

Università degli Studi Milano - Bicocca
Relazione Progetto Aereo 4

Federico Piscitelli - 829666
Davide Rendina - 830730
Omar Zaher - 829701

Milano, 15 Marzo 2020

Indice

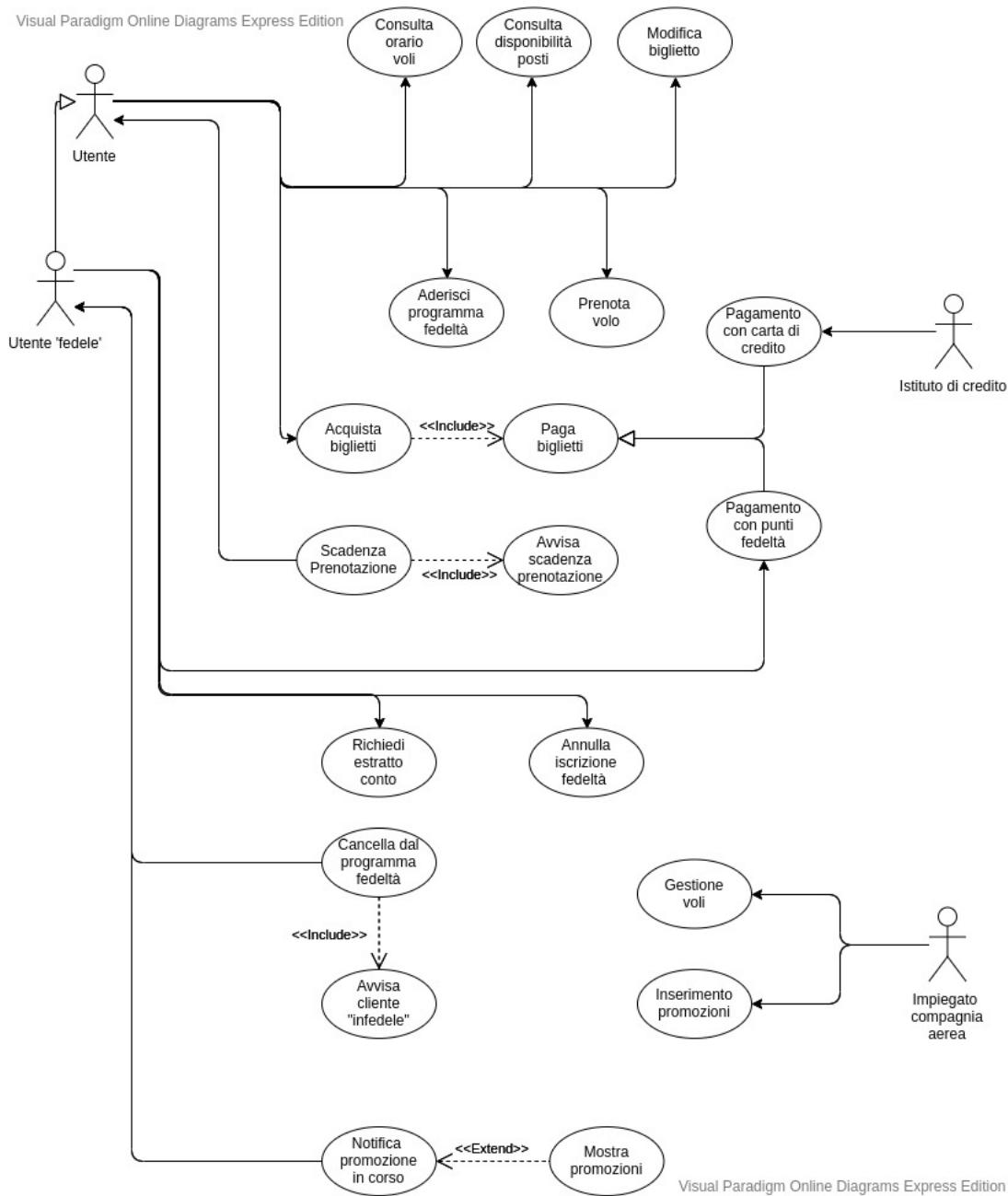
1	Introduzione	3
2	Analisi e progettazione	3
2.1	Casi d'uso	3
2.2	Modello di dominio	4
2.3	Diagrammi delle attività	5
2.3.1	Attività dei clienti	5
2.3.2	Attività per la gestione dei voli	7
2.3.3	Attività di prenotazione biglietti	9
2.3.4	Attività di acquisto biglietti	10
2.4	Diagramma di deployment	13
2.5	Architettura Software	14
2.6	Classi di progetto	16
2.6.1	Gestione dei clienti	18
2.6.2	Gestione dei voli	22
2.6.3	Prenotazione	24
2.6.4	Acquisto di una prenotazione effettuata	26
2.6.5	Cambio data di una prenotazione	29
2.6.6	Funzionamento DBFacade	33
2.6.7	Gestione Promozioni	35
2.6.8	Generazione codici OID	35
2.7	Stati degli oggetti	36
2.8	Design Pattern considerati nello sviluppo	37
3	Note tecniche	38
3.1	Database	38
3.2	Cronjobs	39
4	Funzionalità	40
4.1	Per cominciare	40
4.2	Ricerca voli	40
4.3	Consultazione voli	40
4.4	Prenotazione volo	40
4.5	Acquisto	42
4.6	Cambia data	44
4.7	Area fedeltà	44
4.8	Backend	45
5	Analisi del codice attraverso i tool	46
5.1	Analisi SonarCloud	46
5.2	Analisi Understand	47
6	Conclusioni	49

1 Introduzione

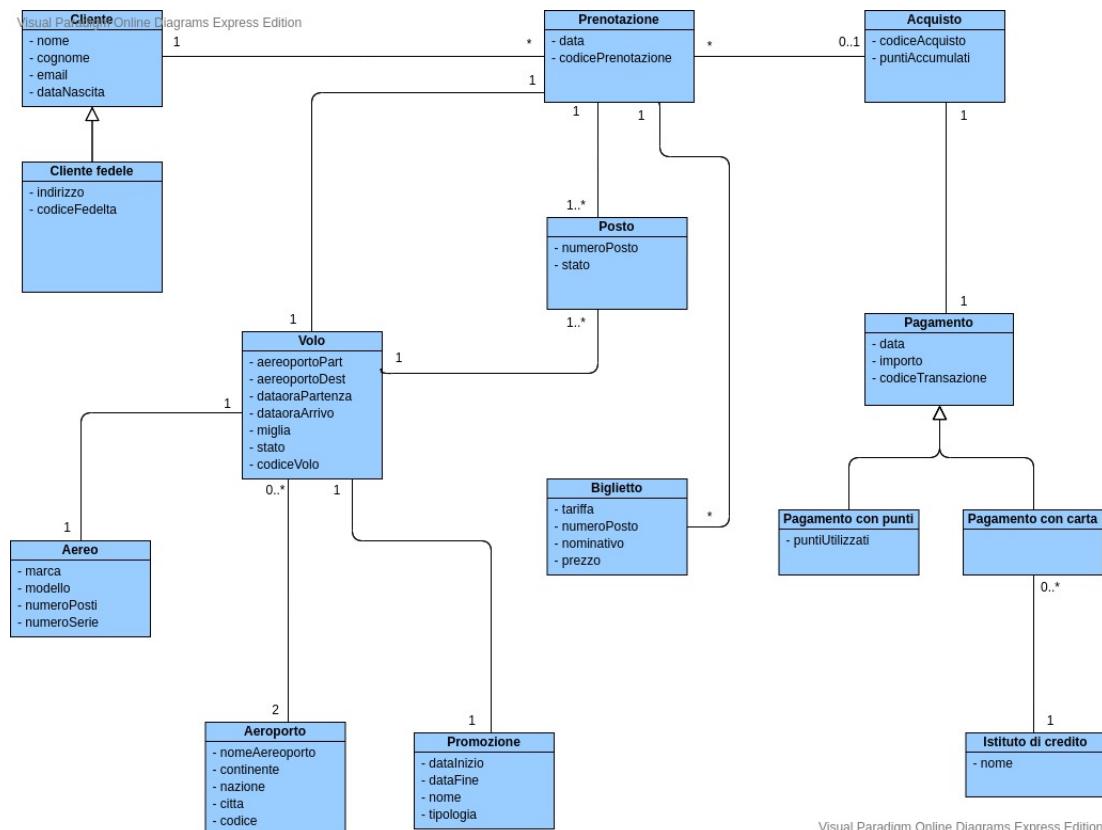
Progetto Aereo 4 ha come scopo quello di fornire all'utente finale un prodotto in grado di soddisfare le sue esigenze da passeggero. Il sito dà la possibilità al fruitore di ricercare voli, prenotare e acquistare biglietti, usufruire di promozioni a tempo o sul volo. Per i clienti che lo vorranno il sito mette a disposizione un programma fedeltà al quale l'utilizzatore potrà registrarsi fornendo i suoi dati personali, un'email e una password. Una volta che l'utente è in possesso di un account, potrà accedere alla sua area privata dove potrà gestire in tutta semplicità le prenotazioni, i suoi acquisti e potrà richiedere l'estratto conto dei punti associati al suo profilo. Il vantaggio di essere un cliente fedeltà è quello di poter accumulare punti ad ogni acquisto e, quando i punti raccolti saranno sufficienti, l'utente potrà pagare con i suoi punti al posto che col denaro.

