

Corso di Ingegneria del Software Deliverable di progetto	2021-2022
---	-----------

“Ingegneria del Software” 2021-2022

Docente: Prof. Angelo Furfaro

<Acquisto-Biglietti-Aerei>

Data	<11-09-2023>
Documento	Documento Finale – PDF

Team Members		
Nome e Cognome	Matricola	E-mail address
Constantin Adrian Antoci	220461	Ntccst01e23z129p@studenti.unical.it

Sommario

List of Challenging/Risky Requirements or Task.....	3
A.Storia dell'Arte.....	4
B.Raffinamento dei Requisiti.....	5
A1. Requisiti con prioritizzazione.....	5
A2. Requisiti non funzionali.....	8
A3. Scenari casi D'uso.....	9
A4. Excluded Requiriments.....	15
A5. Assunzioni.....	15
A6.Use case Diagram.....	16
C.Architettura Software.....	17
C1.The static view of system and component Diagram.....	17
C2.The dynamic view of the software architecture: Sequence Diagram.....	18
D.Dati e loro modellazione.....	20
E.Scelte progettuali.....	21
F.Progettazione di basso livello.....	22
G.Spiegare come il progetto soddisfi i requisiti funzionali e quelli non funzionali....	23

List of Challenging/Risky Requirements or Tasks

Challenging Task	Date the task is identified	Date the challenge is resolved	Explanation on how the challenge has been managed
Realizzare moduli che sfruttassero le strutture dati di Java con gli oggetti generati da GRPC			Per la realizzazione del progetto si e' deciso di utilizzare gli oggetti generati da GRPC, portando ad alcuni problemi in quanto hashCode() generato automaticamente tiene traccia del Timestamp di creazione dell'Oggetto generando cosi errori di incompatibilità tra oggetti che hanno attributi identici (Quindi per logica uguali) per la soluzione si e' proceduto a modificare a modificare l'hashCode() della classe Cliente(Principale protagonista)
Riuscire a gestire i file Json			Si è utilizzata la libreria Gson per la serializzazione delle informazioni riguardanti le prenotazioni e i voli e i clienti fedeltà sviluppata da Google quindi compatibile con GRPC
Organizzazione delle informazioni in strutture Dati			Ho utilizzato una lista contenente tutti i voli disponibili, Una mappa contenente per ogni volo tutti i Clienti che si sono prenotati per quel volo (In questo modo riesco a garantire che lo stesso clienti non si prenoti due volte per lo stesso volo) Mentre una Mappa avente come chiave in l'oggetto Cliente e come argomento un'altra' mappa avente come chiave un intero(chiave segreta) e una lista dei vari estratto conto

A. Stato dell'Arte

Effettuando delle ricerche non sono riuscito a trovare applicazioni per la quale sono certo che usino GRPC e che abbiano la funzionalità di prenotazione di biglietti aerei, questo a causa della privacy di queste applicazioni , di sicuro e' molto conveniente utilizzare GRPC anche per applicazioni di questo tipo in quanto offrono numeri benefici permettendo una comunicazione efficiente tra client e server , semplificando lo sviluppo di un sistema distribuito.

B. Raffinamento dei Requisiti

Si desidera progettare un sistema software distribuito che permetta a un cliente di visionare voli disponibili, prenotare voli, modificare i voli .

Requisiti Funzionali

- 1) Visionare le informazioni sui Voli
- 2) Prenotarsi per un volo fornendo (già all'inizio) Nome-Cognome-Codice Fiscale-Email-Indirizzo-Data di Nascita (se si e' clienti Fedeltà bisogna fornire anche il codice segreto)
- 3) Iscrizione al programma "Cliente Fedeltà"
- 4) Visionare le prenotazioni effettuate
- 5) Visionare le notifiche da parte del Server
- 6) Visionare L'estratto conto
- 7) Modificare il volo

Requisiti non Funzionali

- 1) L'applicazione Client deve avere una GUI
 - 2) L'applicazione Server deve avere una GUI di amministrazione
-

A.1 Servizi (con prioritizzazione)

Visionare le informazioni sui Voli

- **Descrizione:** Fornisce al cliente la possibilità di visualizzare le informazioni(Partenza-Destinazione-Data) di un determinato volo

- **Soluzione concettuale:** Implementazione di una funzionalità per recuperare i voli e presentare le informazioni riguardanti quel volo tramite JButton che premendolo mi crea una nuova finestra e mi mostra in forma tabellare tutte le informazioni riguardanti i voli.
- **ID del servizio:** S1
- **Importanza:** Alta
- **Complessità:** Bassa

Prenotarsi per un volo

- **Descrizione:** Permette a un cliente di prenotare un volo, naturalmente una prenotazione deve contenere le informazioni del cliente, all'inizio prima di avere accesso alle funzionalità bisogna sostanzialmente fornire tutte le informazioni necessarie come Nome-Cognome-Data di Nascita-Email-Indirizzo-Codice Fiscale
- **Soluzione concettuale:** Creazione di un'interfaccia che permette al client di decidere quale volo prenotare e che registri tale prenotazione, ad ogni prenotazione si riceveranno dei feedback.
- **ID del servizio:** S2
- **Importanza:** Alta
- **Complessità:** Media

Iscrizione al "Programma Fedeltà"

- **Descrizione:** Permette a un cliente di iscriversi al programma Fedeltà in maniera tale da avere accesso ad alcune promozioni sulla base dei punti raccolti, i punti si raccolgono per ogni prenotazione di volo effettuato in maniera casuale
- **Soluzione concettuale:** Implementazione consiste in un JButton che effettua una richiesta al server inviando le informazioni dei client e le memorizza in un Json, inoltre, il server risponderà con il codice segreto unico generato casualmente
- **ID del servizio:** S3
- **Importanza:** Alta
- **Complessità:** Alta
-

Visionare le prenotazioni effettuate

- **Descrizione:** Permette a un cliente di visionare i voli per la quale ha effettuato una prenotazione
- **Soluzione concettuale:** Implementazione di una funzionalità per recuperare i voli e presentare le informazioni riguardanti quel volo tramite JButton che premendolo mi crea una nuova finestra e mi mostra in forma tabellare tutte le informazioni riguardanti i voli.
- **Id del servizio:** S4
- **Importanza:** Media
- **Complessità:** Bassa

Visionare le notifiche da parte del Server

- **Descrizione:** Permette a un cliente di ricevere notifiche da parte del Server

- *Soluzione concettuale:* Implementazione consiste in un JButton che permette di aprire una nuova finestra in cui ci sono tutte le notifiche riguardante i voli.
- *Id del servizio:* S5
- *Importanza:* Alta
- *Complessità:* Media

Visionare L'estratto conto

- *Descrizione:* Permette a un cliente di visionare il proprio estratto conto
- *Soluzione concettuale:* Implementazione consiste in un JButton che apre una nuova finestra e che mostra in forma tabellare i punti accumulati per ogni volo
- *Id del servizio:* S6
- *Importanza:* Alta
- *Complessità:* Media

