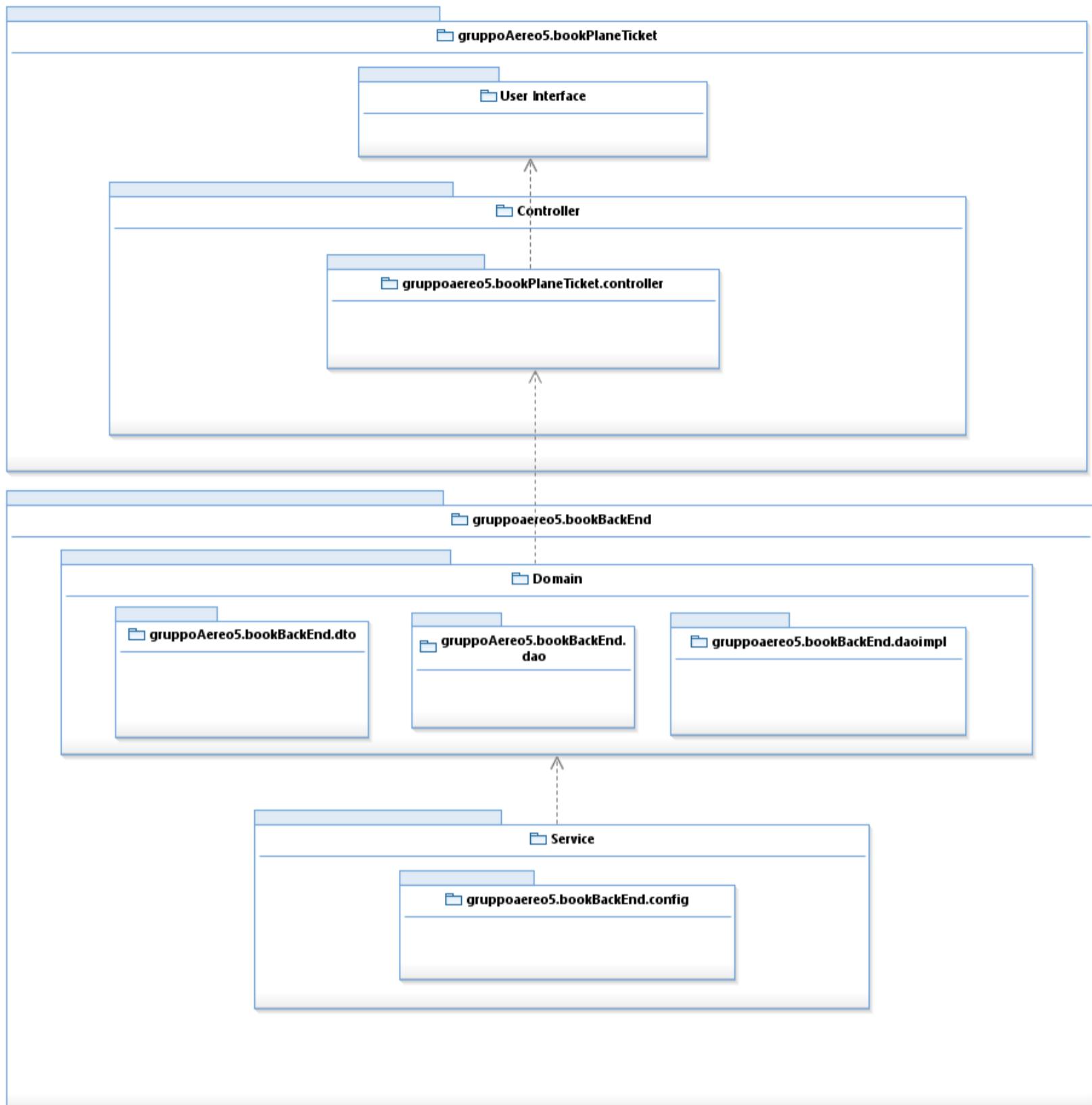


Diagramma dell'Architettura Software

Controller

Back-End

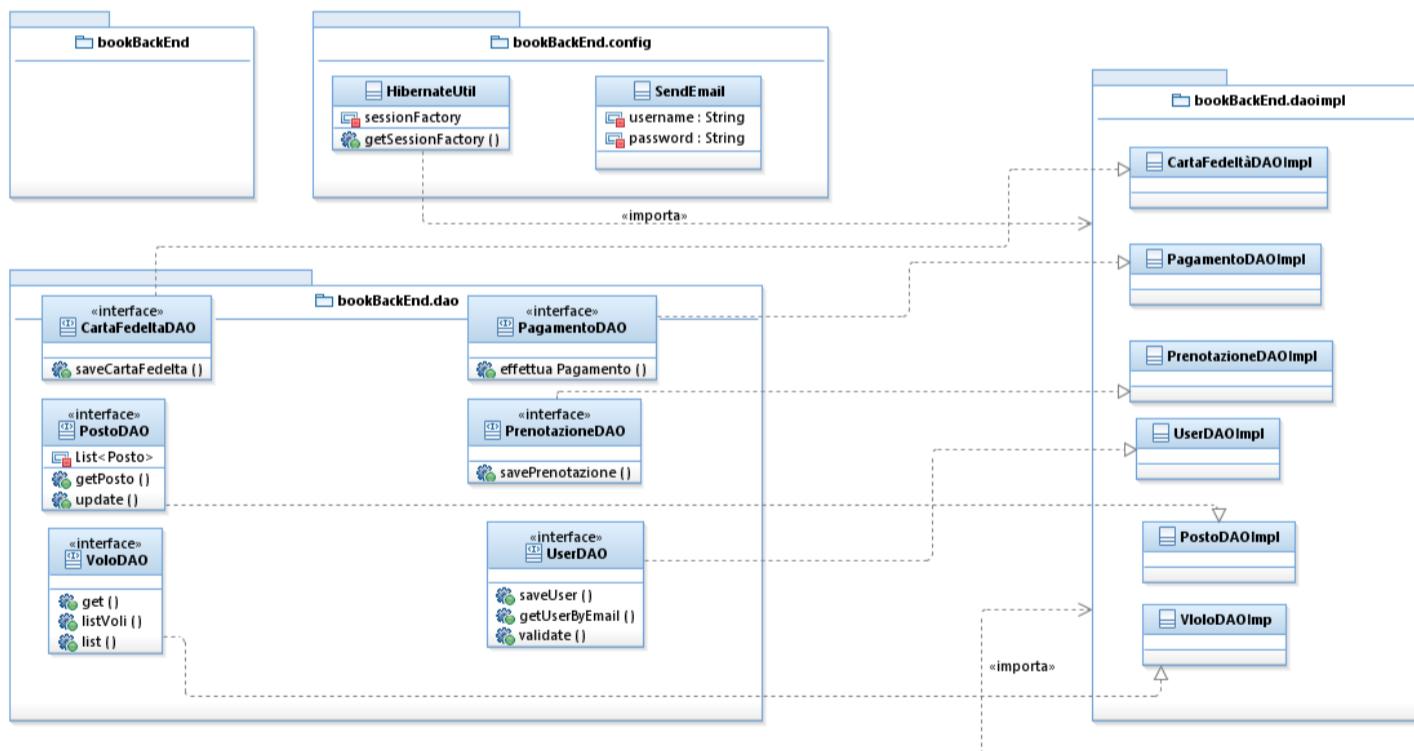
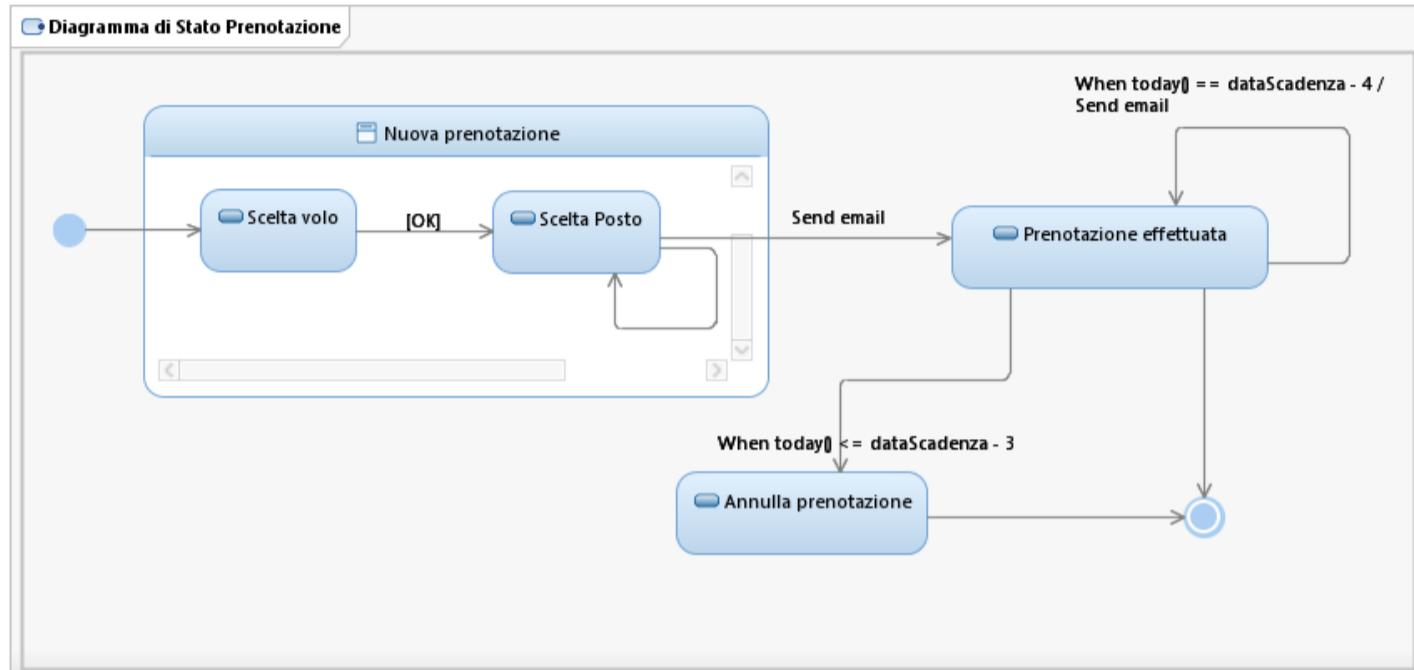
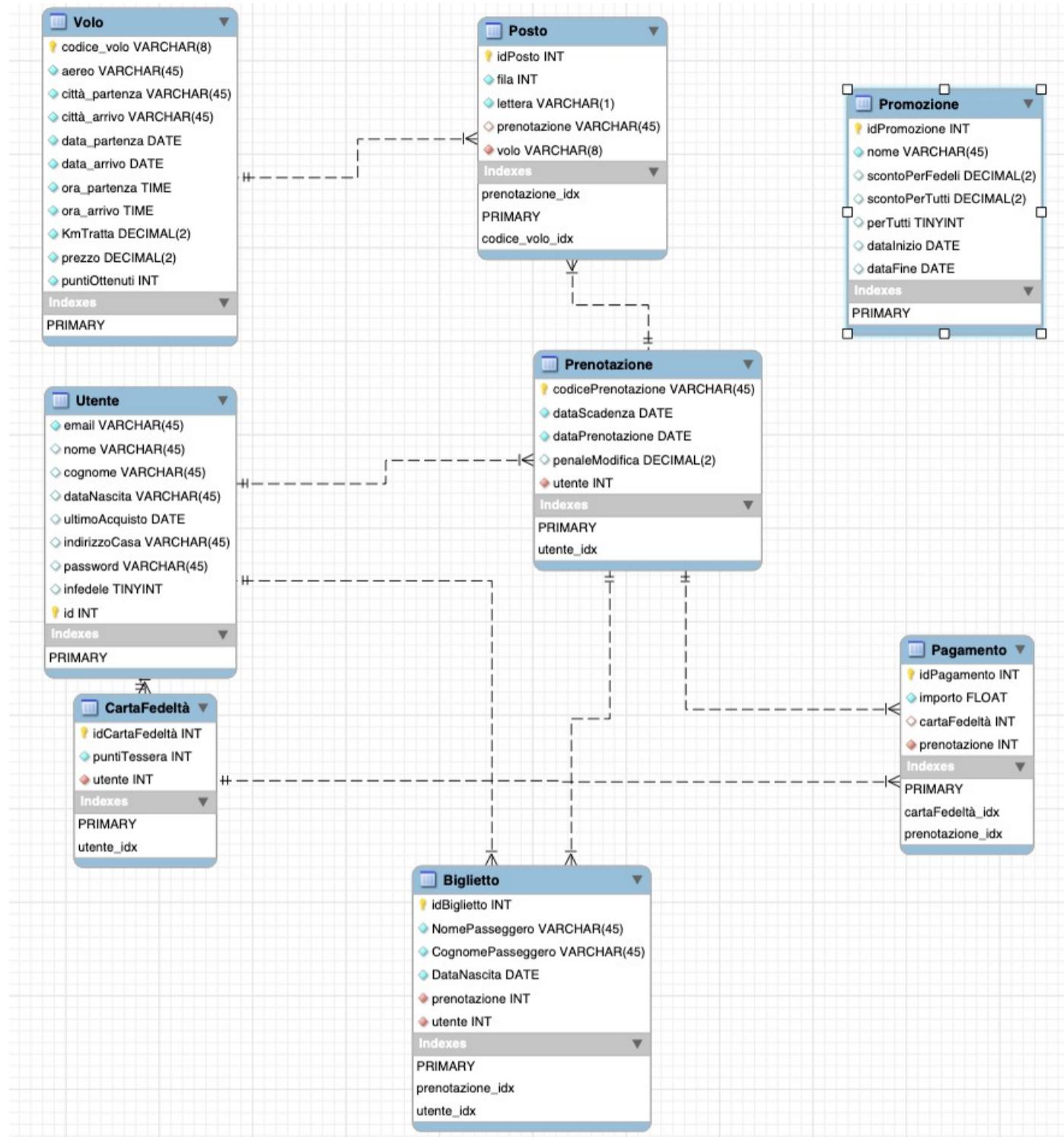


Diagramma di stato

Il diagramma di stato specifica gli stati e i cambiamenti di stato di un oggetto causati da eventi o segnali inviati da altri oggetti. In questo diagramma rappresentiamo gli stati della classe Prenotazione.



DATABASE: Schema ER



Design Pattern

I design pattern da noi utilizzati nella realizzazione di questo progetto sono legati tutti ai framework che abbiamo deciso di sfruttare. Infatti, framework come Hibernate o Spring web sfruttano molti design pattern.

HIBERNATE è un framework che fornisce un servizio di Object-relational mapping (ORM) ovvero gestisce la persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su DB relazionale di un sistema di oggetti java.

I design pattern che utilizza il framework Hibernate sono:

- Data Access Object (DAO) Design Pattern: è un pattern architettonico per la gestione della persistenza. Si tratta fondamentalmente di una classe con relativi metodi che rappresenta un'entità tabellare di un DBMS.
- Abstract Factory: essa fornisce un'interfaccia per creare oggetti di famiglie di oggetti connessi o dipendenti tra loro, in modo che non vi sia necessità da parte del client di specificare i nomi delle classi concrete all'interno del proprio codice. In questo modo si permette ad un sistema di essere indipendente dall'implementazione degli oggetti concreti e che il client, attraverso l'interfaccia, utilizzi diverse famiglie di prodotti.
- Data Mapper: Definisce una classe che si occupa della materializzazione e “dematerializzazione” degli oggetti con l'obiettivo di aumentare le performance del sistema.