2 Analisi e progettazione

2.1 Casi d'uso



2.2 Modello di dominio



Visual Paradigm Online Diagrams Express Edition

2.3 Diagrammi delle attività

2.3.1 Attività dei clienti

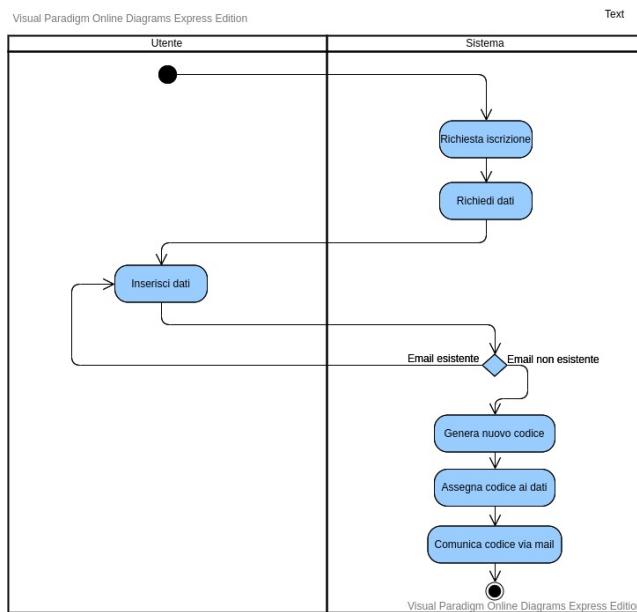


Figure 1: Registrazione programma fedelta

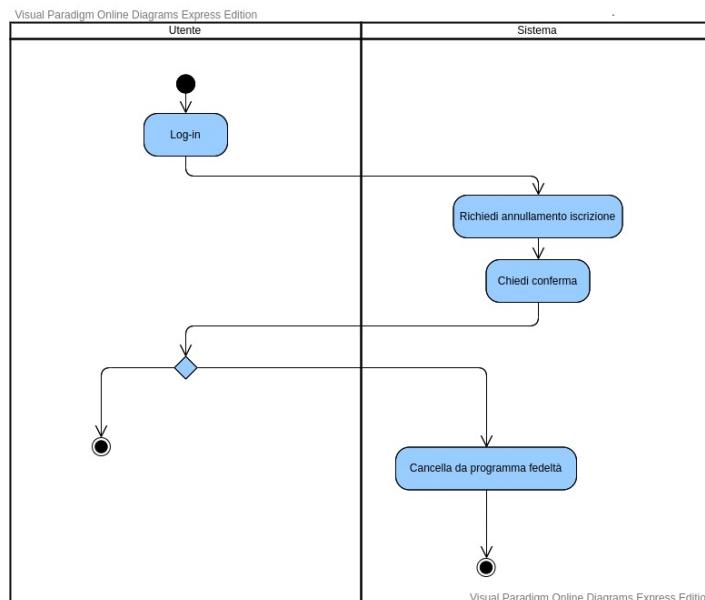


Figure 2: Annullamento iscrizione fedelta

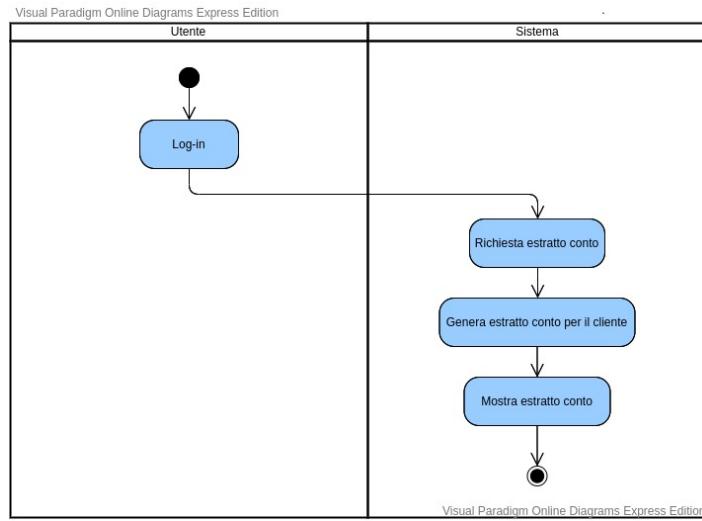


Figure 3: Richiesta estratto conto

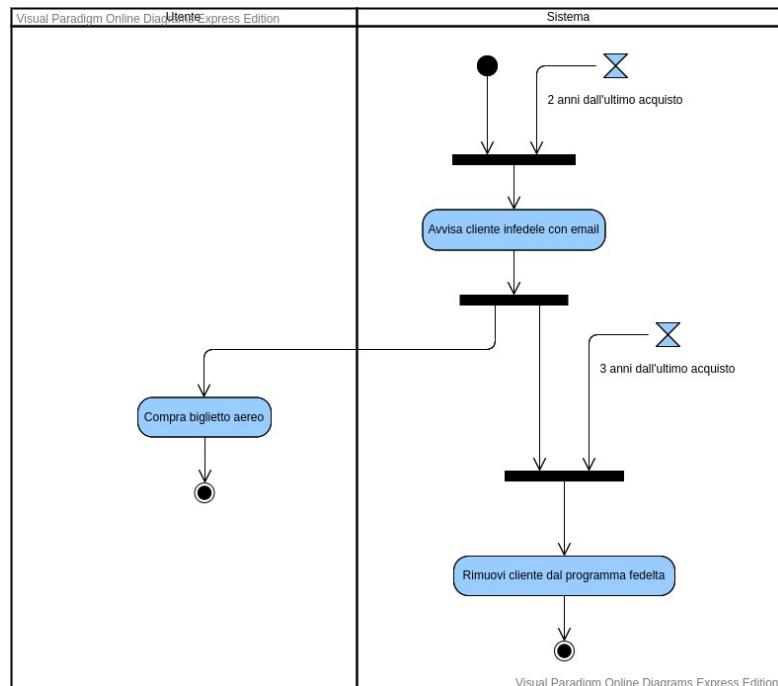


Figure 4: Gestione clienti infedeli

2.3.2 Attività per la gestione dei voli