Modificare il volo

- *Descrizione:* Permette a un cliente di modificare il proprio volo
- *Soluzione concettuale:* Implementazione di un interfaccia che riceve come informazioni il volo desiderato in più con la nuova data di partenza , lato server controlla se esiste un volo simile con quella data ed prevede a far prenotare il client su quel volo eliminandolo dal volo precedente.
- *Id del servizio:* S7
- *Importanza:* Alta
- *Complessità:* Media

Interfaccia di amministrazione

- *Descrizione:* Fornire un'interfaccia dedicata agli amministratori per gestire i voli
- *Soluzione concettuale:* Sviluppo di una GUI che permette di aggiungere nuovi Voli
- *Id del servizio:* S8
- *Importanza:* Alta
- *Complessità:* Alta

Gestione della persistenza dei dati

- *Descrizione:* Utilizzo del framework JUnit per testare in modo automatizzato i moduli significativi del software, garantendo il corretto funzionamento delle funzionalità implementate.
- *Soluzione concettuale:* Implementazione di un sistema di persistenza sfruttando file Json che permettono di archiviare e prelevare informazioni.
- *Id del servizio:* S
- *Importanza:* Alta
- *Complessità:* Alta

Testing con JUnit

- *Descrizione:* Consente di memorizzare in modo persistente le informazioni riguardante i voli, clienti e prenotazioni.

- *Soluzione concettuale:* Creazione di suite di test JUnit per i vari moduli del sistema, verificando che i risultati siano conformi alle aspettative. Si utilizzeranno asserzioni e annotazioni di JUnit per facilitare l'esecuzione dei test.
- *Id del servizio:* S9
- *Importanza:* Media
- *Complessità:* Media

Priorità dell'Importanza

- 1) S1
- 2) S2
- 3) S3
- 4) S4
- 5) S5
- 6) S6
- 7) S7
- 8) S8
- 9) S9
- 10) S10

A.2 Requisiti non Funzionali

I Requisiti non funzionali più importanti sono:

Affidabilità: Assicurare che il sistema sia sempre disponibile e che le funzionalità principali siano sempre operative senza interruzioni o errori di vario genere, bisogna considerare che il progetto è pensato per l'accesso simultaneo da parte di un numero consistente di utenti che non possono avere esperienze negative con la prenotazione di un volo.

Prestazioni: *Bisogna ridurre i tempi d'attesa dei clienti, e far in modo che il sistema possa rispondere in maniera efficiente e rapida a tutte le richieste dei client come la visualizzazione dei voli, la prenotazione dei voli, la visualizzazione dei voli prenotati ecc... Questo è possibile anche grazie ad apposite strutture dati.*

Usabilità: *Fornire un'interfaccia intuitiva che permette al cliente di interagire con il sistema in maniera semplice e senza troppe difficoltà.*

Scalabilità: *Il sistema deve essere in grado di scalare, cioè, deve essere in grado di gestire un numero sempre crescente di cliente che prenotano voli senza perdere di prestazioni*

Manutenibilità: *Il sistema è stato progettato con codice pulito e documentato in maniera tale da permettere la manutenzione dello stesso.*

Compatibilità: *Il sistema deve essere compatibile con diverse piattaforme, browser e dispositivi, garantendo un'esperienza utente uniforme e di alta qualità su diversi ambienti di utilizzo. Ciò è facilmente ottenibile grazie all'uso intrinseco di http2.0 di Google remote procedure call.*

A.3 Scenari d'uso dettagliati

<i>Caso d'uso</i>	<i>Visionare i voli</i>
<i>Tipo</i>	<i>Primario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail-indirizzo</i>
<i>Svolgimento Normale</i>	<i>Il cliente visualizza le informazioni dei voli (partenza-destinazione-data partenza)</i>
<i>Post condizione</i>	<i>Il cliente ha accesso alle informazioni necessarie e può procedere per la prenotazione</i>
<i>Descrizione</i>	<i>In seguito alla registrazione il cliente interagisce con una finestra che metterebbe in comunicazione con il servizio tramite un bottone, premendolo verrà aperta una nuova finestra in cui ci saranno tutte le informazioni dei vari voli</i>

<i>Caso d'uso</i>	<i>Prenotazione di un volo</i>
<i>Tipo</i>	<i>Primario</i>

<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail- indirizzo e visualizza le informazioni sui voli</i>
<i>Svolgimento Normale</i>	<i>Il cliente si prenota per un volo</i>
<i>Post condizione</i>	<i>Il cliente si prenota per un determinato volo e riceve una notifica comunicandogli il successo/fallimento della prenotazione</i>
<i>Descrizione</i>	<i>Il cliente fornisce informazione su quale volo intende prenotarsi (Partenza-Destinazione-Data tutte e tre sono dei JTextField) e preme sul Bottone, da non dimenticare che se il cliente fa parte del programma Fedeltà deve inserire anche il codice segreto in questo modo riceverà anche la notifica dell'aggiunta dei punti</i>

<i>Caso d'uso</i>	<i>Iscriversi al Programma Fedeltà</i>
<i>Tipo</i>	<i>Primario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail- indirizzo</i>
<i>Svolgimento Normale</i>	<i>Il cliente ha la possibilità di iscriversi al programma Fedeltà</i>

<i>Post condizione</i>	<i>Il Cliente ha possibilità di prenotarsi nei successivi voli e di accumulare punti per ogni prenotazione in maniera tale da ricevere sconti sui voli.</i>
<i>Descrizione</i>	<i>Il cliente si iscrive tramite un bottone e successivamente ricevere la notifica di successo con il relativo codice segreto o di fallimento</i>

<i>Caso d'uso</i>	<i>Visionare le prenotazioni effettuate</i>
<i>Tipo</i>	<i>Secondario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail-indirizzo</i>
<i>Svolgimento Normale</i>	<i>Il cliente visualizza le proprie prenotazioni</i>
<i>Post condizione</i>	<i>Il cliente ha accesso alle informazioni come la data di partenza del proprio volo</i>
<i>Descrizione</i>	<i>In seguito alla registrazione il cliente interagisce con una finestra che mettere lo metterà in comunicazione con il servizio tramite un bottone, premendolo verrà aperta una nuova finestra in cui ci saranno tutte le informazioni delle varie prenotazioni</i>

<i>Caso d'uso</i>	<i>Visionare le notifiche</i>
<i>Tipo</i>	<i>Primario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail-indirizzo</i>
<i>Svolgimento Normale</i>	<i>Il cliente visualizza le notifiche</i>
<i>Post condizione</i>	<i>Il cliente ha accesso alle informazioni sui voli o informazioni di vario genere</i>
<i>Descrizione</i>	<i>In seguito alla registrazione il cliente interagisce con una finestra che metterà in comunicazione con il servizio tramite un bottone, premendolo verrà aperta una nuova finestra in cui ci saranno informazioni generali sui voli o di vario genere</i>

<i>Caso d'uso</i>	<i>Visionare l'estratto conto</i>
<i>Tipo</i>	<i>Primario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail-indirizzo, il cliente deve essere iscritto al programma Fedeltà e dopo essersi</i>