SPRING FRAMEWORK è un framework che utilizza soprattutto quattro design pattern, ovvero:

- Singleton Pattern
- Factory Method Pattern
- Proxy Pattern
- Template Pattern

Nel nostro progetto è possibile soprattutto notare il Singleton Pattern. Esso è un design pattern che ha lo scopo di garantire che di una determinata classe venga creata una ed una sola istanza, e di fornire un punto di accesso globale a tale istanza.

In diversi casi abbiamo utilizzato `@Autowired` con uno scopo simile a questo:

```
1 | @Component("fooFormatter")
2 | public class FooFormatter {
3 |
4 |     public String format() {
5 |         return "foo";
6 |     }
7 | }
```

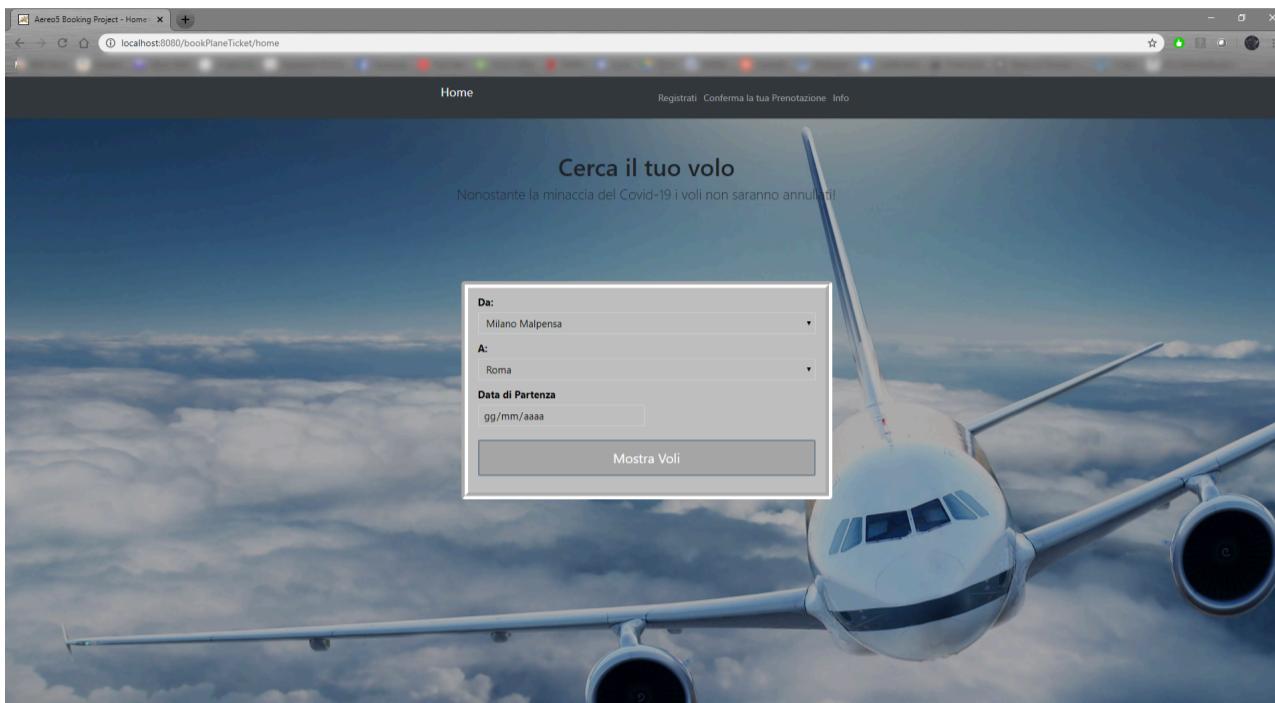


```
1 | @Component
2 | public class FooService {
3 |
4 |     @Autowired
5 |     private FooFormatter fooFormatter;
6 |
7 | }
```

Tutorial di utilizzo

Gestione Biglietti Compagnia Aerea

La pagina iniziale che viene aperta una volta avviata l'applicazione “Gestione Biglietti Compagnia Aerea” è la seguente HomePage:

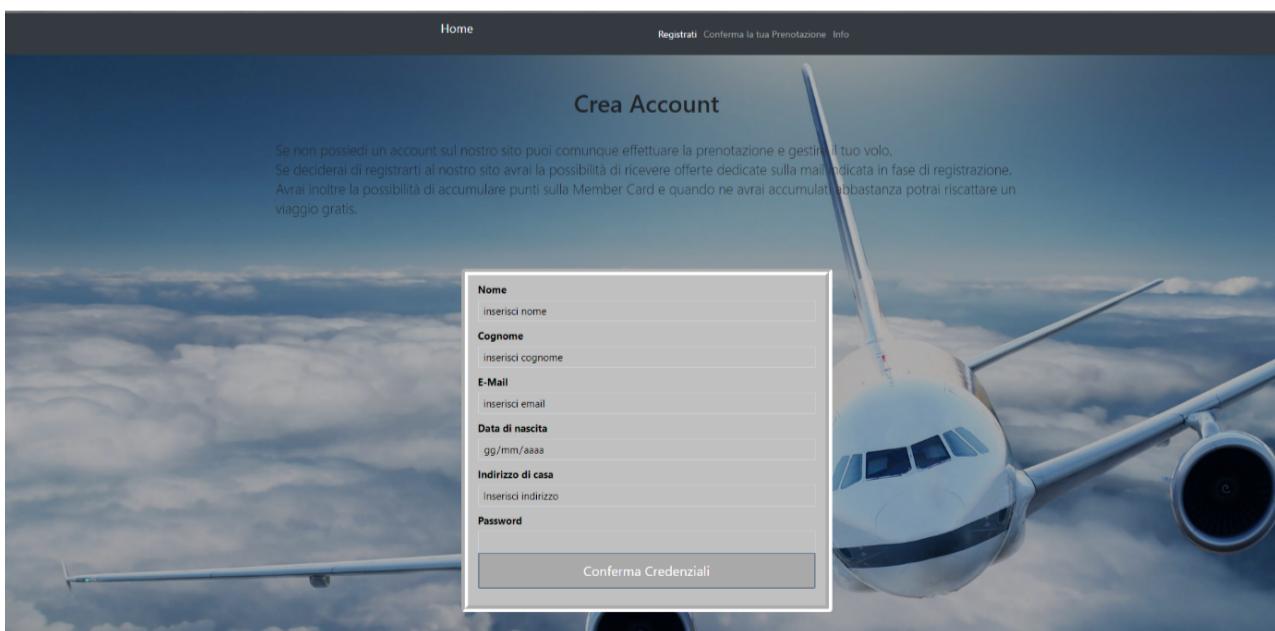


La parte Front-End è stata realizzata con un modello template base di Bootstrap ("<https://startbootstrap.com/templates/bare/>").

Dalla Navigation-bar in alto è possibile accedere alle funzionalità che offre il sistema: registrazione oppure conferma di una prenotazione fatta in precedenza.

Il cliente che utilizza tale sistema è libero di cercare un determinato volo inserendo i dati richiesti senza essere necessariamente registrato.

Il pulsante della registrazione porta alla pagina per la creazione dell'account dove vengono richiesti i dati all'utente per diventare cliente fedele.



Dopo la conferma dei dati inseriti verrà inviata un'e-mail con la conferma della prenotazione.

 gruppo5.progettoaereo@gmail.com
a me ▾

Ciao Rossi Mario, la tua iscrizione è avvenuta con successo.