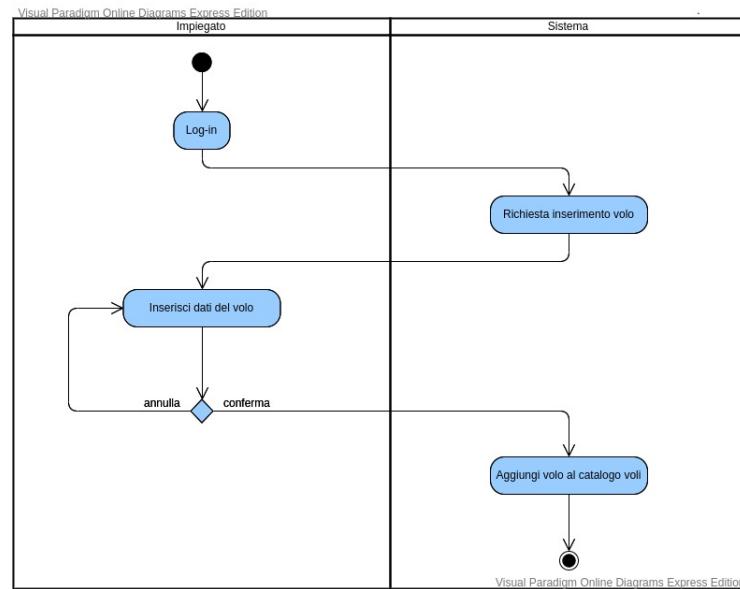


Figure 5: Inserimento nuovo volo

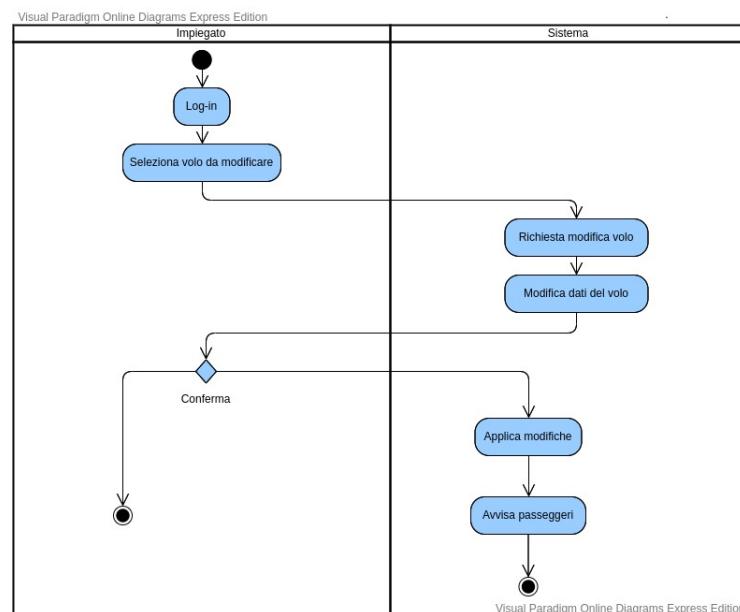


Figure 6: Modifica volo

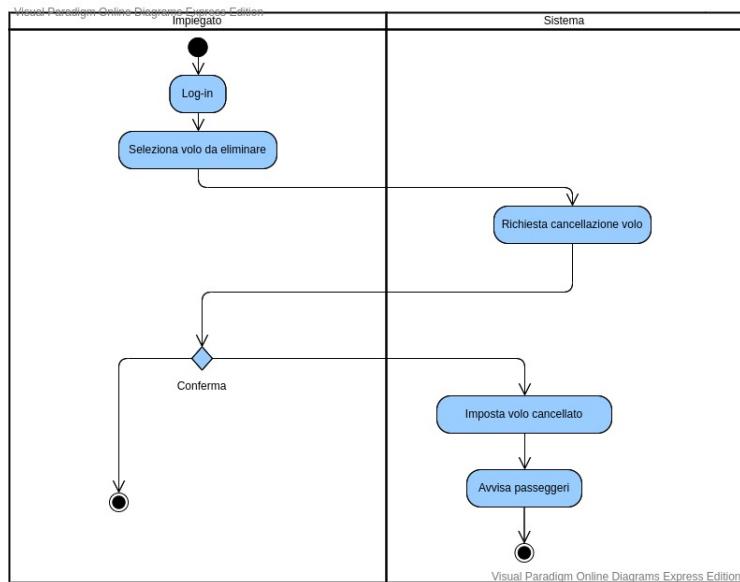


Figure 7: Cancellazione volo

2.3.3 Attività di prenotazione biglietti

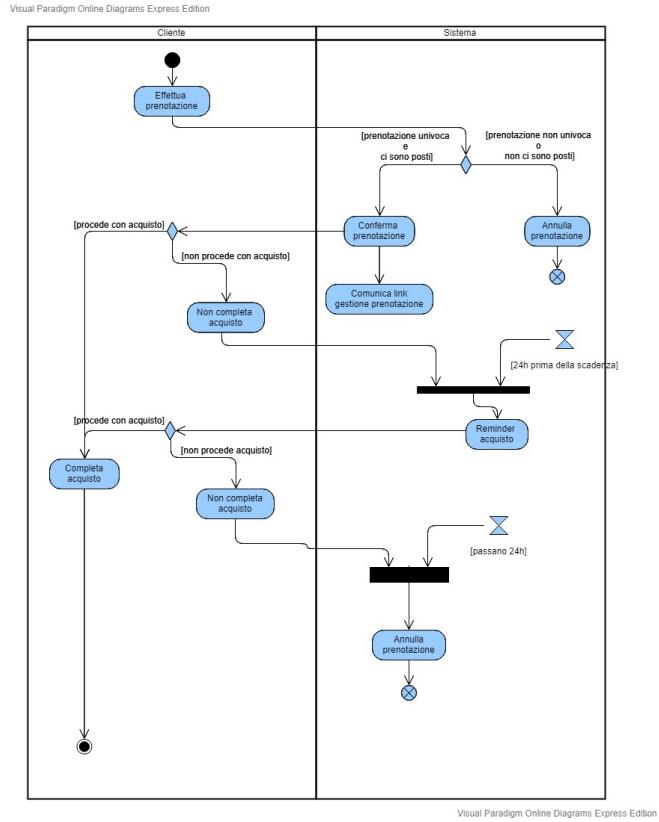


Figure 8: Effettua prenotazione

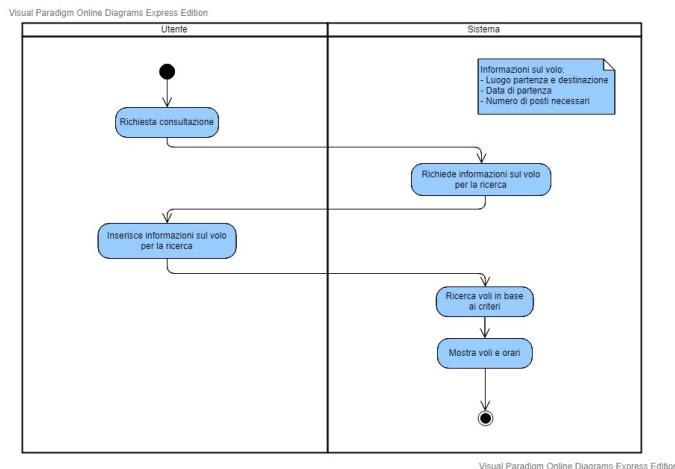


Figure 9: Consultazione dei voli

2.3.4 Attività di acquisto biglietti

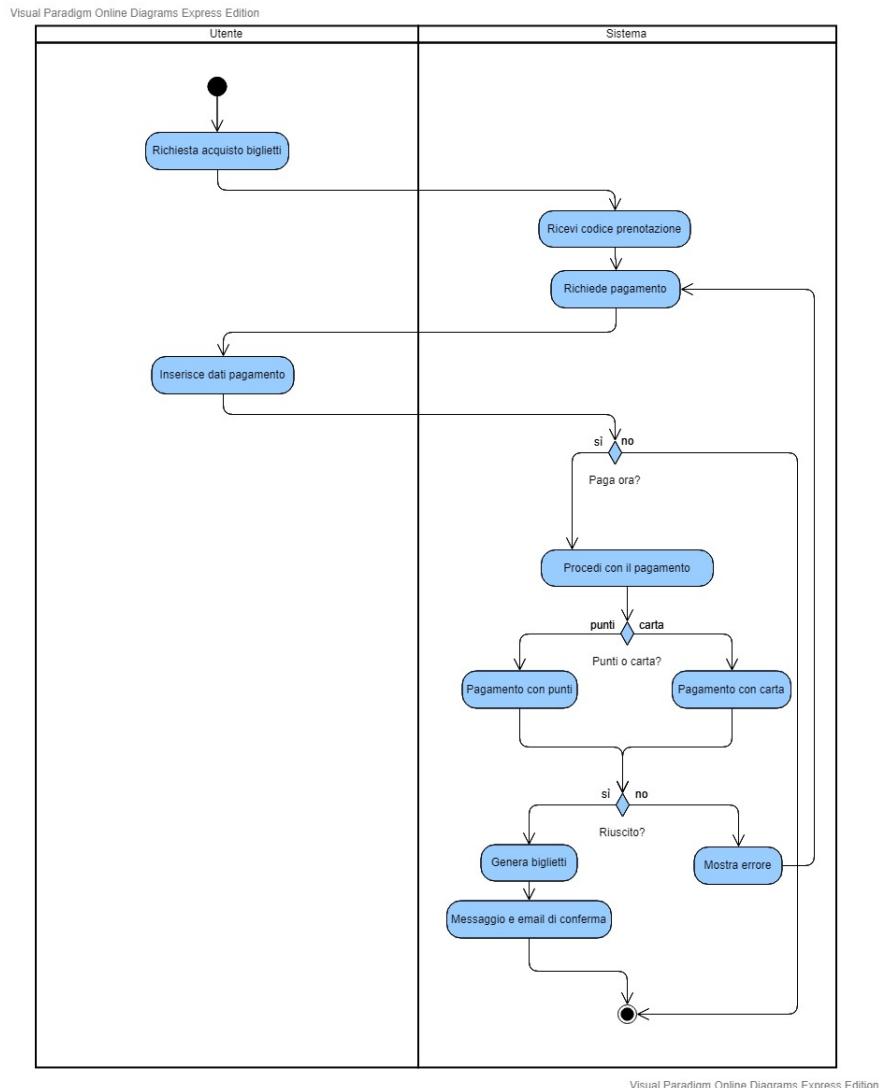
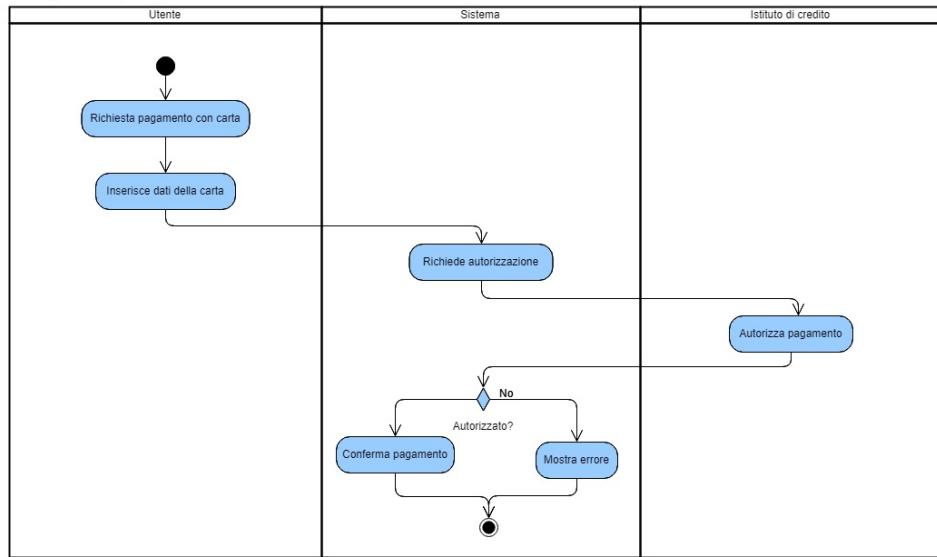
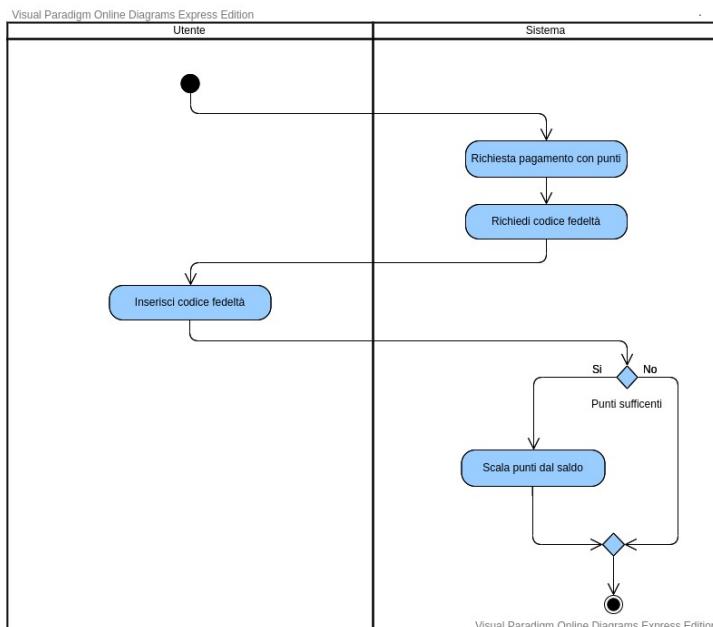