	<i>iscritto deve aver effettuato una prenotazione</i>
<i>Svolgimento Normale</i>	<i>Il cliente visualizza l'estratto conto</i>
<i>Post condizione</i>	<i>Il cliente ha accesso alle informazioni come i punti che ha accumulato per ogni volo</i>
<i>Descrizione</i>	<i>In seguito alla registrazione il cliente interagisce con una finestra che mettere lo metterà in comunicazione con il servizio tramite un bottone, premendolo verrà aperta una nuova finestra in cui ci saranno tutte le informazioni come, ad esempio, il numeri di punti che ha accumulato per ogni volo prenotato</i>

<i>Caso d'uso</i>	<i>Modificare un volo</i>
<i>Tipo</i>	<i>Primario</i>
<i>Precondizione</i>	<i>Il cliente accede all'applicazione identificandosi tramite nome-cognome-codice fiscale – data di nascita- e-mail-indirizzo, il cliente deve aver già effettuato la prenotazione per un volo</i>
<i>Svolgimento Normale</i>	<i>Il cliente modifica la data di partenza per un determinato volo</i>
<i>Post condizione</i>	<i>Il cliente ha modificato la data di partenza per un altro volo, ovvero vede se c'è</i>

	<i>disponibile un volo che fa la stessa tratta con data diversa (cioè, la data desiderata)</i>
<i>Descrizione</i>	<i>Il cliente Inserisci le informazioni del relativo volo con la nuova data e riceve un riscontro di successo o di fallimento</i>

A.4 Excluded Requirements

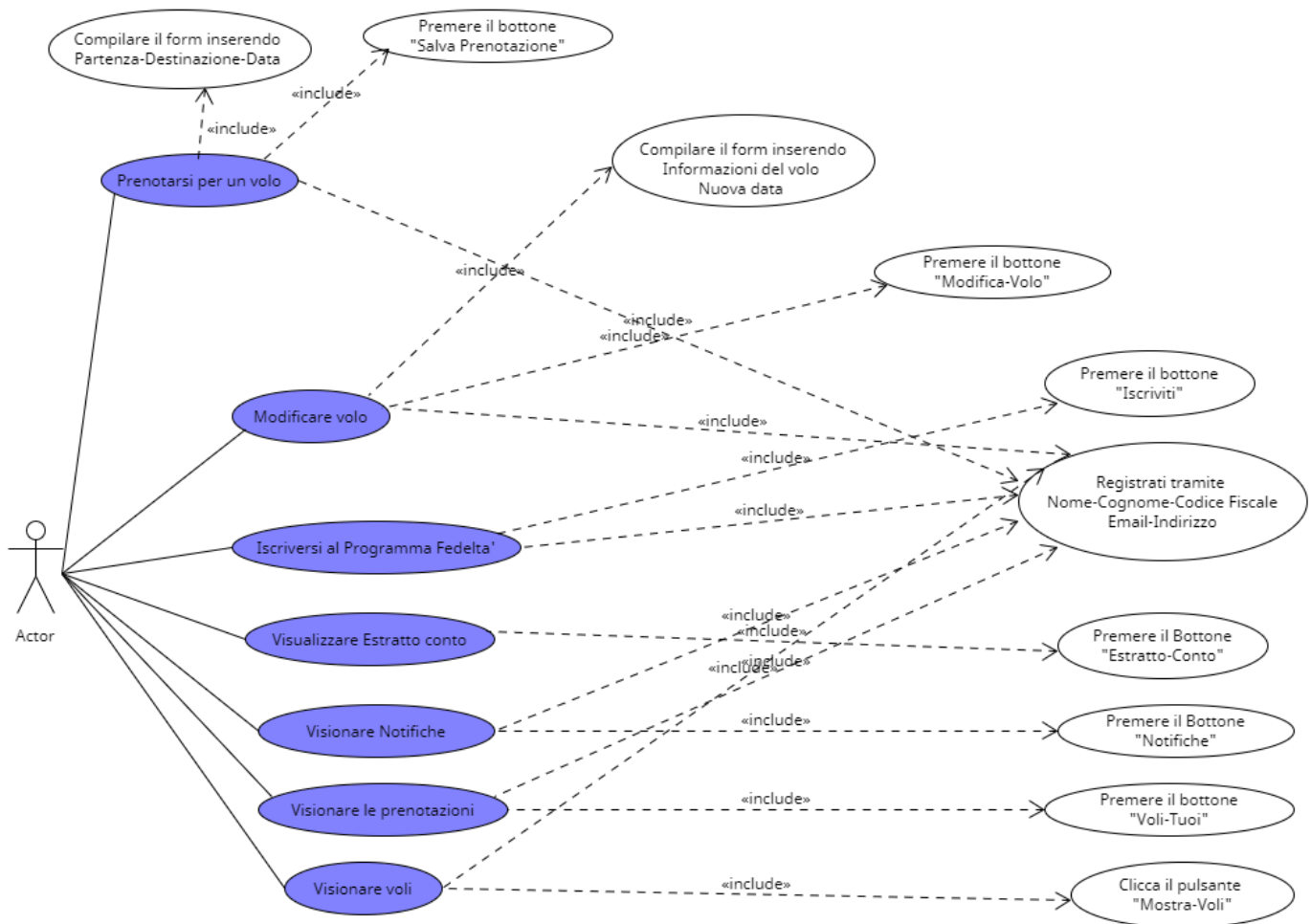
Durante la progettazione dell'applicazione sono emerse funzionalità che avrebbero potuto migliorare la qualità di essa stessa come ad esempio:

- 1)Possibilità di cancellare un volo*
 - 2)Possibilità di rimuovere voli lato server*
 - 3)Possibilità di inserire sistemi di sicurezza per tutelare i clienti*
 - 4)Possibilità di recuperare il codice segreto in caso di smarrimento*
-

A.5 Assunzioni

Si assuma che il cliente non provi a registrarsi tramite codice fiscale, email , indirizzo , nome e cognome di altri clienti.