...

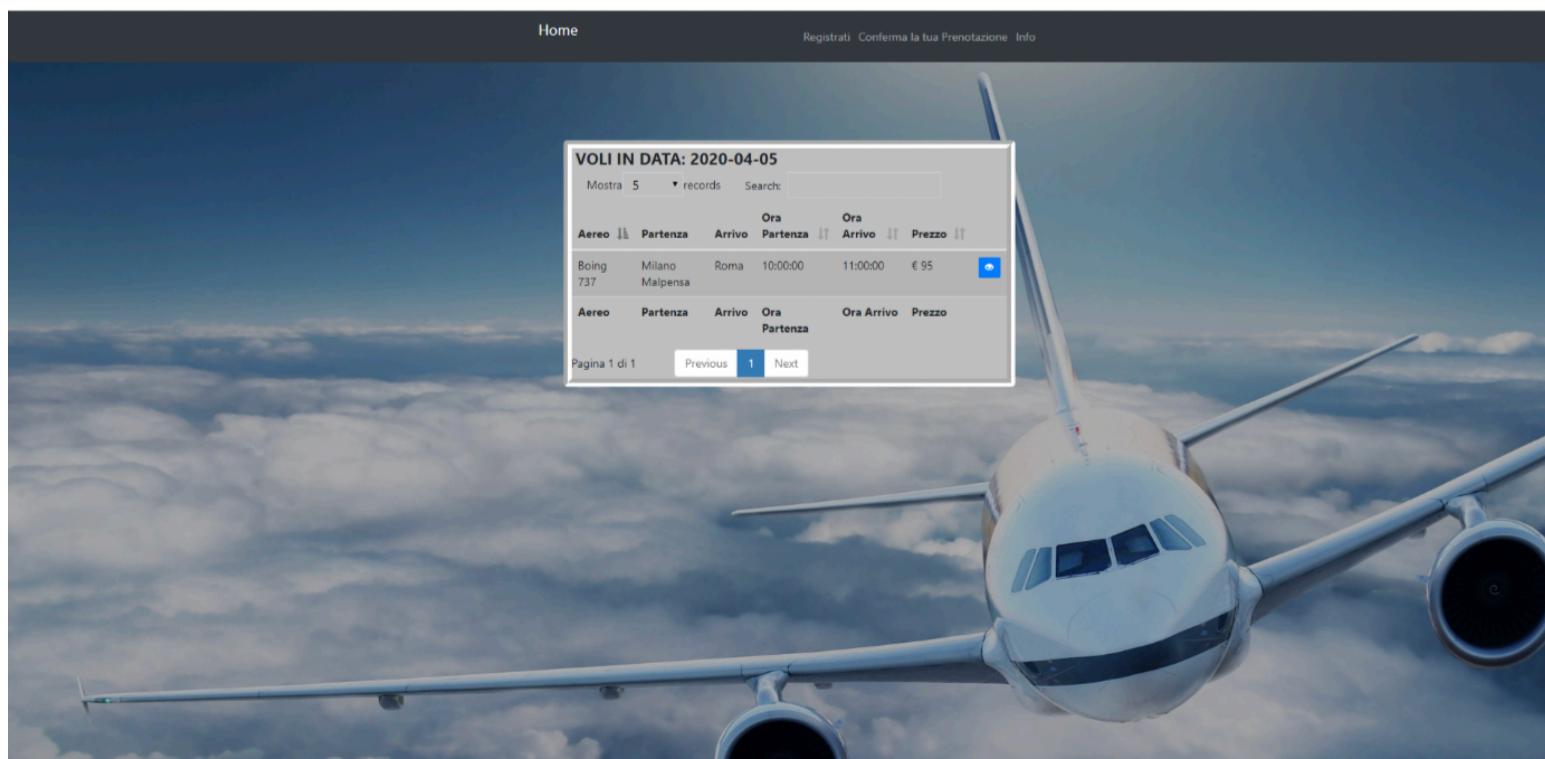
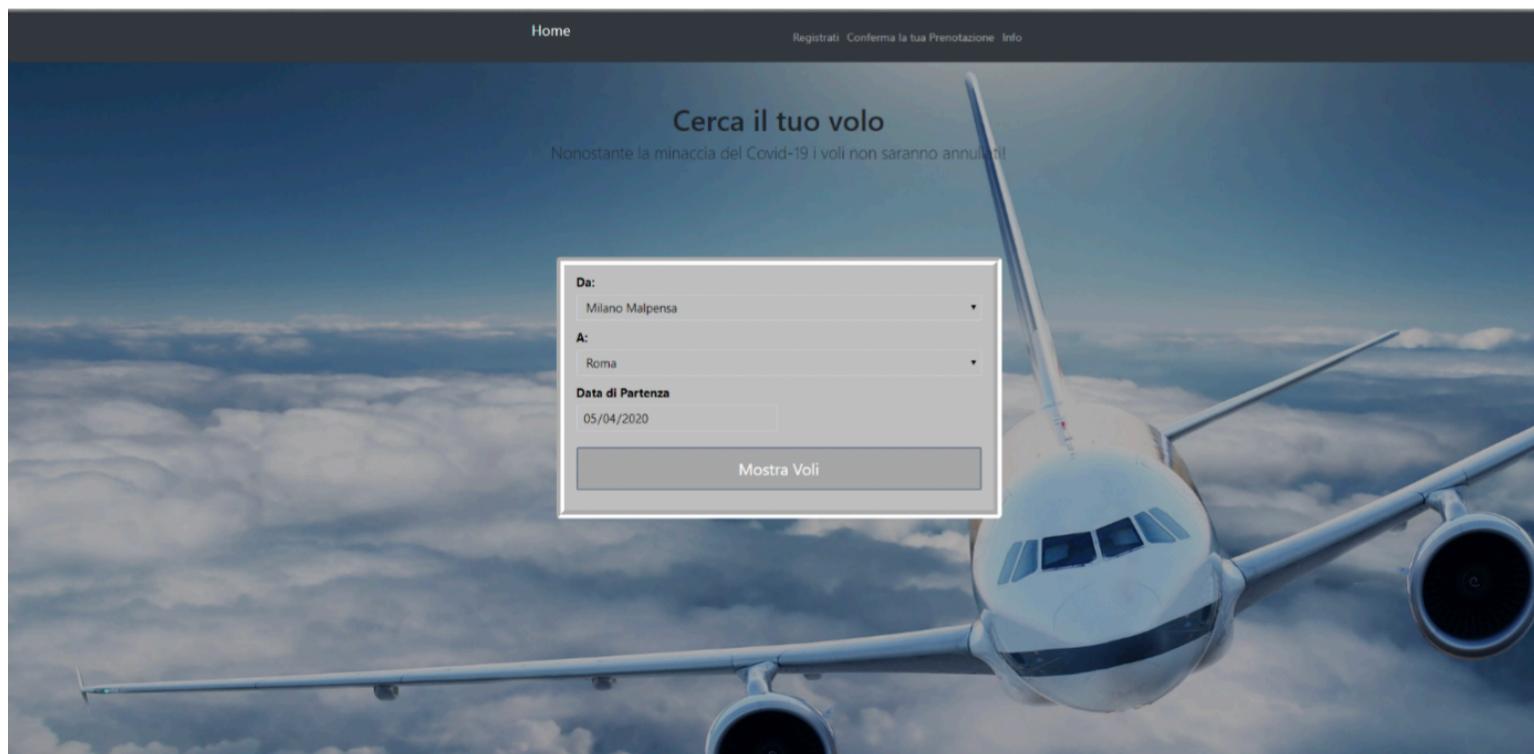
...