Figure 10: Acquisto dei biglietti precedentemente prenotati



Visual Paradigm Online Diagrams Express Edition

Figure 11: Pagamento con carta



Visual Paradigm Online Diagrams Express Edition

Figure 12: Pagamento con punti

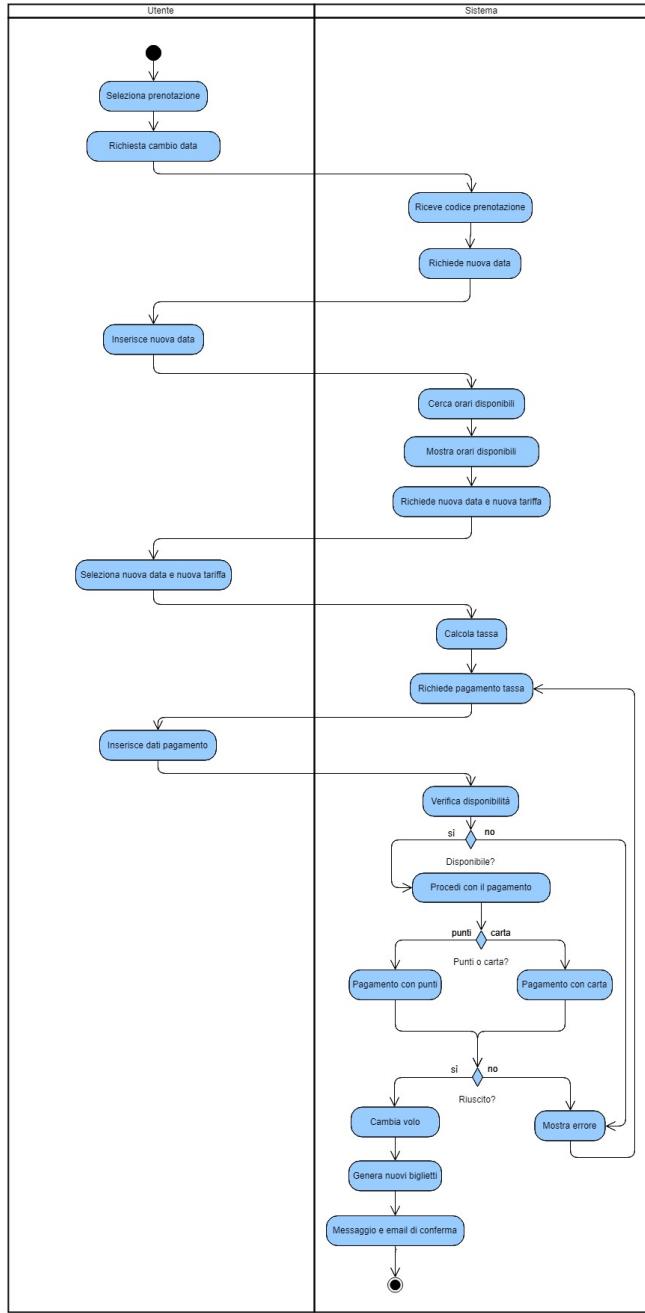
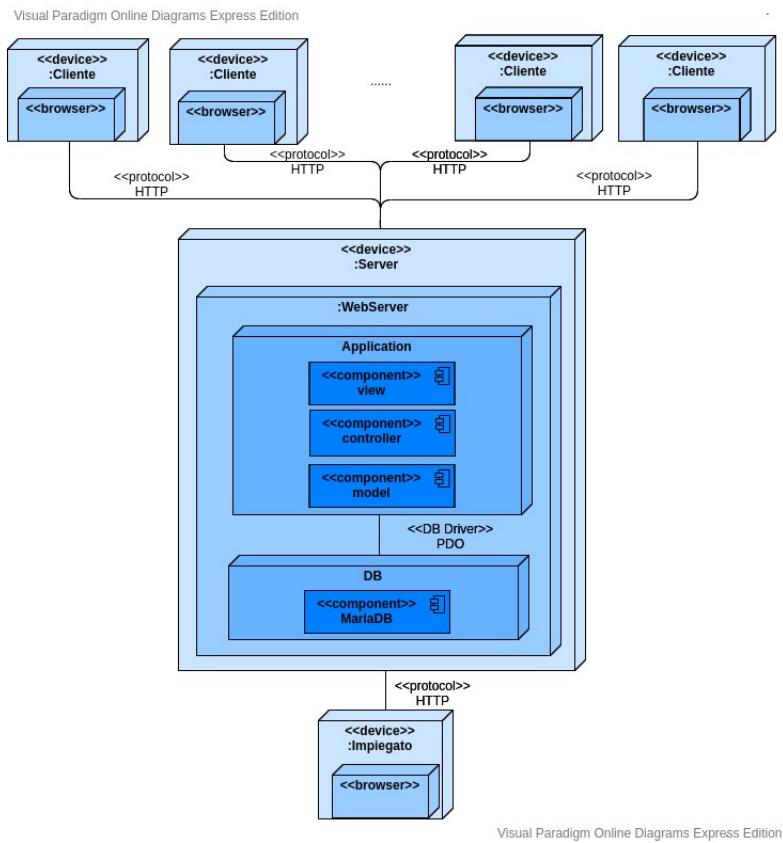
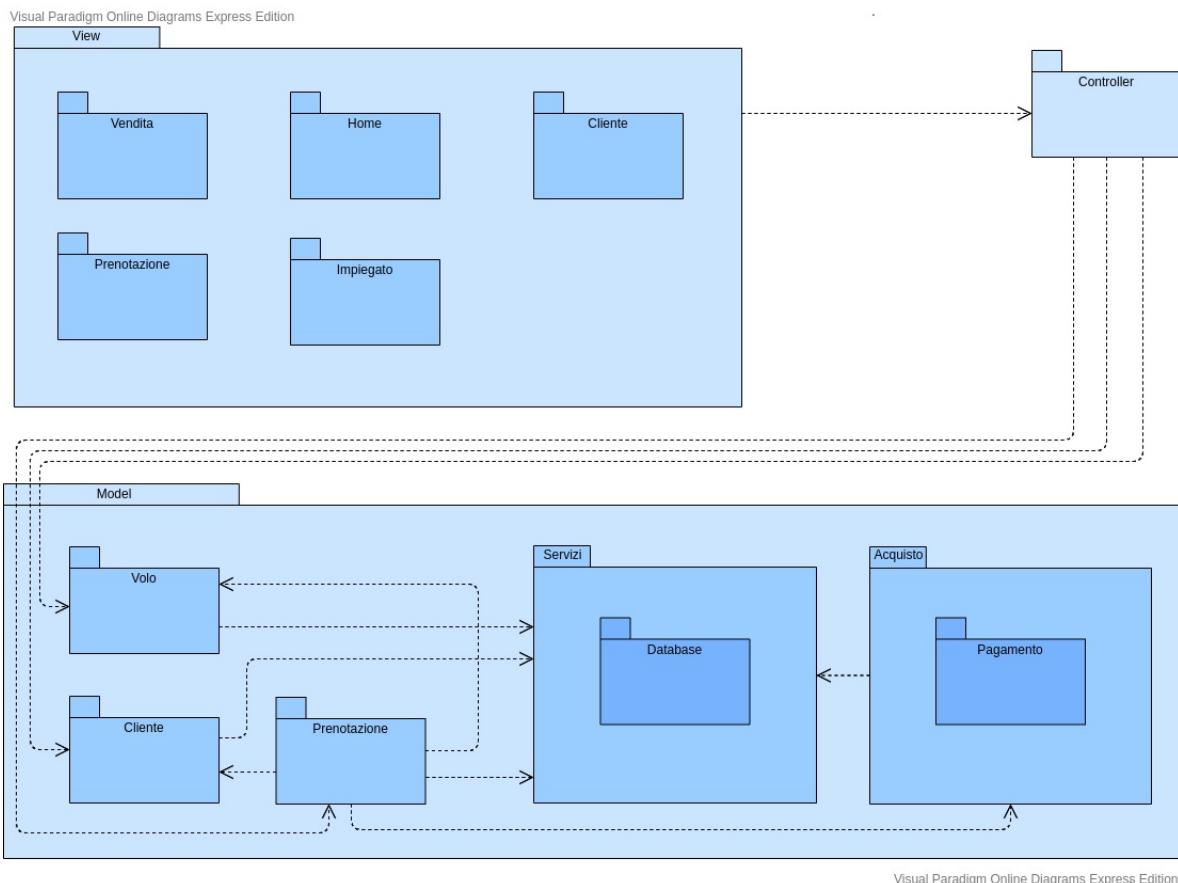


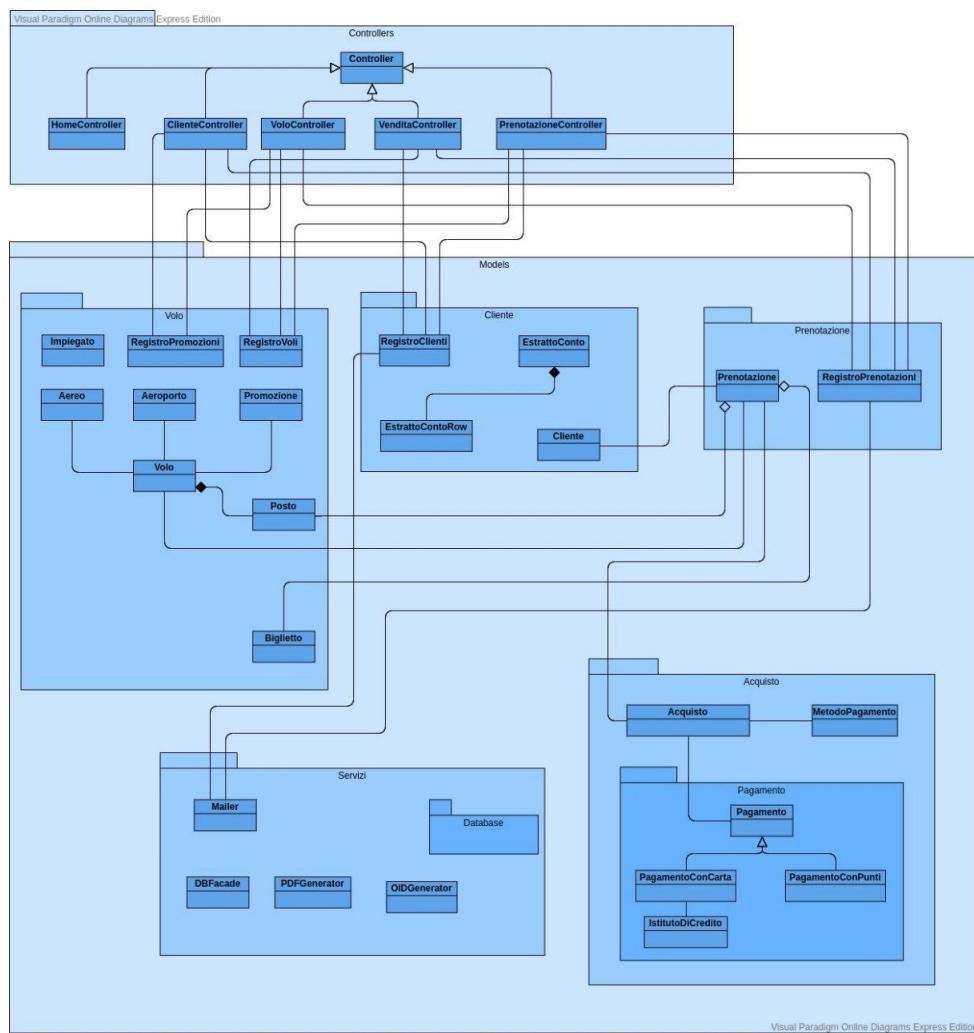
Figure 13: Cambiamento data dei biglietti precedentemente prenotati