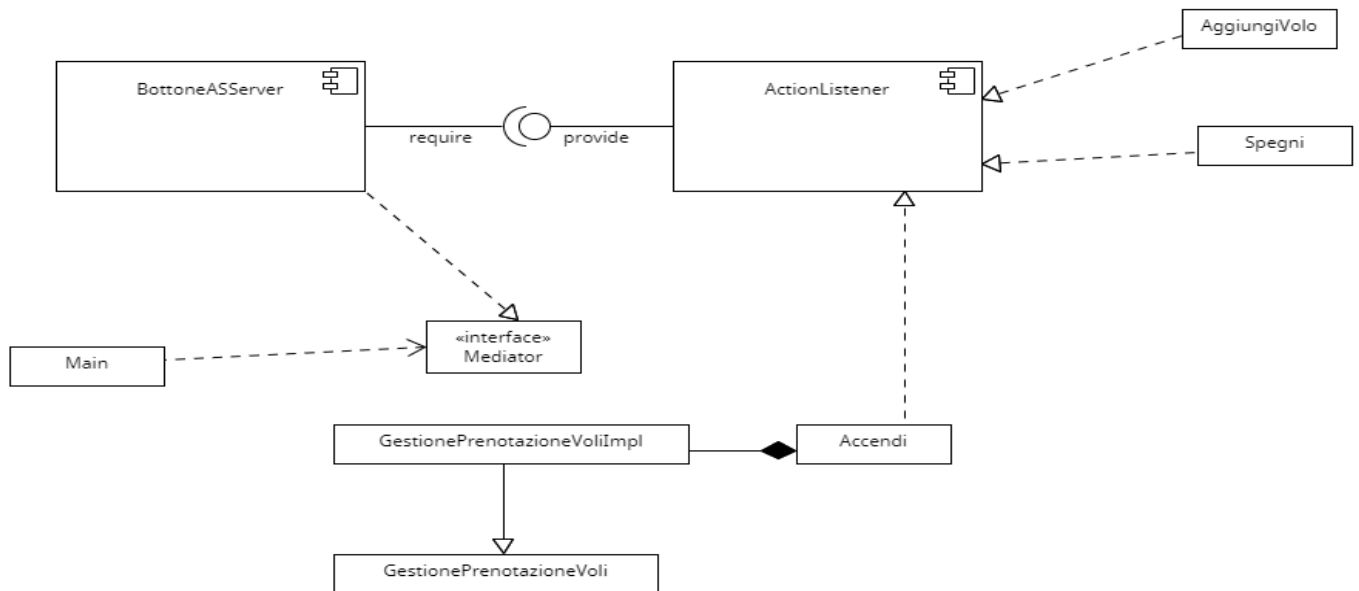
A.6 Use Case Diagrams



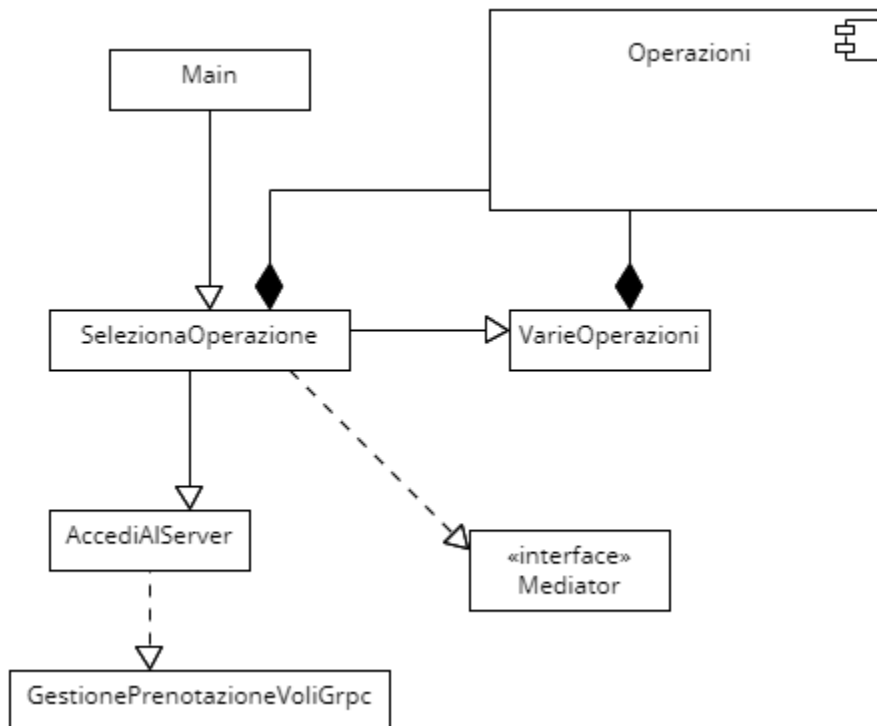
C. Architettura Software

C.1 The static view of the system: Component Diagram

Component Diagram Lato Server



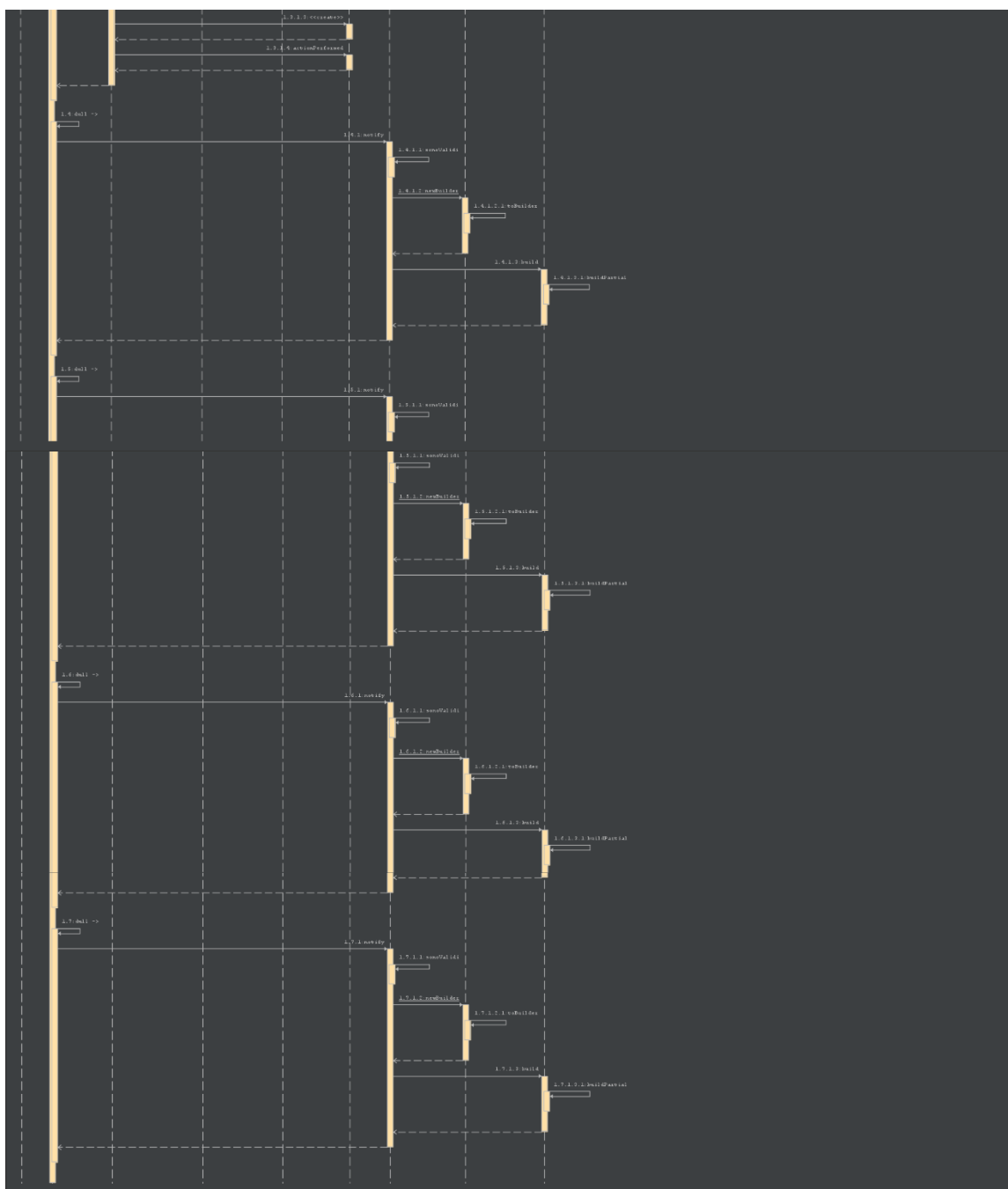
Component Diagram Lato Client



C.2 The dynamic view of the software architecture: Sequence Diagram

Server





D. Dati e loro modellazione (se il sistema si interfaccia con un DBMS)

L'applicazione offre un meccanismo di persistenza dei dati tramite una serie di file json. L'applicazione server fornisce un'interfaccia grafica per aggiungere Voli e dunque serializzarli in degli appositi file json.