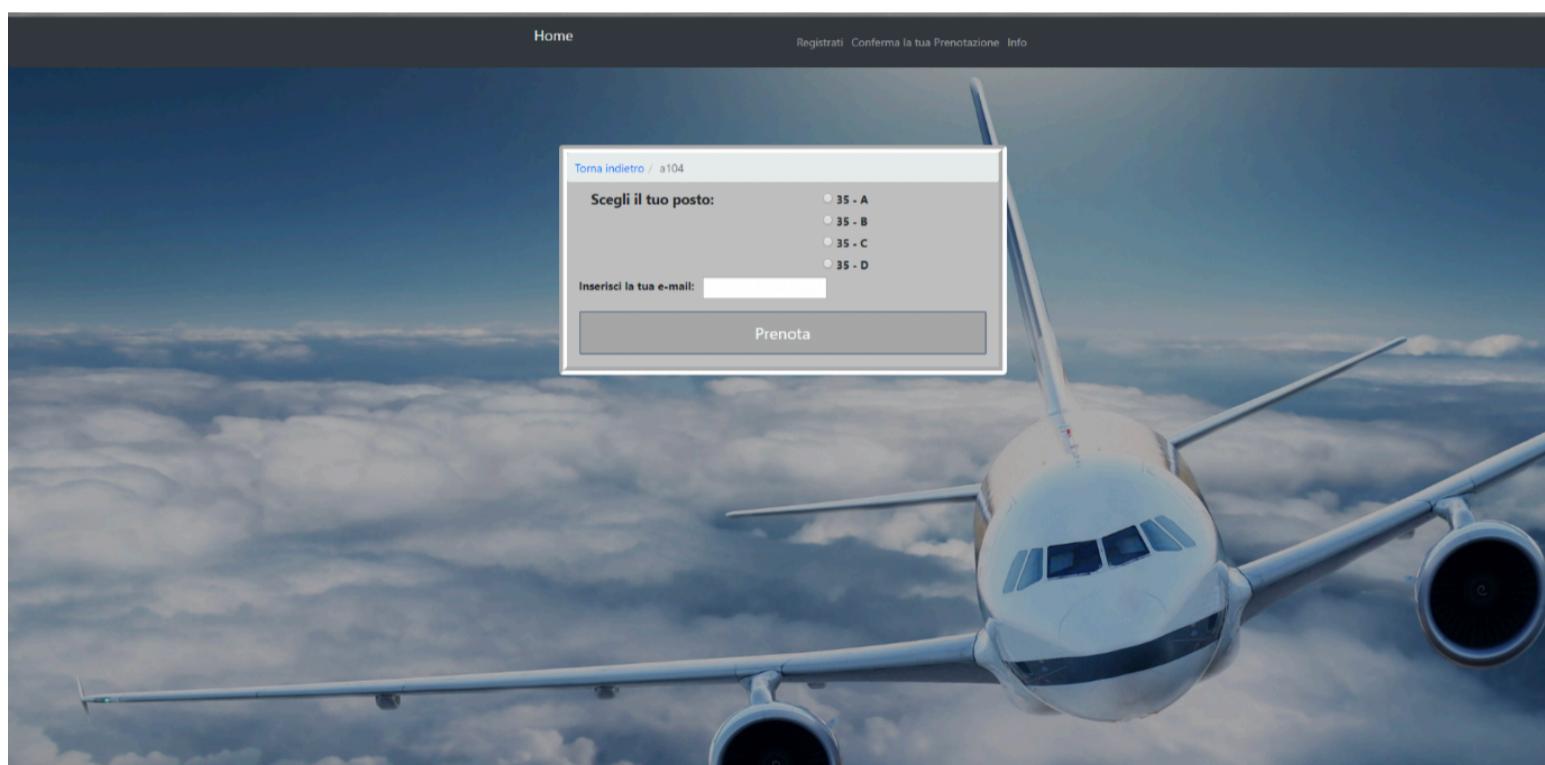
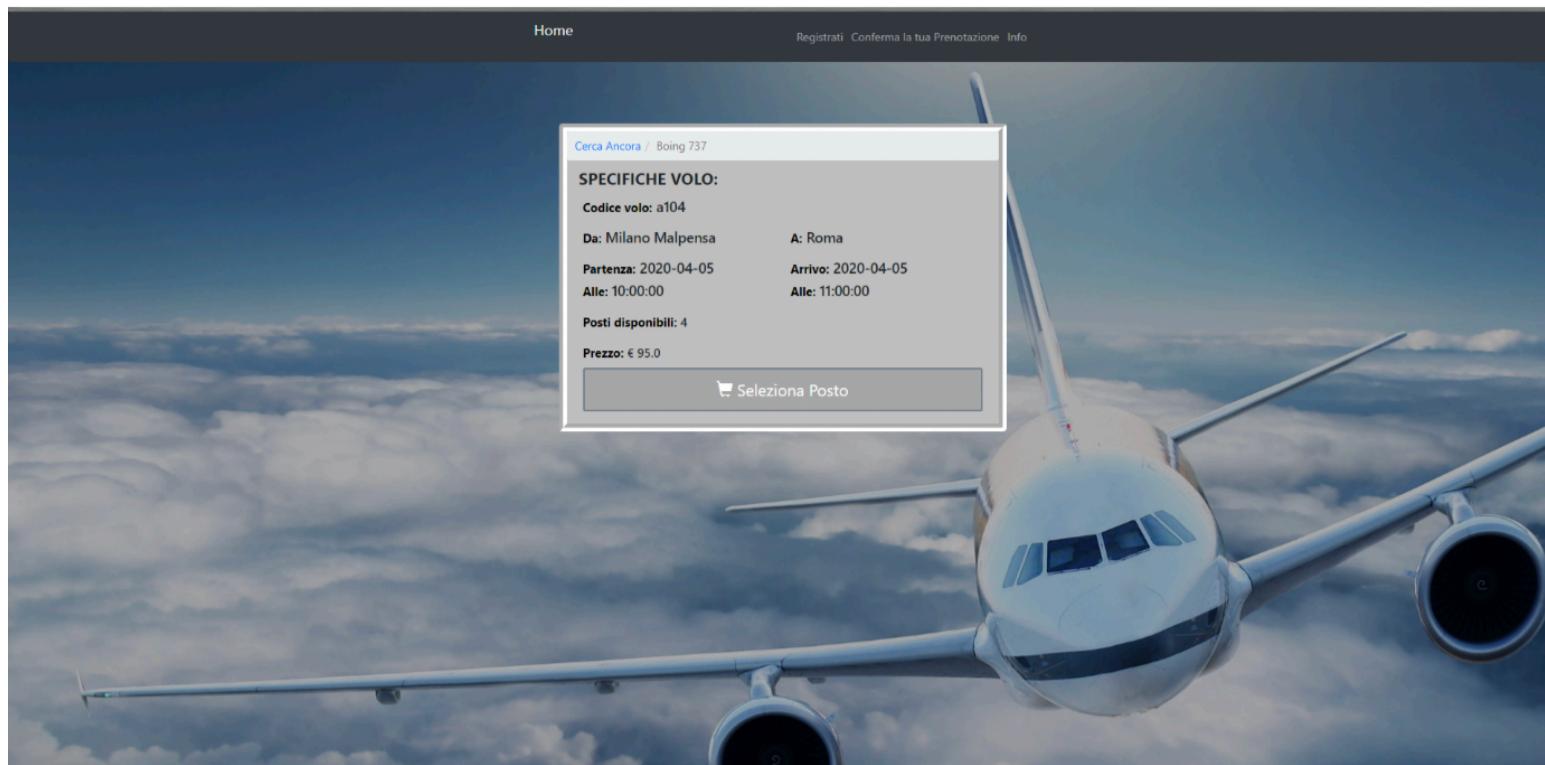
[Messaggio troncato] [Visualizza intero messaggio](#)

 Rispondi

 Inoltra

La ricerca e la prenotazione di un volo non richiedono la registrazione al sistema. Il cliente, in questo caso, sarà tenuto a fornire una propria mail dove riceverà, una volta confermata la scelta del posto, il codice identificativo necessario per poter proseguire con il pagamento.