2.4 Diagramma di deployment

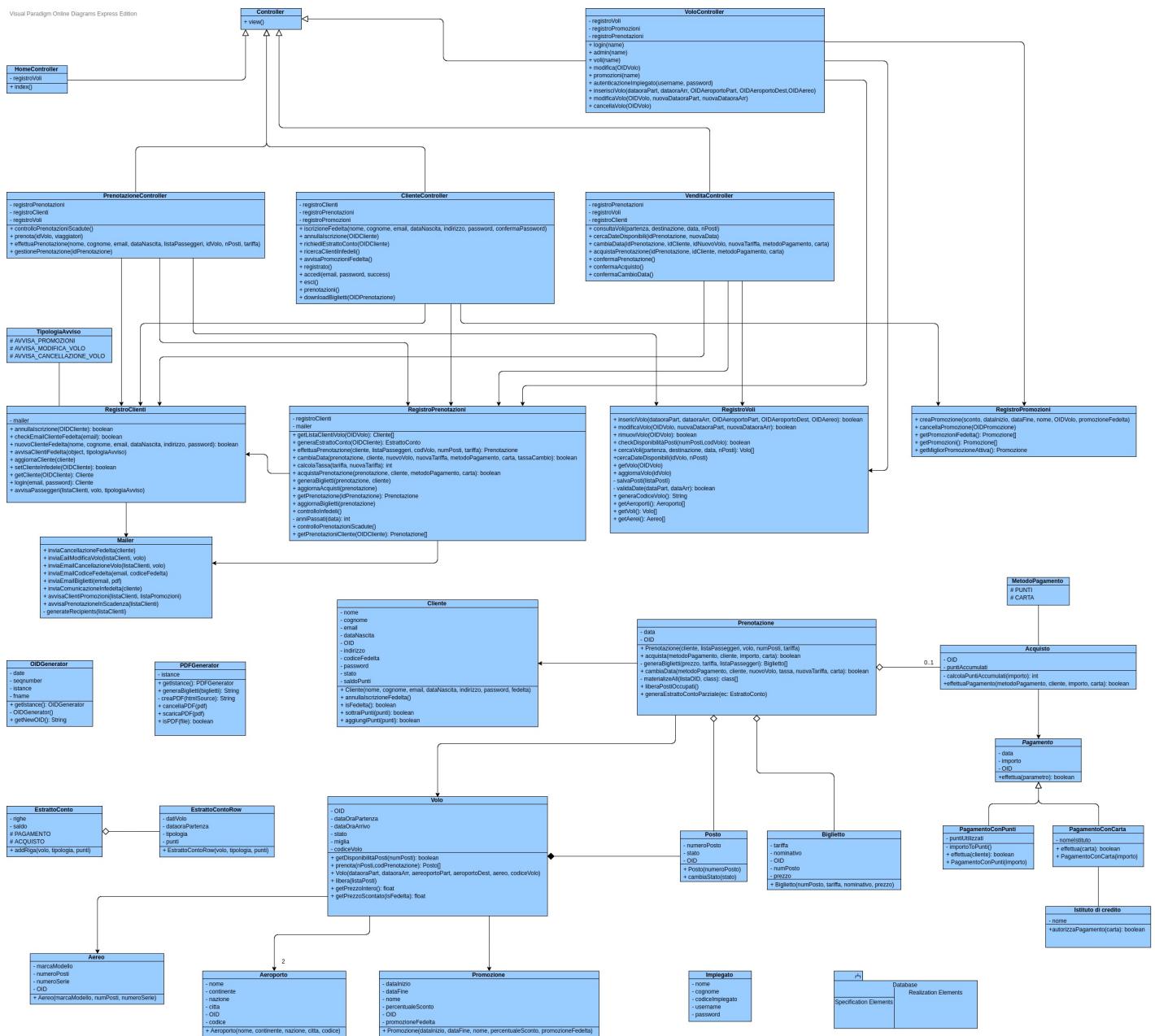


2.5 Architettura Software

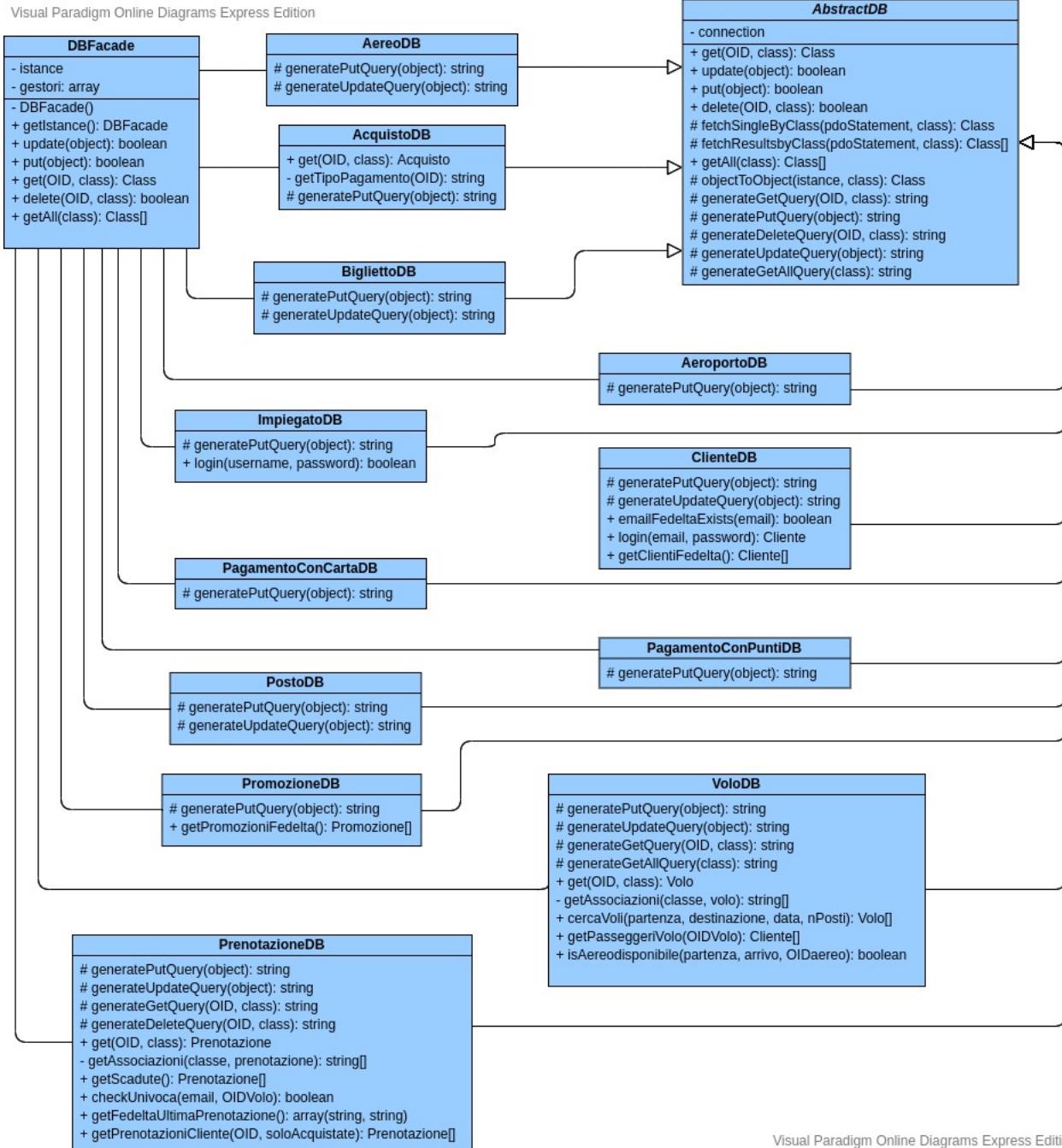




2.6 Classi di progetto



Visual Paradigm Online Diagrams Express Edition



2.6.1 Gestione dei clienti

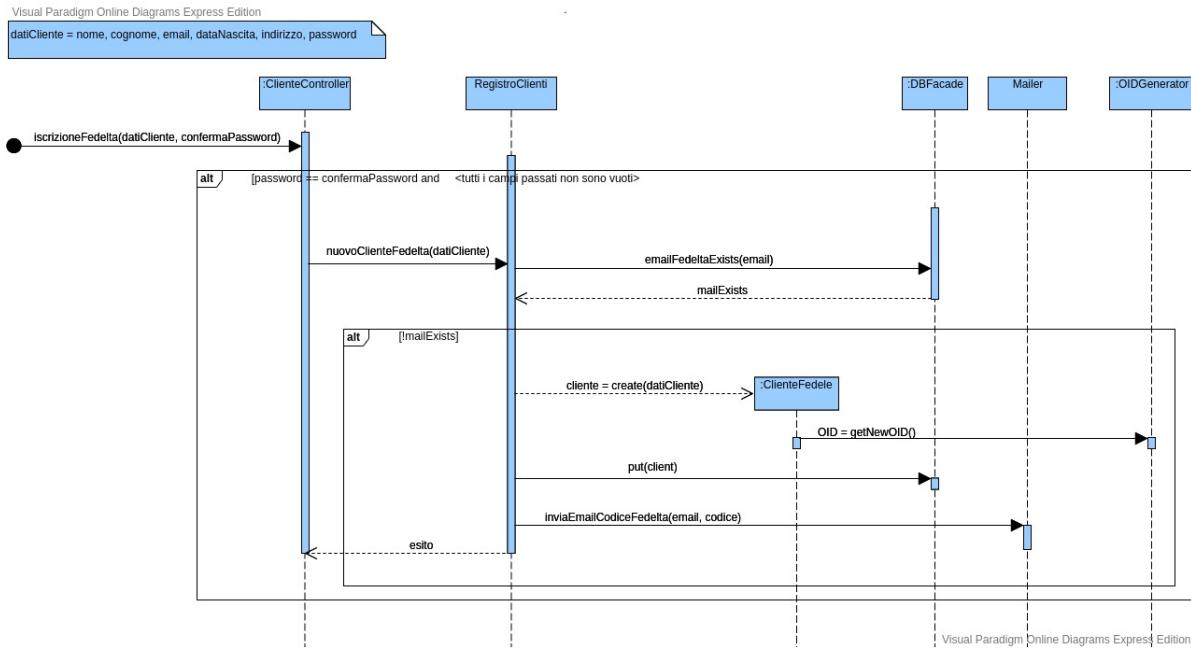


Figure 14: Iscrizione al programma fedelta

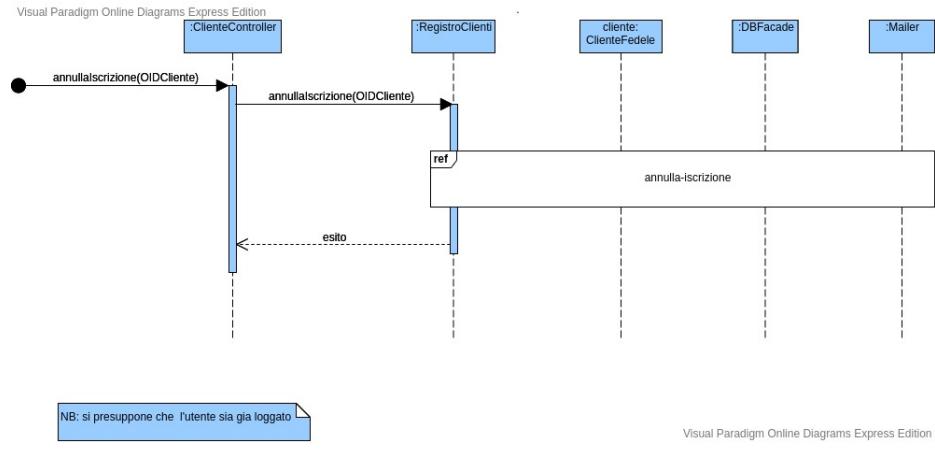
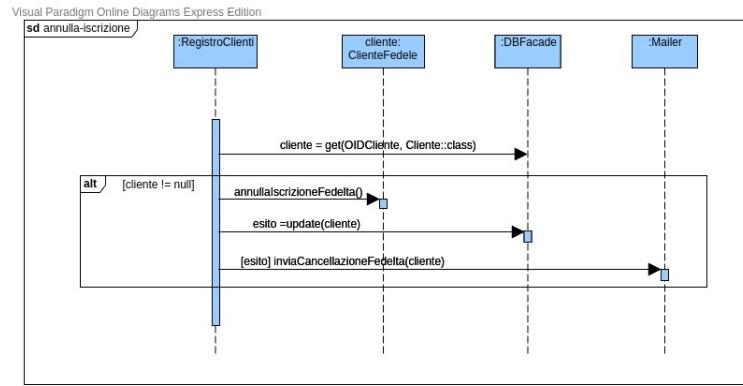


Figure 15: Richiesta annullamento programma fedelta



NB: si presuppone che l'utente sia già loggato

Visual Paradigm Online Diagrams Express Edition