E. Scelte Progettuali (Design Decisions)

Il progetto è stato realizzato utilizzando i seguenti design pattern:

1) Mediator

2) Command

3) Observer

4) Proxy (fornito nativamente da GRPC)

5) Builder (fornito nativamente da GRPC)

Il pattern più utilizzato è stato il Mediator per poter permettere una facile gestione dell'interfaccia grafica

F. Progettazione di Basso Livello

Mediator:

Si è utilizzato molto il design pattern Mediator così da mediare e disaccoppiare le interazioni tra i vari componenti grafici, in particolare, è stato molto utilizzato sia lato client che lato server.

Proxy:

Il pattern proxy è stato fornito nativamente da GRPC, in particolare, lato client e lato server. è possibile utilizzare un'istanza di un oggetto di tipo BlockingStub che permette al client di effettuare richieste remote al server, in particolare richieste ai metodi esposti dal server.

Command/Observer:

Inizialmente si è pensato di introdurre degli oggetti che seguissero la filosofia dei design pattern Command, così da poter prima sviluppare la logica del programma su cui poi mappare una interfaccia grafica. In fase di sviluppo, non avendo bisogno di un command handler per introdurre politiche già gestite dai vari mediator, si sono attribuite caratteristiche quasi da Observer a tali oggetti, così da renderli compatibili con i componenti JButton; infatti, è possibile notare come tali oggetti implementino la classe ActionListener.

Builder:

Il pattern Builder e' utilizzato nativamente da GRPC, in particolare mi consente di creare oggetti complessi questo però quando, per creare un oggetto ci sono più configurazioni possibili, quindi senza utilizzare un costruttore con molti parametri (parametri che avvolte possono essere nulle cc...)

G. Spiegare come il progetto soddisfa i requisiti funzionali (FRs) e quelli non funzionali (NFRs)

Requisiti Funzionali

L'Applicazione soddisfa i requisiti funzionali nel seguente modo:

Visionare i voli: Il sistema permette ai client di visionare le informazioni dei voli come la partenza la destinazione e la data, In pratica questo requisito viene soddisfatto fornendo al client la possibilità di richiedere e recuperare le informazioni necessarie che saranno successivamente visualizzate su una tabella di un'altra finestra.

Prenotare un volo: Dopo che un client ha visualizzato le informazioni necessarie può prenotare un volo, la prenotazione del volo si suddivide in due casistiche: Clienti iscritti al programma Fedeltà e non, coloro che sono iscritti al programma Fedeltà oltre che inserire nei vari JTextField la partenza, la destinazione e la data del volo devono inserire anche il codice Segreto. Questo requisito è soddisfatto dal server che mette a disposizione la funzionalità per memorizzare una prenotazione e ritorna l'esito al client, Inoltre tiene conto se un client ha effettuato già una prenotazione per quel volo

Iscrizione al Programma “Cliente Fedeltà”: Il sistema permette ai client di iscriversi al programma Fedeltà, questa funzionalità è garantita lato server che mette a disposizione il servizio, garantendo che il client sappia se l'iscrizione e' andata a buon fine oppure no e generando un codice segreto unico inoltre controlla se il client non si sia già iscritto.

Visionare le prenotazioni: Il sistema permette ai client di visionare le prenotazioni effettuate come la partenza la destinazione e la data, In pratica

questo requisito viene soddisfatto fornendo al client la possibilità di richiedere e recuperare le informazioni necessarie che saranno successivamente visualizzate su una tabella di un'altra finestra.

Visionare le notifiche: *Il sistema permette ai client di visionare notifiche da parte del server, questa funzionalità è garantita lato server permettendo al client di richiedere le notifiche principali dal server che verranno visualizzate su un'altra finestra.*

Visionare l'estratto conto: *Il sistema permette ai client di visionare l'estratto conto, ovvero i punti accumulati per ogni volo prenotato, naturalmente questa funzionalità vale se il cliente fa parte del Programma Fedeltà e se ha effettuato prenotazioni usando il codice segreto. In pratica questo requisito viene soddisfatto fornendo al client la possibilità di richiedere e recuperare le informazioni necessarie che saranno successivamente visualizzate su una tabella di un'altra finestra.*

Modificare un volo: *Il sistema permette ai client di modificare il proprio volo, in sostanza il client inserisce su un form le informazioni del suo volo (partenza-destinazione-data) e inserisce una nuova data, il server soddisfa questo requisito mettendo a disposizione il servizio che si occupa di trovare un volo che fa la stessa tratta ma con data di partenza pari a quella inserita dal client.*

Requisiti non Funzionali

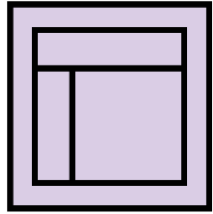
L'Applicazione soddisfa i requisiti non funzionali nel seguente modo:

Persistenza delle informazioni: *Il sistema permette di memorizzare informazioni sui clienti, voli e prenotazioni, questo dà al sistema una grande affidabilità in quanto garantisce che il sistema funziona anche al riavvio del server. Il requisito è soddisfatto tramite meccanismi di persistenza sul server.*

GUI: *Il sistema è dotato di una GUI minima per permette sia ai client sia ai vari amministrato di interagire con l'applicazione in maniera semplice e intuitiva, il requisito e' soddisfatto sia lato client che lato server.*

Identificazione: *Il sistema permette ai client di identificarsi in maniera tale da garantire autenticità e integrità della prenotazione, questo requisito è soddisfatto su lato client attraverso l'implementazione di un meccanismo d'identificazione.*

Appendix. Prototype



GUI lato server

The screenshot shows a window titled 'Server-Voli' with three buttons at the top: 'Accendi-Server' (highlighted in blue), 'Spegni-Server', and 'Aggiungi Volo'. On the left, there are three labels: 'Inserisci partenza:', 'Inserisci destinazione:', and 'Inserisci data:'. To the right of these labels are three empty text input fields stacked vertically.