Una volta che il cliente, registrato o non, avrà selezionato il volo interessato, il posto che andrà ad occupare e dopo aver inserito la mail riceverà il codice relativo alla prenotazione con il quale potrà proseguire cliccando su “Conferma la tua Prenotazione”.

Conferma prenotazione Posta in arrivo x

 gruppo5.progettoaereo@gmail.com

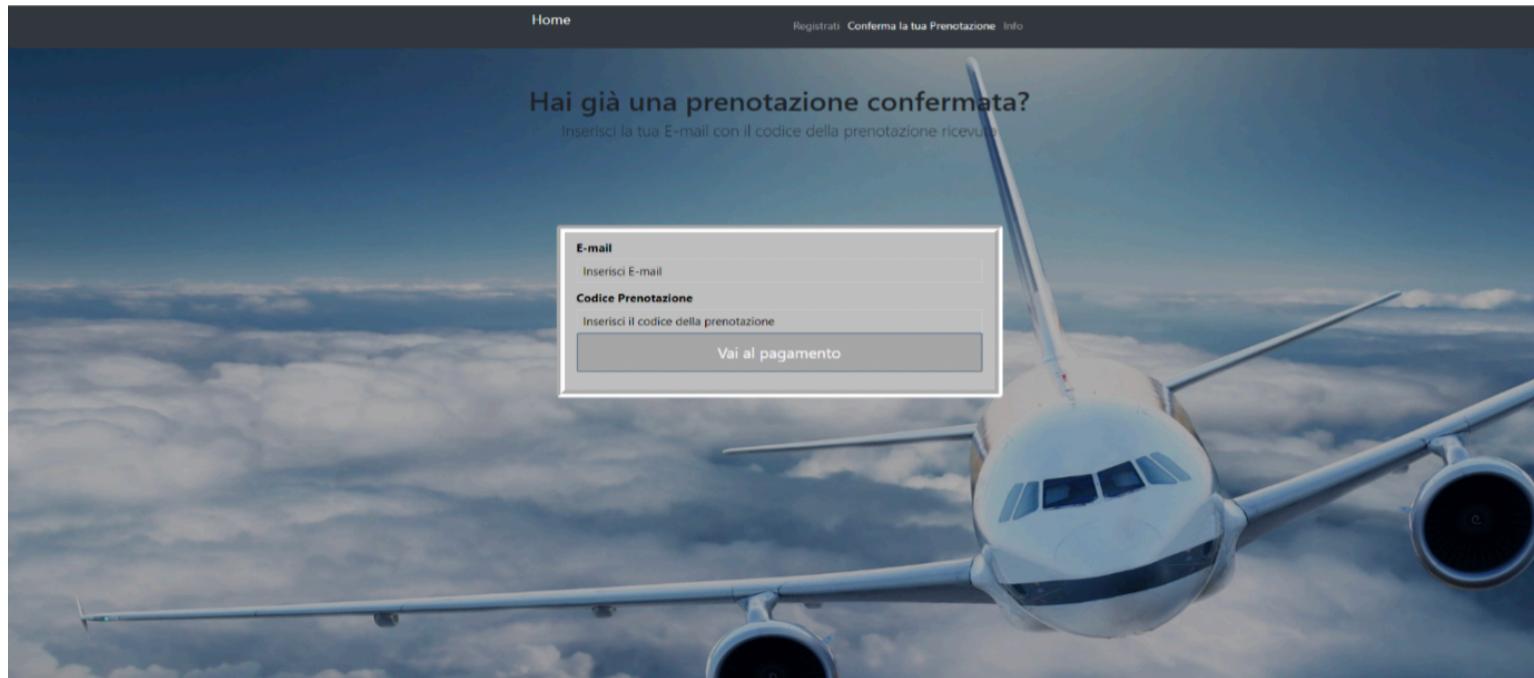
a me ▾

Il codice della sua prenotazione è: VilU

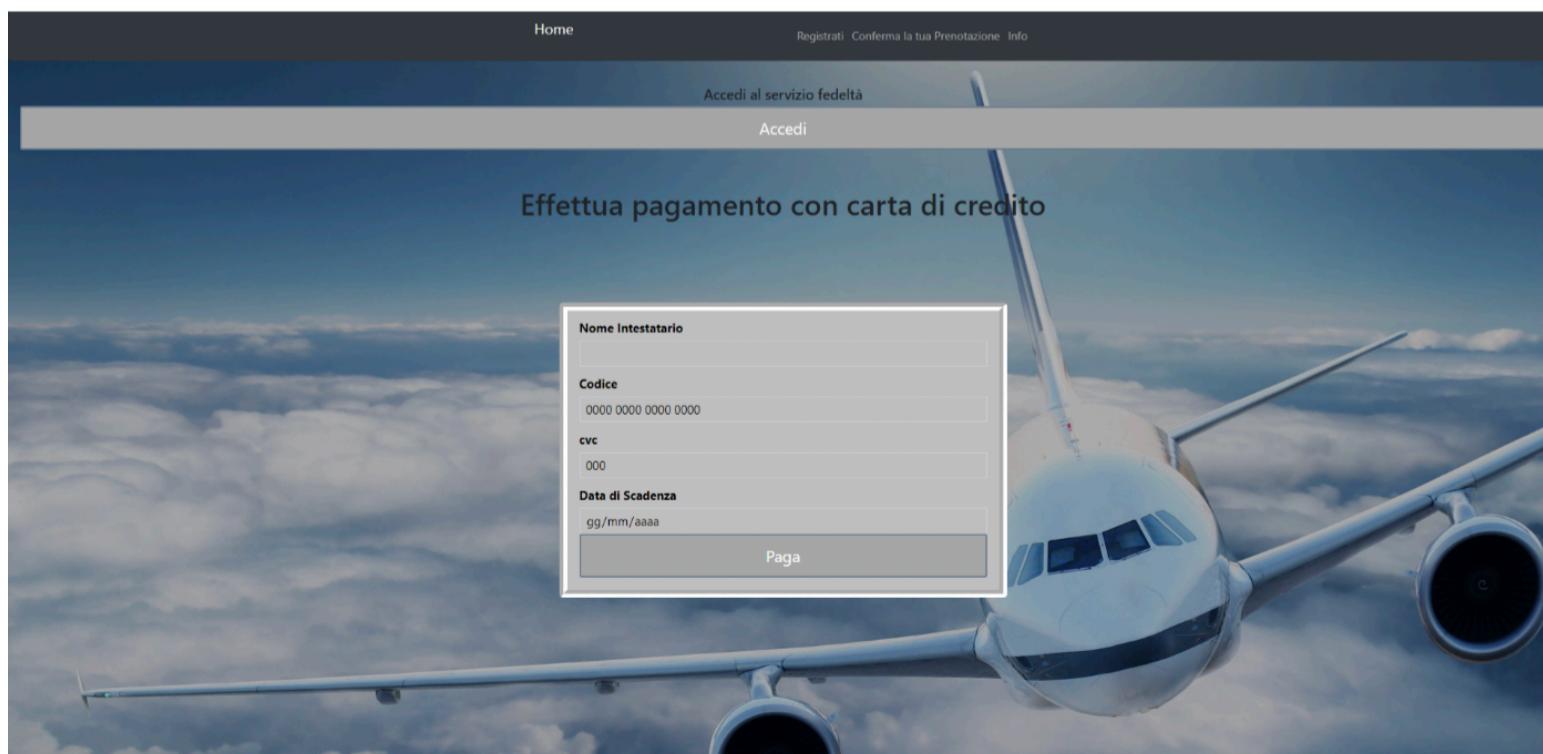
...