Figure 16: Annullamento iscrizione programma fedelta

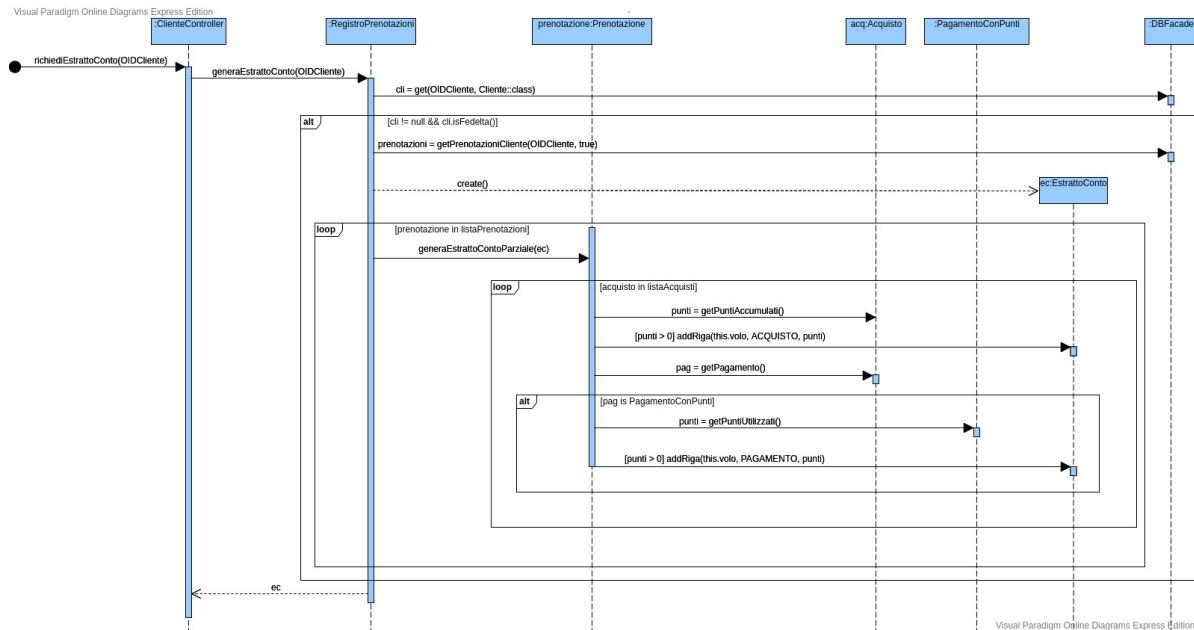


Figure 17: Realizzazione estratto conto del cliente

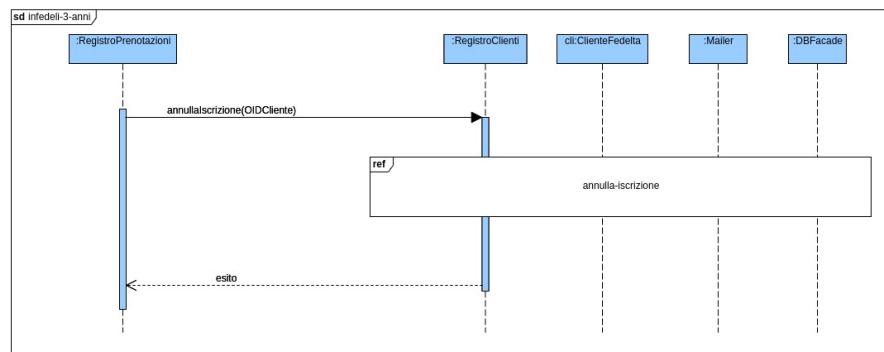
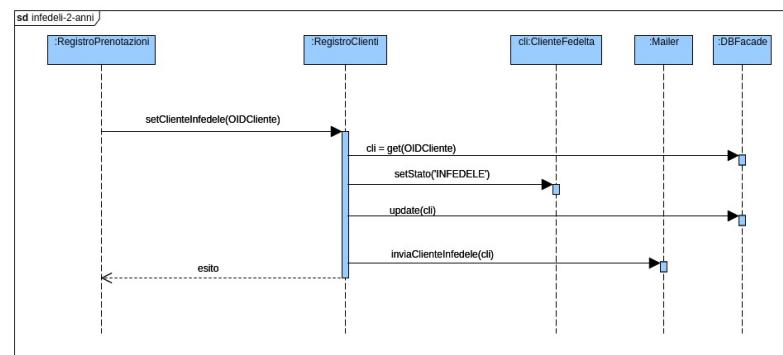
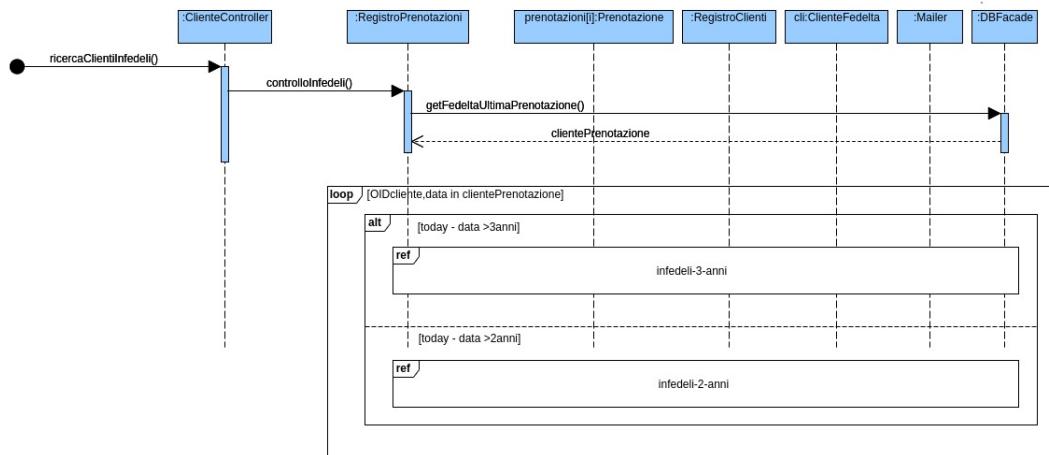


Figure 18: Ricerca clienti infedeli e cancellazione dei clienti infedeli che non effettuano un acquisto da un anno

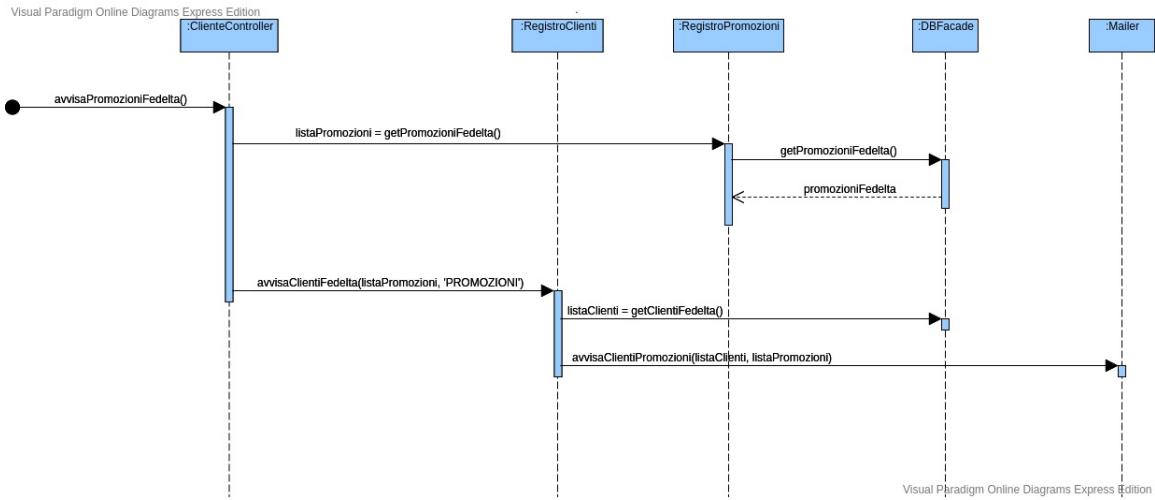


Figure 19: Avviso delle promozioni fedelta per i clienti fedelta

2.6.2 Gestione dei voli

Tutti questi, vengono effettuati dall'impiegato tramite le schermate dedicate.