La GUI lato server è composta da 3 bottoni principali e da un form, Accendi-Server utilizzato per accendere il server. Spegni-Server utilizzato per spegnere il server. Aggiungi-Volo viene utilizzato per aggiungere un volo dopo aver compilato il form.

The screenshot shows the same 'Server-Voli' window. The 'Accendi-Server' button is no longer highlighted. The input fields now contain sample data: 'Roma' for the departure, 'Milano' for the destination, and '20-12-2023 10:00' for the date and time.

GUI lato Client

Per prima cosa ci verrà mostrato un form dove inserire le nostre credenziali

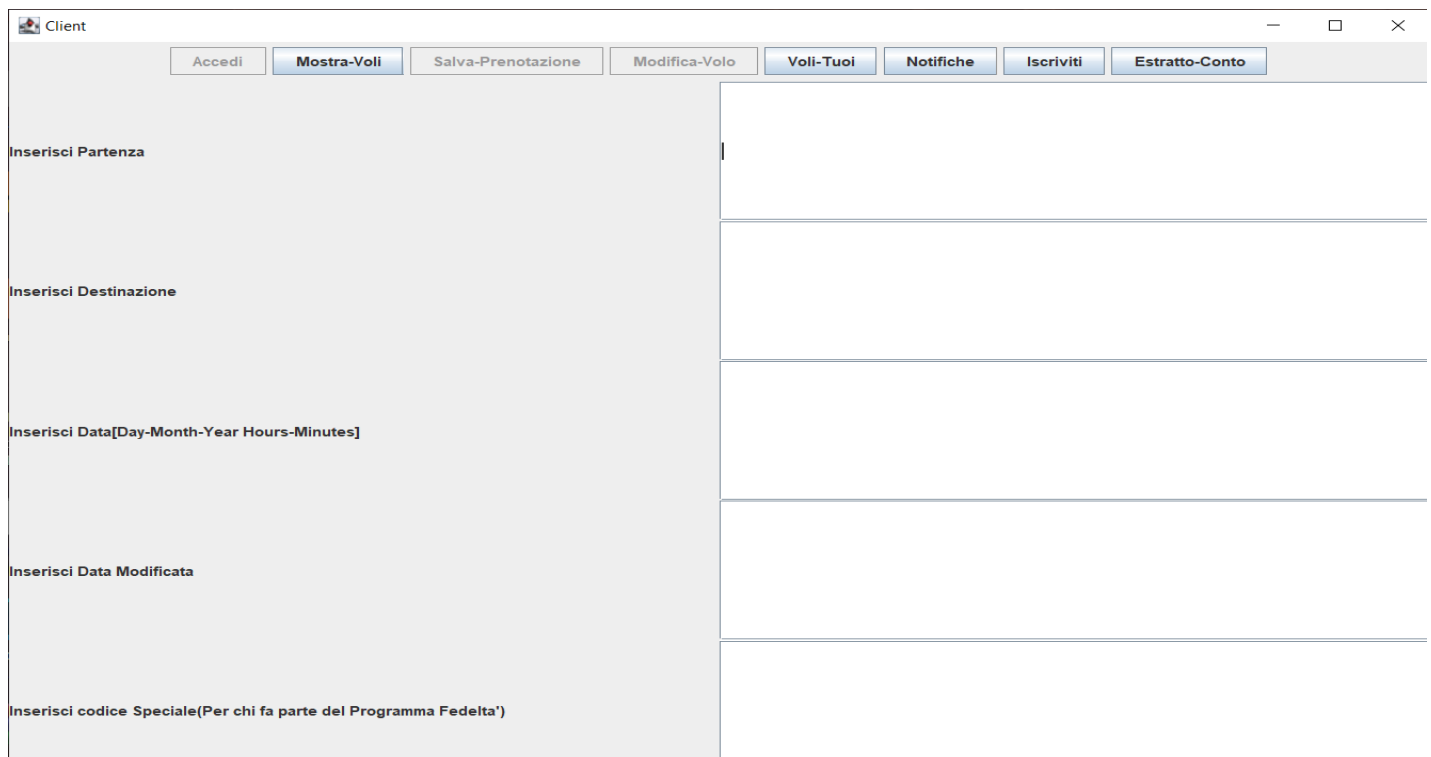


The screenshot shows a window titled "Client-Login" with a standard Windows title bar (minimize, maximize, close buttons). The window contains a form with the following fields and values:

Label	Value
Inserisci nome	Constantin Adrian
Inserisci cognome	Antoci
Inserisci data di Nascita [gg-MM-AA]	23-05-2001
Inserisci codiceFiscale	ntccst01e23z129p
Inserisci indirizzo	via milano
Inserisci indirizzo-Email	antociconstantinadrian@yahoo.com

Below the form is a large button labeled "Accedi".

Dopo aver effettuato l'accesso si aprirà un'altra finestra ovvero la finestra in cui il client può usufruire sei servizi di prenotazione ecc...



The screenshot shows a window titled "Client" with a standard Windows title bar. The window contains a horizontal bar with eight buttons: "Accedi", "Mostra-Voli", "Salva-Prenotazione", "Modifica-Volo", "Voli-Tuoi", "Notifiche", "Iscriviti", and "Estratto-Conto". Below this bar is a form with the following fields and labels:

Label	Value
Inserisci Partenza	
Inserisci Destinazione	
Inserisci Data[Day-Month-Year Hours-Minutes]	
Inserisci Data Modificata	
Inserisci codice Speciale(Per chi fa parte del Programma Fedelta')	

Come possiamo vedere la finestra è formata da 8 bottoni ognuno con una specifica funzionalità:

1) **Accedi**: permette al client di accedere al server.

- 2) **Mostra-Voli:** Permette al client di visionare i voli
- 3) **Salva-Prenotazione:** Dopo aver compilato il form permette di salvare la prenotazione
- 4) **Modifica-Volo:** Dopo aver compilato il form permette al client di modificare il suo volo.
- 5) **Voli-Tuoi:** Mostra le prenotazioni effettuate
- 6) **Notifiche:** Permette al client di visionare le notifiche da parte del server
- 7) **Iscriviti:** Permette al client di iscriversi al Programma Fedeltà
- 8) **Estratto Conto:** Permette al client di visionare il proprio estratto conto

Ecco ciò che viene mostrato premendo il bottone Mostra-Voli:

Partenza	Destinazione	Data
Roma	Milano	28/09/2023 10:00
Lamezia	Milano	22/09/2023 10:00
Lamezia	Roma	25/09/2023 10:00

Ecco ciò che viene mostrato premendo il bottone Iscriviti:

Client

AccediMostra-VoliSalva-PrenotazioneModifica-VoloVoli-TuoiNotificheIscriviti

Inserisci Partenza

Inserisci Destinazione

Iscrizione avvenuta con successo, ecco il codice214174

Per prenotare il volo non si deve necessariamente far parte del Programma Fedeltà, ho deciso di mostrare prima L'iscrizione al programma di Fedeltà perché' successivamente mostrerò la prenotazione a un volo. Dopo essersi iscritti non c'è la possibilità di iscriversi nuovamente.