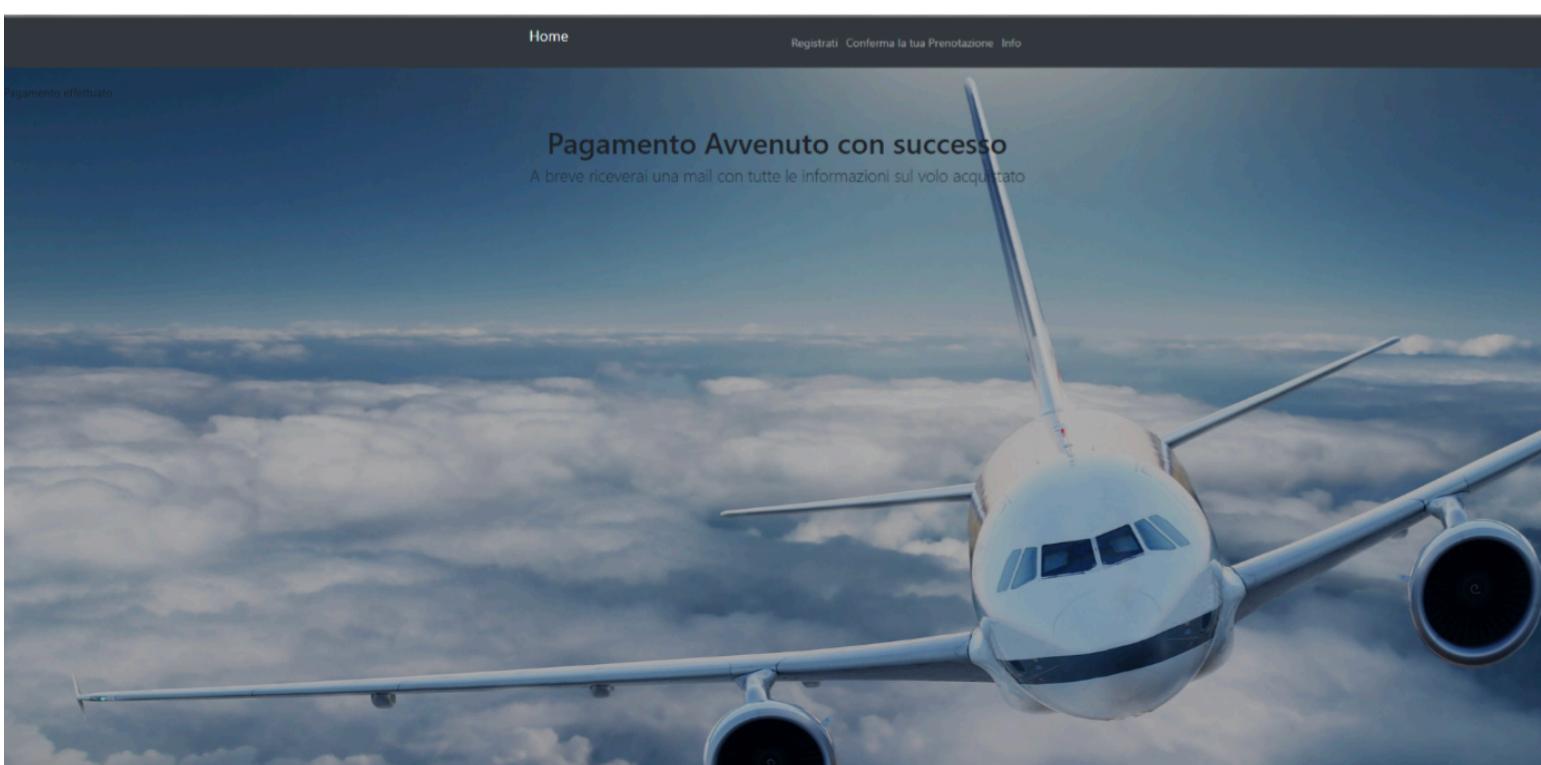
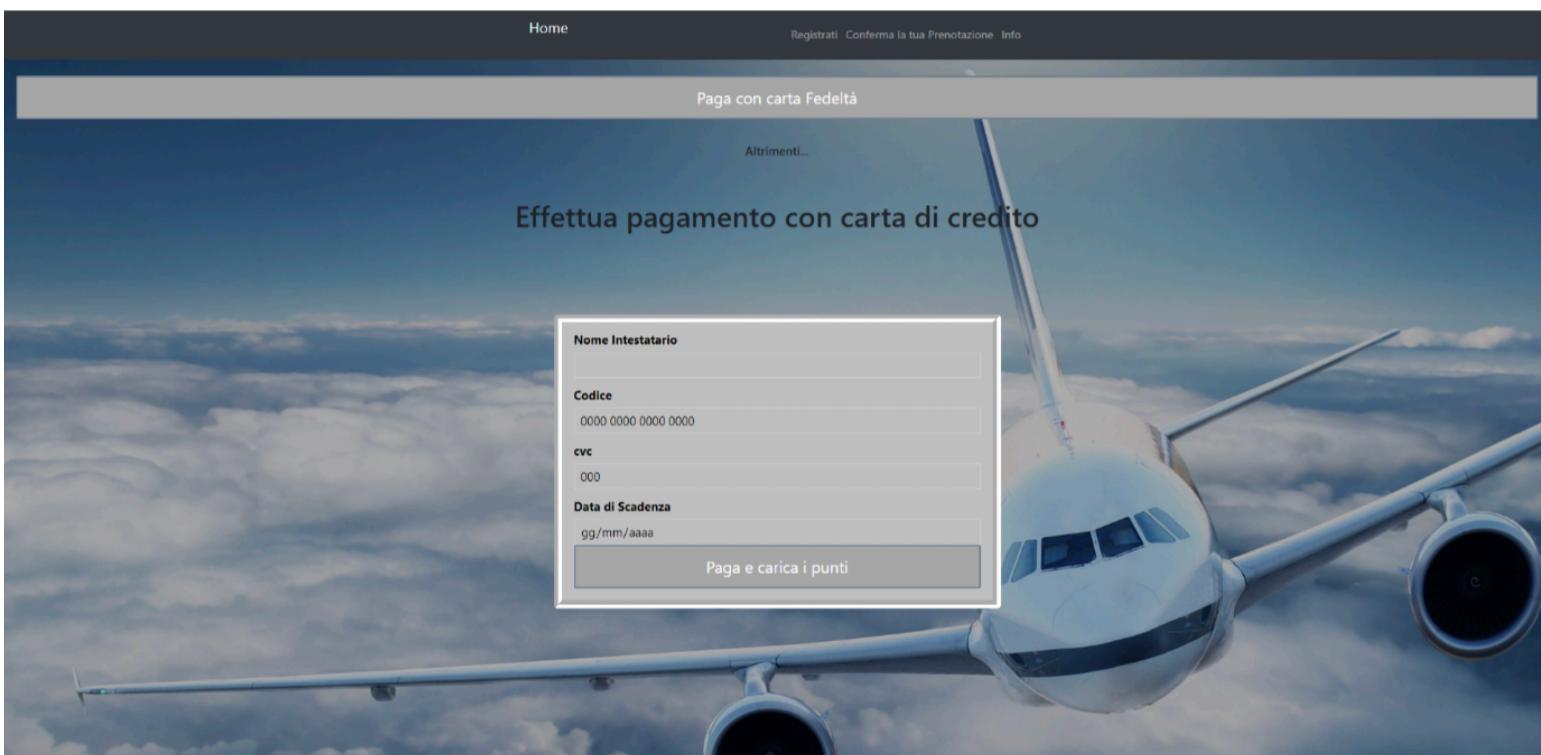
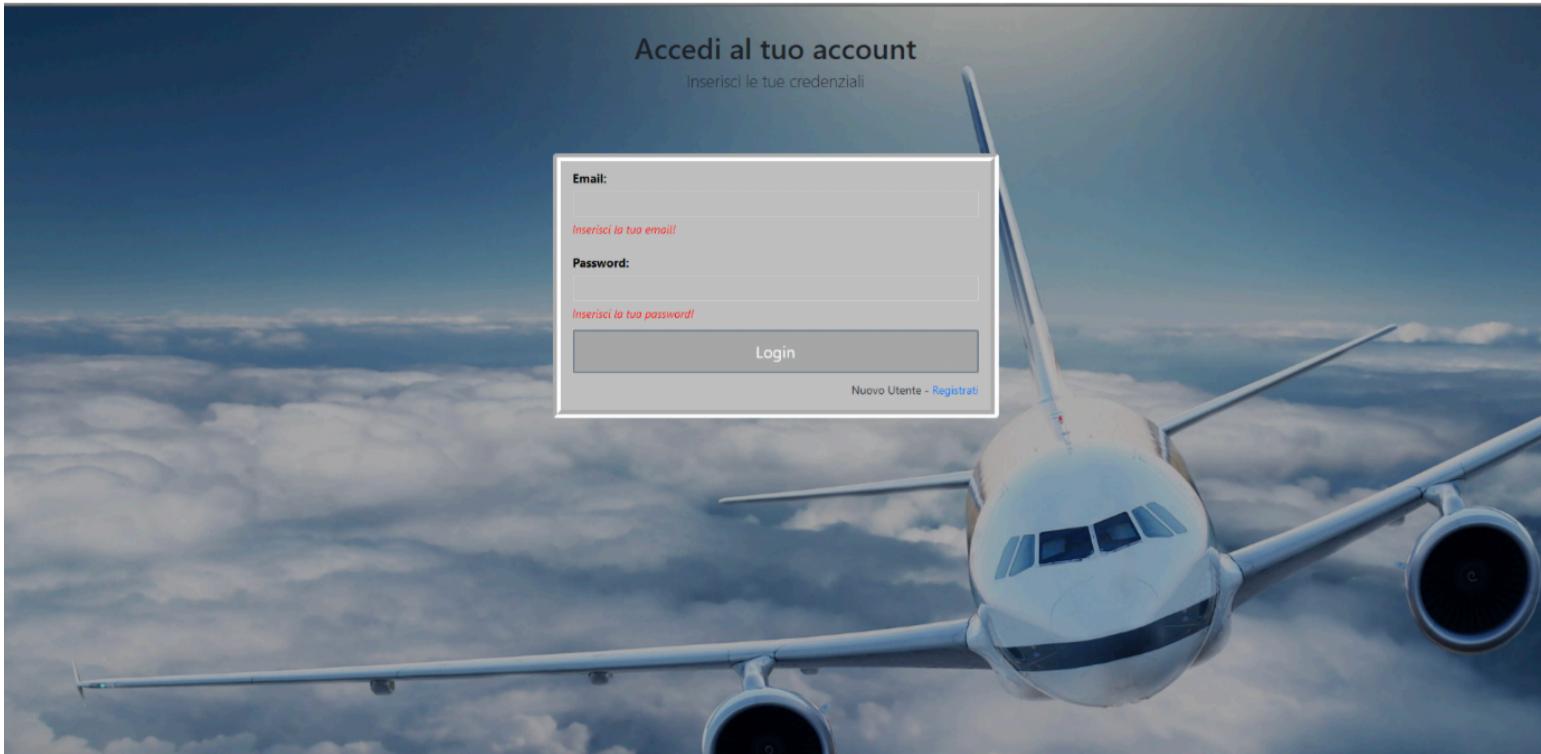
[Messaggio troncato] [Visualizza intero messaggio](#)