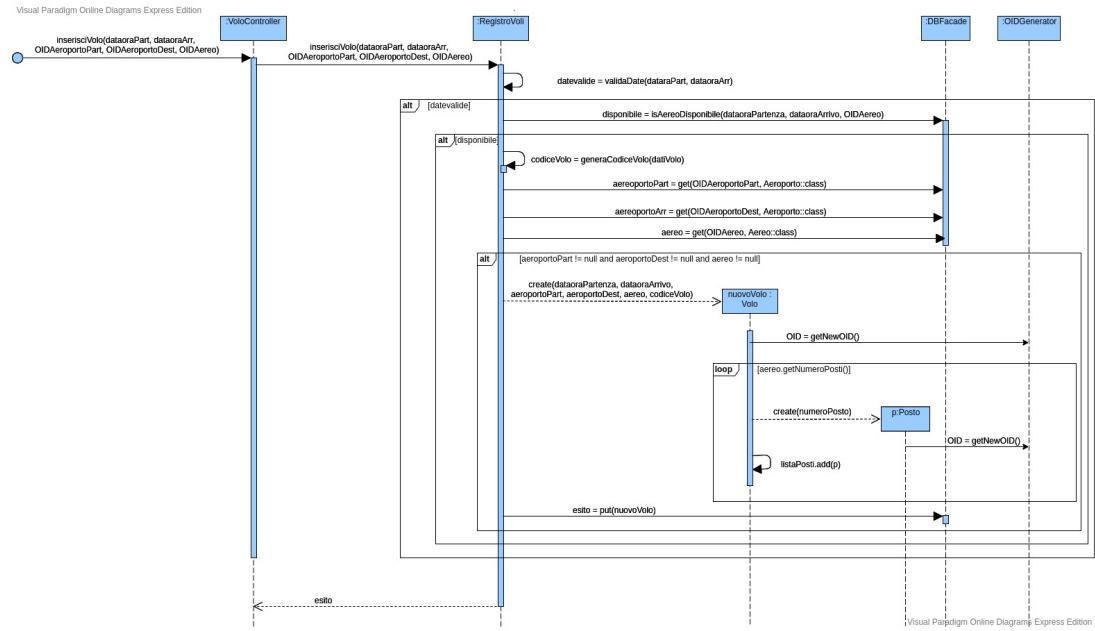


Figure 20: Inserimento nuovo volo

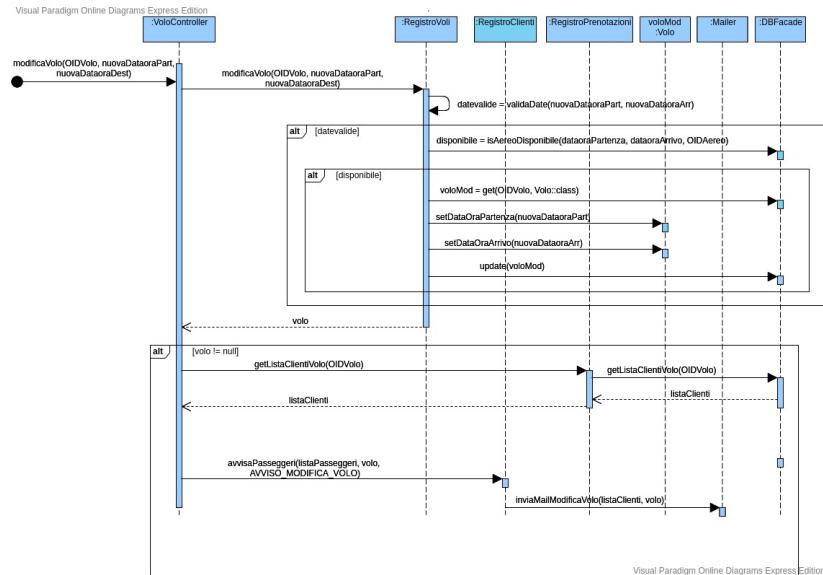


Figure 21: Modifica di un volo

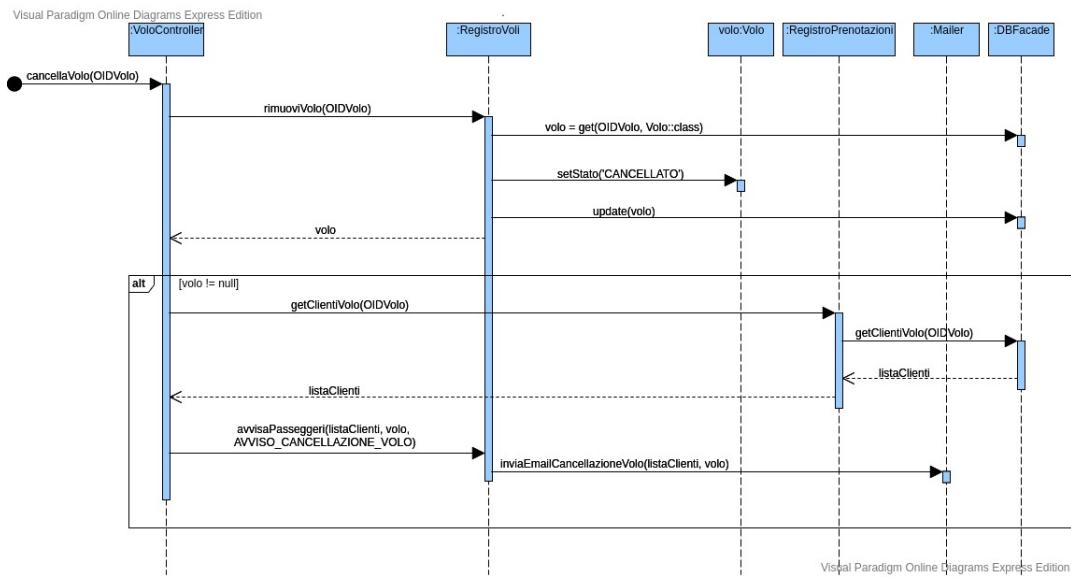


Figure 22: Cancellazione del volo

2.6.3 Prenotazione

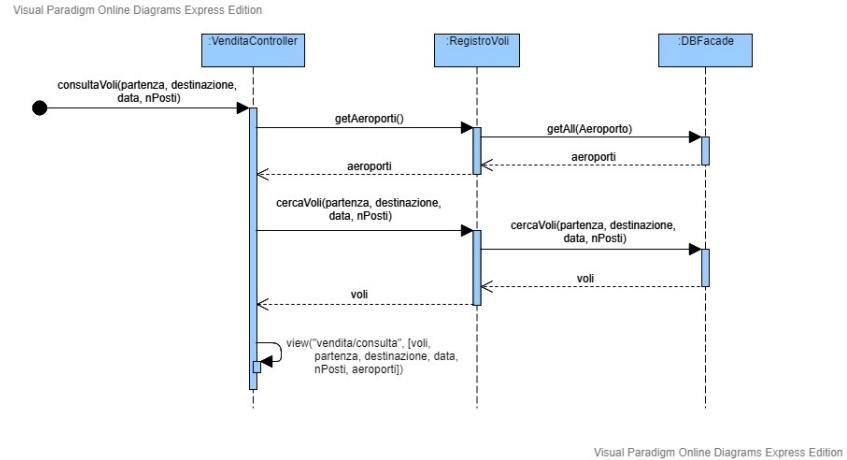


Figure 23: Consultazione dei voli disponibili

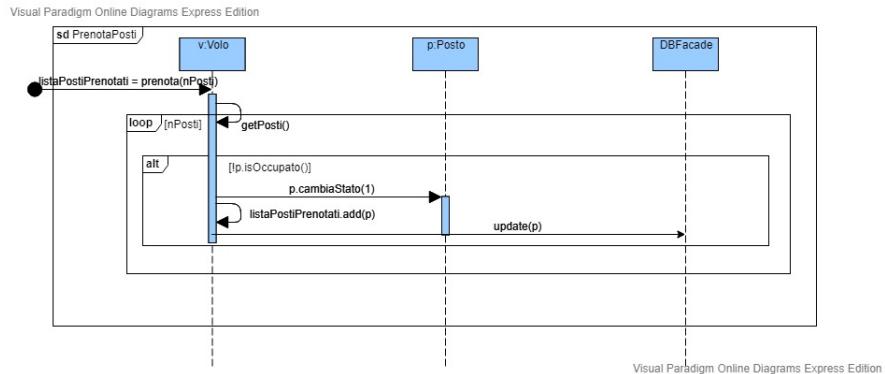
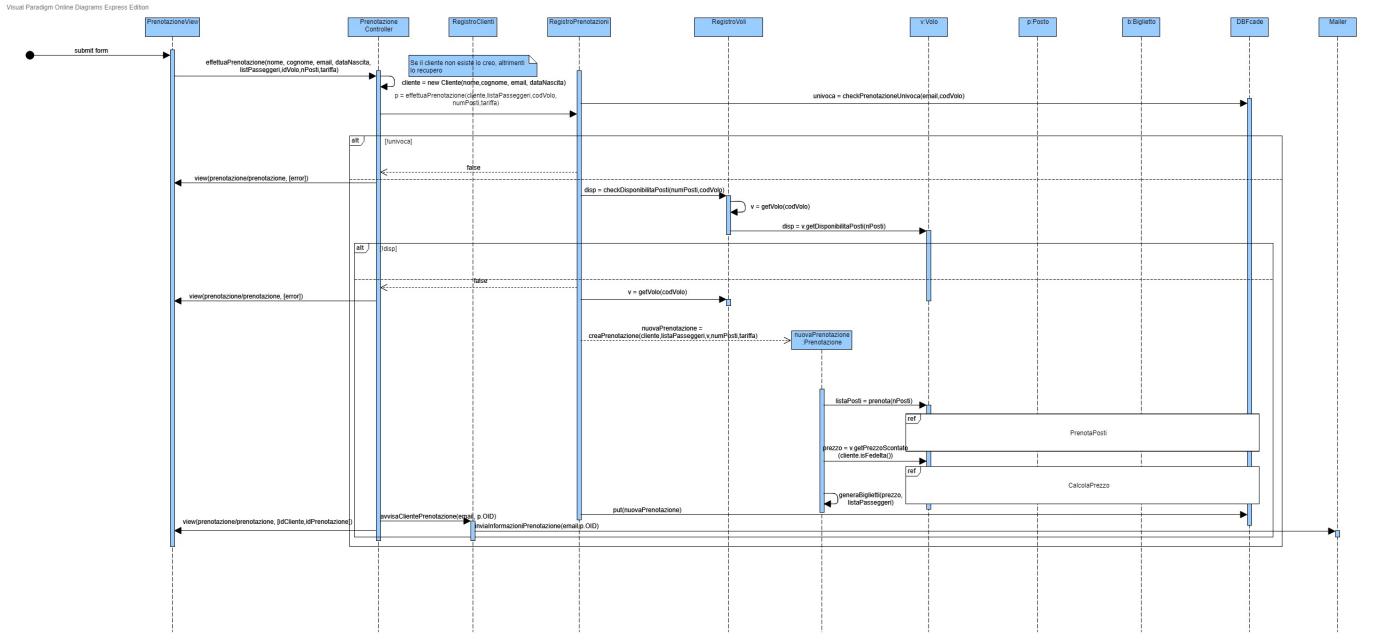