Errore oppure sei già iscritto

```
{
  "dati_": [
    {
      "cliente_": {
        "nome_": "Constantin Adrian",
        "cognome_": "Antoci",
        "codiceFiscale_": "ntccst01e23z129p",
        "indirizzo_": "via milano",
        "dataDiNascita_": {
          "seconds_": 990568800,
          "nanos_": 0,
          "memoizedIsInitialized": -1,
          "unknownFields": {
            "fields": {}
          },
          "memoizedSize": -1,
          "memoizedHashCode": 0
        },
        "indirizzoEmail_": "antociconstantinadrian@yahoo.com",
        "memoizedIsInitialized": 1,
        "unknownFields": {
          "fields": {}
        },
        "memoizedSize": -1,
        "memoizedHashCode": 58354001
      },
      "codiceSegreto_": 214174,
      "estratti_": {
        "estrattoConto_": [],
        "memoizedIsInitialized": 1,

```

Questa e' una parte del file Json che memorizza i clienti Fedeltà , come possiamo vedere memorizza il cliente, il codice segreto e i suoi "estratti" ovvero tutti i voli che ha effettuato in cui gli sono stati assegnati dei punti.

Ecco ciò che viene mostrato premendo il bottone Salva-Prenotazione:

The screenshot shows the 'Salva-Prenotazione' (Save Booking) form in the 'Client' application. The form has a header bar with buttons: 'Accedi', 'Mostra-Voli', 'Salva-Prenotazione' (highlighted), 'Modifica-Volo', 'Voli-Tuoi', 'Notifiche', 'Iscriviti', and 'Estratto-Conto'. The form fields are as follows:

- Inserisci Partenza:** Lamezia
- Inserisci Destinazione:** Roma
- Inserisci Data[Day-Month-Year Hours-Minutes]:** 25-09-2023 10:00
- Inserisci Data Modificata:** (empty)
- Inserisci codice Speciale(Per chi fa parte del Programma Fedelta'):** 214174

Per far sì che il bottone Salva-Prenotazione sia visibile bisogna prima compilare il Form

This screenshot shows the same 'Salva-Prenotazione' form, but with two success message windows displayed:

- A small window titled 'Successo punti aggiunti' (Success points added) is shown over the 'Inserisci Partenza' field.
- A larger window titled 'Prenotazione avvenuta con successo' (Booking successful) is shown over the 'Inserisci Destinazione' field.

The form fields contain the same data as the previous screenshot: Lamezia, Roma, 25-09-2023 10:00, and 214174.

*Dopo aver effettuato la prenotazione
Ci viene comunicato che sono stati aggiunti punti e che la prenotazione ha avuto successo*

Ecco ciò che viene mostrato premendo il bottone Voli-Tuoi:

Voli-Miei					
Partenza	Destinazione	Data			
Lamezia	Roma	25/09/2023 10:00			

Ecco ciò che viene mostrato premendo il bottone Estratto-Conto:

Estratto-Conto					
Partenza	Destinazione	Data		Punti-Accumulati	
Lamezia	Roma	25/09/2023 10:00		32	Constantin Adrian (antociconstantinadri)

Mostra per ogni volo i punti che abbiamo accumulato.

Ecco ciò che viene mostrato premendo il bottone Modifica-Volo:

Client

Accedi

Mostra-Voli

Salva-Prenotazione

Modifica-Volo

Voli-Tuoi

Notifiche

Iscriviti

Estratto-Conto

Inserisci Partenza

Lamezia

Inserisci Destinazione

Roma

Inserisci Data[Day-Month-Year Hours-Minutes]

25-09-2023 10:00

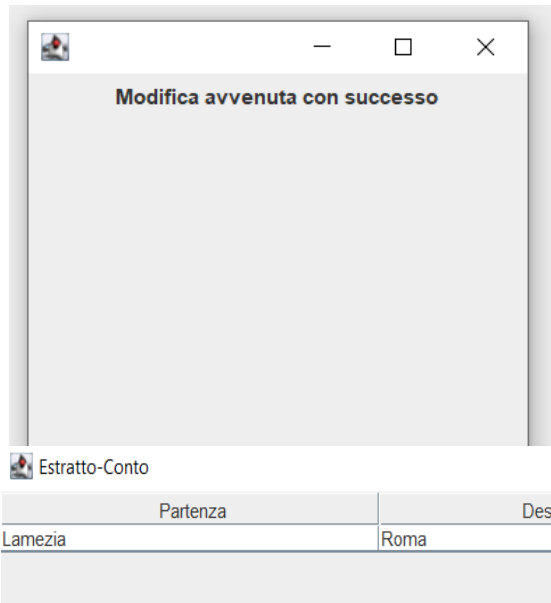
Inserisci Data Modificata

22-09-2023 10:00

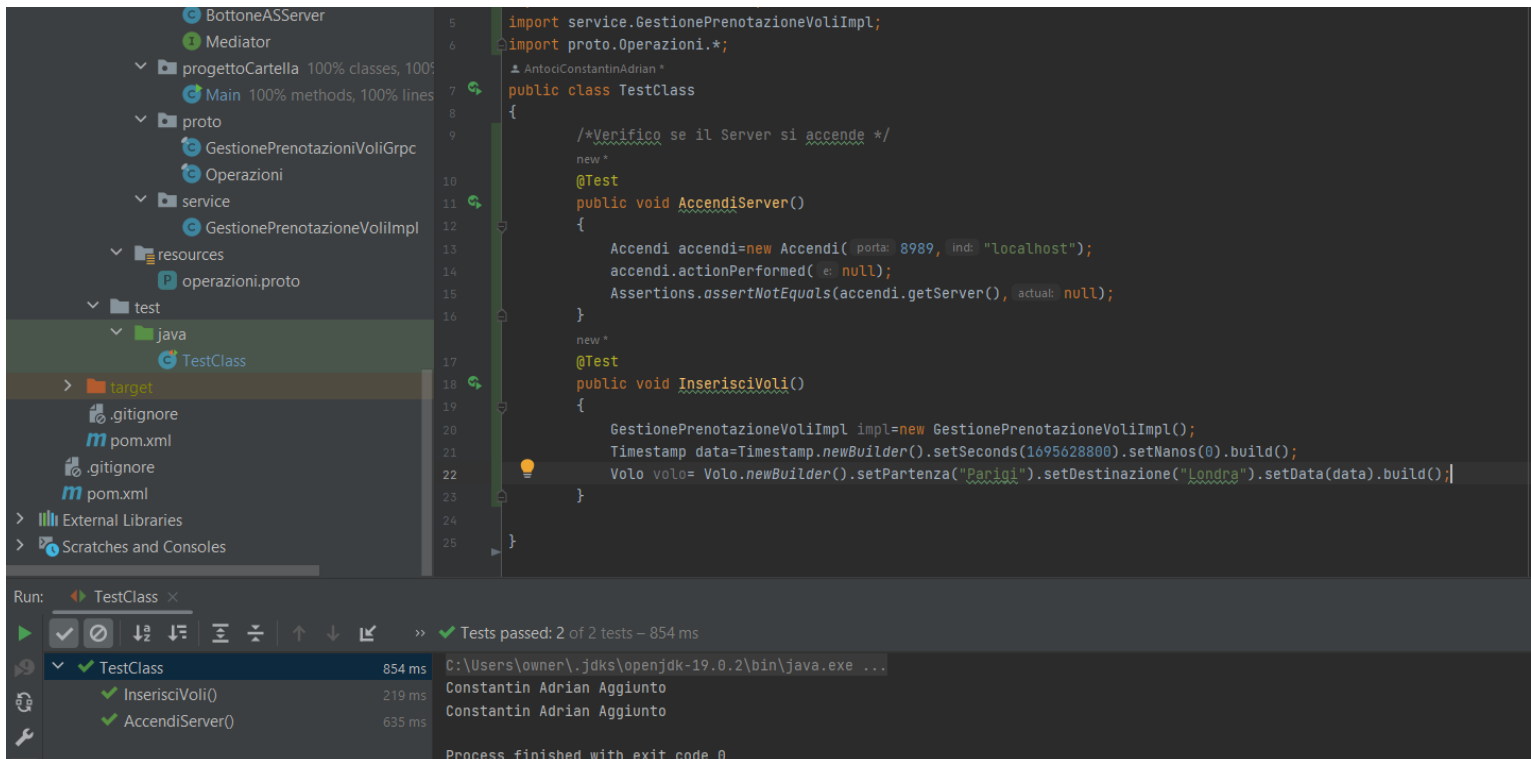
Inserisci codice Speciale(Per chi fa parte del Programma Fedelta')