La pagina del pagamento prevede la liquidazione del costo della prenotazione tramite carta di credito a meno che il cliente non sia un “cliente fedele”. Nel caso in cui il cliente sia registrato al sistema avrà la possibilità di effettuare il pagamento anche usufruendo dei punti presenti sulla carta sempre che questi siano sufficienti. Un cliente non registrato al servizio potrà proseguire con l’inserimento dei dati della propria carta oppure registrarsi e fare successivamente l’accesso.



Una volta effettuato l’accesso la schermata mostrerà al posto del pulsante “accedi” un pulsante “Paga con carta fedeltà”. A pagamento effettuato con successo il cliente verrà rimandato su una pagina di conferma dove gli verrà comunicato che la sua prenotazione è stata pagata e confermata.





Funzionalità mancanti

Abbiamo riscontrato diverse problematiche durante lo sviluppo del progetto che ci hanno impedito di completare tutte le funzionalità del progetto.

I problemi principali sono dovuti alla nostra inesperienza, che ha causato un dilatamento dei tempi, alla connessione con database e l'implementazione di Spring Security.

A causa di queste problematiche le seguenti funzionalità non sono state sviluppate:

- Annullamento prenotazione
- Modifica data/ora volo acquistato
- Gestione delle promozioni
- Notifica 24 ore prima della data di scadenza della prenotazione e annullamento prenotazione in caso di mancata conferma
- E' possibile acquistare un solo biglietto alla volta
- Richiesta estratto conto
- Gestione infedeltà utente

Guida installazione

Dopo aver importato i due file “bookBackEnd” e “bookPlaneTicket” all’interno del Project Explorer di Eclipse, il corretto funzionamento del programma prevede che venga avviato il file “h2-1.4.200.jar” dalla cartella bookBackEnd/src/resources/lib per connettersi al database H2 e configurarlo correttamente. Nella casella di accesso al database è necessario che le impostazioni siano impostate nel seguente modo:

- Classe driver : org.h2.Driver
- JDBC URL: jdbc:h2:./localhost/~/bookPlaneTicket
- Nome utente: gruppo5
- Password: progetto

Completato questo passaggio è necessario accedere al file databaseQueries.sql nella cartella bookBackEnd, selezionare tutto quello che c’è scritto al suo interno e copiare i comandi SQL nella casella di testo indicata in H2 dopo l’accesso. In questo modo verranno generate le tabelle e i dati necessari per poter usufruire della web application. Prima di avviare l’applicazione è necessario chiudere h2 completamente dalla barra sul pc in basso a destra. H2 non permette connessioni simultanee perciò potrebbe causare problemi durante l’avvio della web application.

Una volta configurato il database, fare maven → Update Project dal menù che si apre cliccando con il tasto destro del mouse una delle due directory per essere sicuri che tutte le dipendenze siano aggiornate.

Infine è possibile avviare il progetto : run → run on server, con il server Tomcat integrato in eclipse e visitare la pagina <http://localhost:8080/bookPlaneTicket/> .

Conclusioni

Ogni membro del gruppo si è ritiene soddisfatto del lavoro fatto. Il progetto ci ha dato la possibilità di metterci alla prova e, in maniera autonoma, cercare soluzioni ai diversi problemi che ci si sono posti davanti.

L’inesperienza del gruppo, insieme ai disagi creati dal Covid-19, hanno allungato un po’ i tempi pianificati e questo ci ha costretto a non implementare alcune parti del progetto, che inizialmente avevamo intenzione di fare.

Siamo tuttavia contenti del lavoro svolto e riteniamo tutto ciò utile come esperienza da cui possiamo trarre benefici futuri.