Si inseriscono i dati del volo precedente più la nuova data e il server va a controllare se e' presente un volo con tale data e inserisce il cliente nel nuovo volo.

Dopo aver modificato ci viene mostrata una notifica e viene modificato anche L'Estratto Conto.



TEST LATO SERVER



TEST LATO CLIENT

The screenshot shows the IntelliJ IDEA IDE with a project named 'ProgettoIngegneriaAntoci'. The left sidebar displays the project structure, including a 'test' directory with a 'TestClass' file. The main editor shows the code for 'TestClass.java', which contains two test methods: 'testAperturaSocket()' and 'testPrenotazione()'. The 'Run' tab at the bottom shows the test results: 'Tests passed: 2 of 2 tests - 843 ms'. The 'TestClass' entry is expanded, showing 'testPrenotazione()' (840 ms) and 'testAperturaSocket()' (3 ms). The console output indicates 'Prenotazione avvenuta con successo' and 'Process finished with exit code 0'.

```
8  @Test
9  public void testAperturaSocket()
10 {
11     AccediAlServer server=new AccediAlServer( porta: 8989, ind: "localhost");
12     server.actionPerformed( e: null);
13     Assertions.assertNotEquals(server, actual: null);
14 }
15
16  @Test
17  public void testPrenotazione()
18 {
19     AccediAlServer server=new AccediAlServer( porta: 8989, ind: "127.0.0.1");
20     server.actionPerformed( e: null);
21     VarieOperazioni varie=new VarieOperazioni(server.getClientStub());
22     /*Inserisco data giusta*/
23     Timestamp time=Timestamp.newBuilder().setSeconds(1695628800).setNanos(0).build();
24     Timestamp dataNascita=Timestamp.newBuilder().setNanos(0).setSeconds(990568800).build();
25     VoLo volo= VoLo.newBuilder().setPartenza("Lamezia").setDestinazione("Roma").setData(time).build();
26     Cliente cliente= Cliente.newBuilder()
27         .setName("Luca")
28         .setCognome("Rossi")
29         .setIndirizzo("Via bo").setIndirizzoEmail("luca@gmail.com").setCodiceFiscale("aa").setDataDiNascita(dataNascita).build();
30     Boolean vedi=varie.prenotaVolo(cliente, volo);
31     Assertions.assertEquals(vedi, actual: true);
32 }
```

Run: Main x TestClass x

Tests passed: 2 of 2 tests - 843 ms

TestClass 843 ms

testPrenotazione() 840 ms

testAperturaSocket() 3 ms

C:\Users\owner\.jdk\openjdk-19.0.2\bin\java.exe ...

Prenotazione avvenuta con successo

Process finished with exit code 0

Ho effettuato dei test sia lato client sia lato server per controllare il giusto funzionamento dell'applicazione, come possiamo vedere, sia lato client sia lato server i test sono andati a buon fine, tuttavia, ho voluto effettuare un ulteriore test lato Client per verificare se inserendo un volo con data sbagliato (Quindi un volo non presente nell'elenco dei voli).

The screenshot shows the IntelliJ IDEA IDE with the same project. The 'Run' tab now shows 'Tests failed: 1, passed: 1 of 2 tests - 904 ms'. The 'TestClass' entry is expanded, showing 'testPrenotazione()' (900 ms) with a red 'X' icon and 'testAperturaSocket()' (4 ms) with a green checkmark. The console output indicates 'Prenotazione non avvenuta con successo' and an 'org.opentest4j.AssertionFailedError: Expected :false Actual :true'.

```
15  @Test
16  public void testPrenotazione()
17 {
18     AccediAlServer server=new AccediAlServer( porta: 8989, ind: "127.0.0.1");
19     server.actionPerformed( e: null);
20     VarieOperazioni varie=new VarieOperazioni(server.getClientStub());
21     /*Inserisco data giusta*/
22     Timestamp time=Timestamp.newBuilder().setSeconds(1695628800).setNanos(0).build();
23     Timestamp dataNascita=Timestamp.newBuilder().setNanos(0).setSeconds(990568800).build();
24     VoLo volo= VoLo.newBuilder().setPartenza("Lamezia").setDestinazione("Roma").setData(time).build();
25     Cliente cliente= Cliente.newBuilder()
26         .setName("Luca")
27         .setCognome("Rossi")
28         .setIndirizzo("Via bo").setIndirizzoEmail("luca@gmail.com").setCodiceFiscale("aa").setDataDiNascita(dataNascita).build();
29     Boolean vedi=varie.prenotaVolo(cliente, volo);
30     Assertions.assertEquals(vedi, actual: true);
31 }
32
33
34
35 }
```

Run: Main x TestClass x

Tests failed: 1, passed: 1 of 2 tests - 904 ms

TestClass 904 ms

testPrenotazione() 900 ms

testAperturaSocket() 4 ms

C:\Users\owner\.jdk\openjdk-19.0.2\bin\java.exe ...

Prenotazione non avvenuta con successo

org.opentest4j.AssertionFailedError:
Expected :false
Actual :true
<Click to see difference>

Come possiamo vedere il test ha funzionato (Questo perché la prenotazione di un volo inesistente non può avvenire).

Si noti che per il corretto funzionamento dell'applicazione bisogna modificare sia lato server che lato client nella classe Operazioni, all'interno della classe Cliente il metodo hashCode come segue:

```
@java.lang.Override
public int hashCode() {
    int hash=41;
    hash=(53*hash)+getCodiceFiscale().hashCode();
    memoizedHashCode=hash;
    return hash;
}
```