Figure 24: Prenotazione dei posti sul volo



Visual Paradigm Online Diagrams Express Edition

Figure 25: Creazione di una nuova prenotazione

2.6.4 Acquisto di una prenotazione effettuata

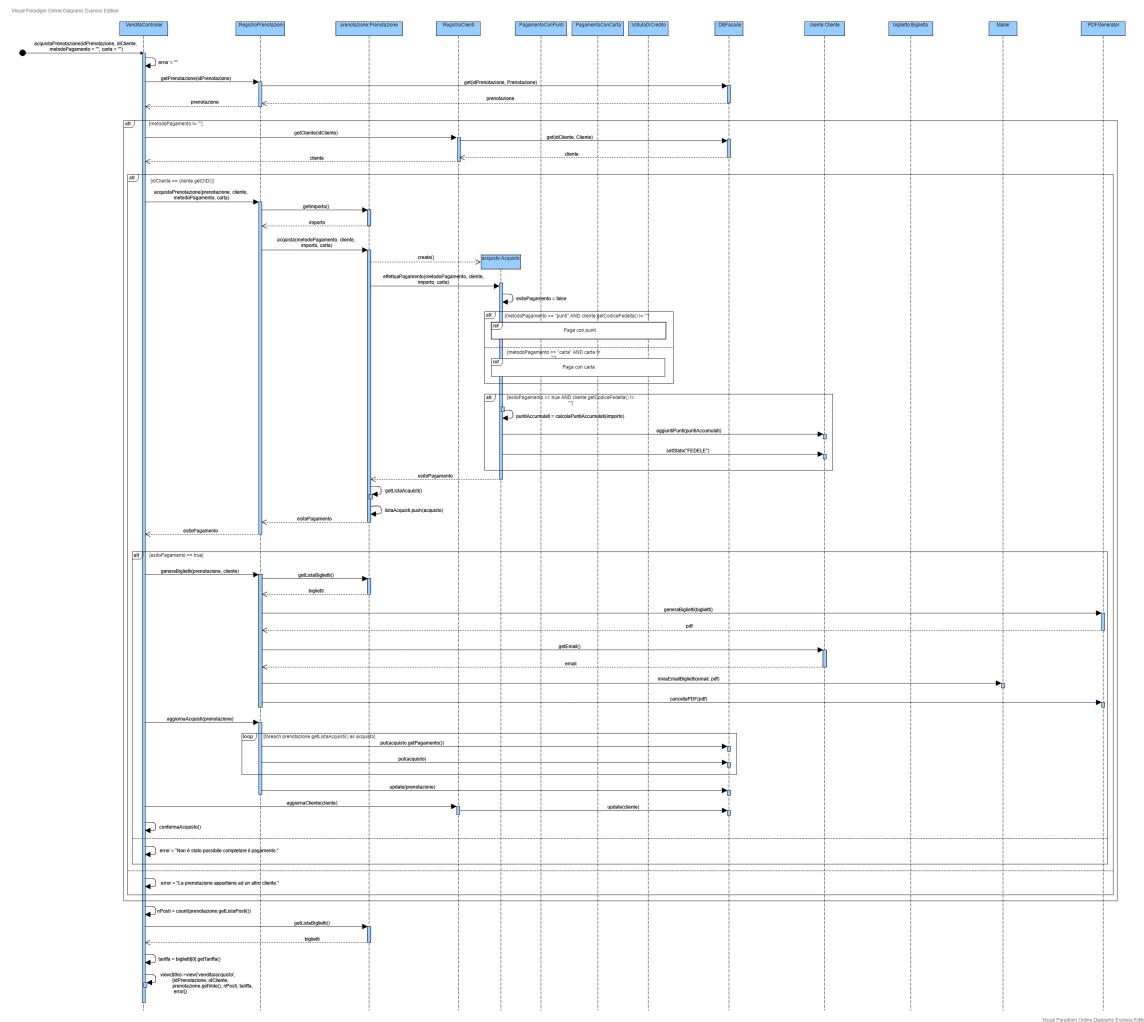


Figure 26: Procedura per l'acquisto di una prenotazione effettuata in precedenza

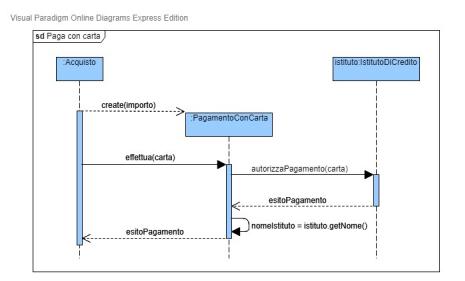


Figure 27: Pagamento usando la carta di credito

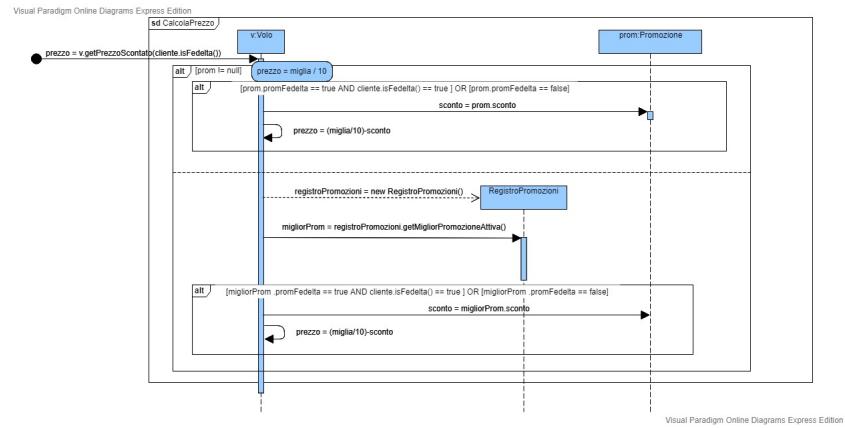


Figure 28: Calcolo del prezzo del volo

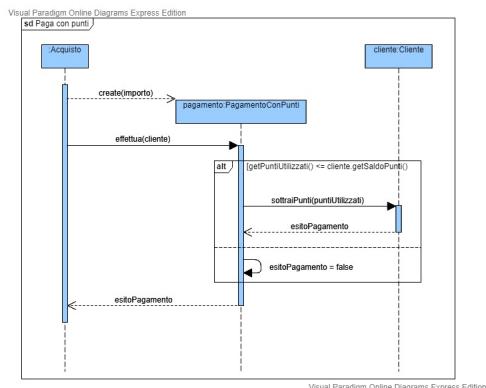


Figure 29: Pagamento usando i punti nel caso di cliente fedelta

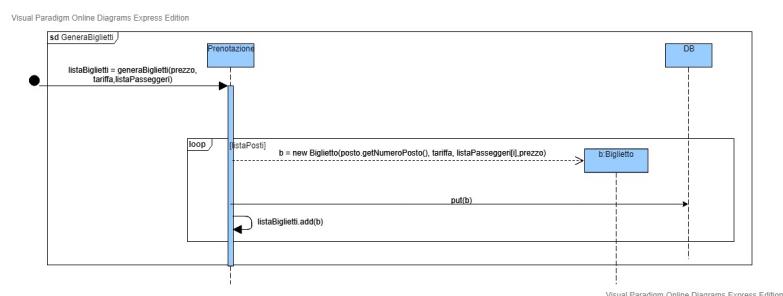


Figure 30: Generazione dei biglietti

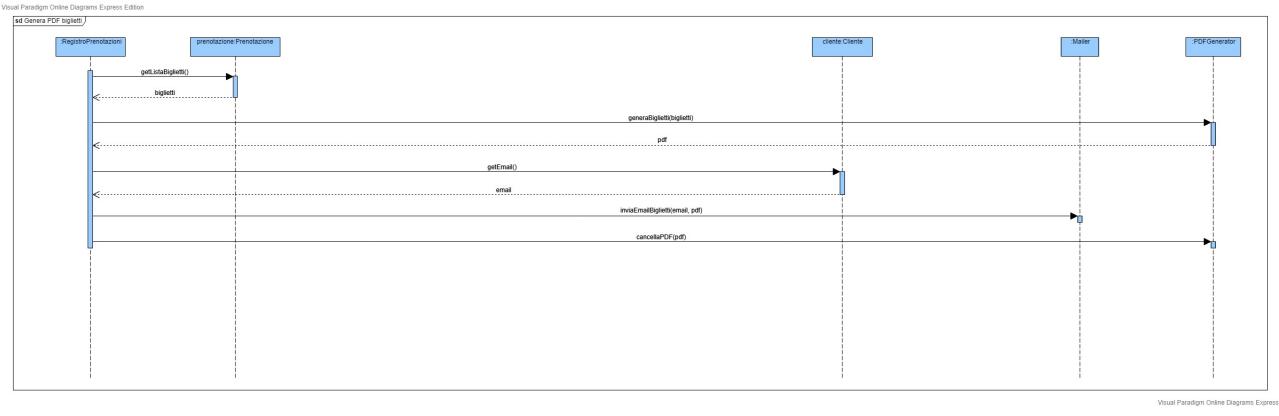


Figure 31: Generazione e invio dei PDF dei biglietti al cliente

2.6.5 Cambio data di una prenotazione

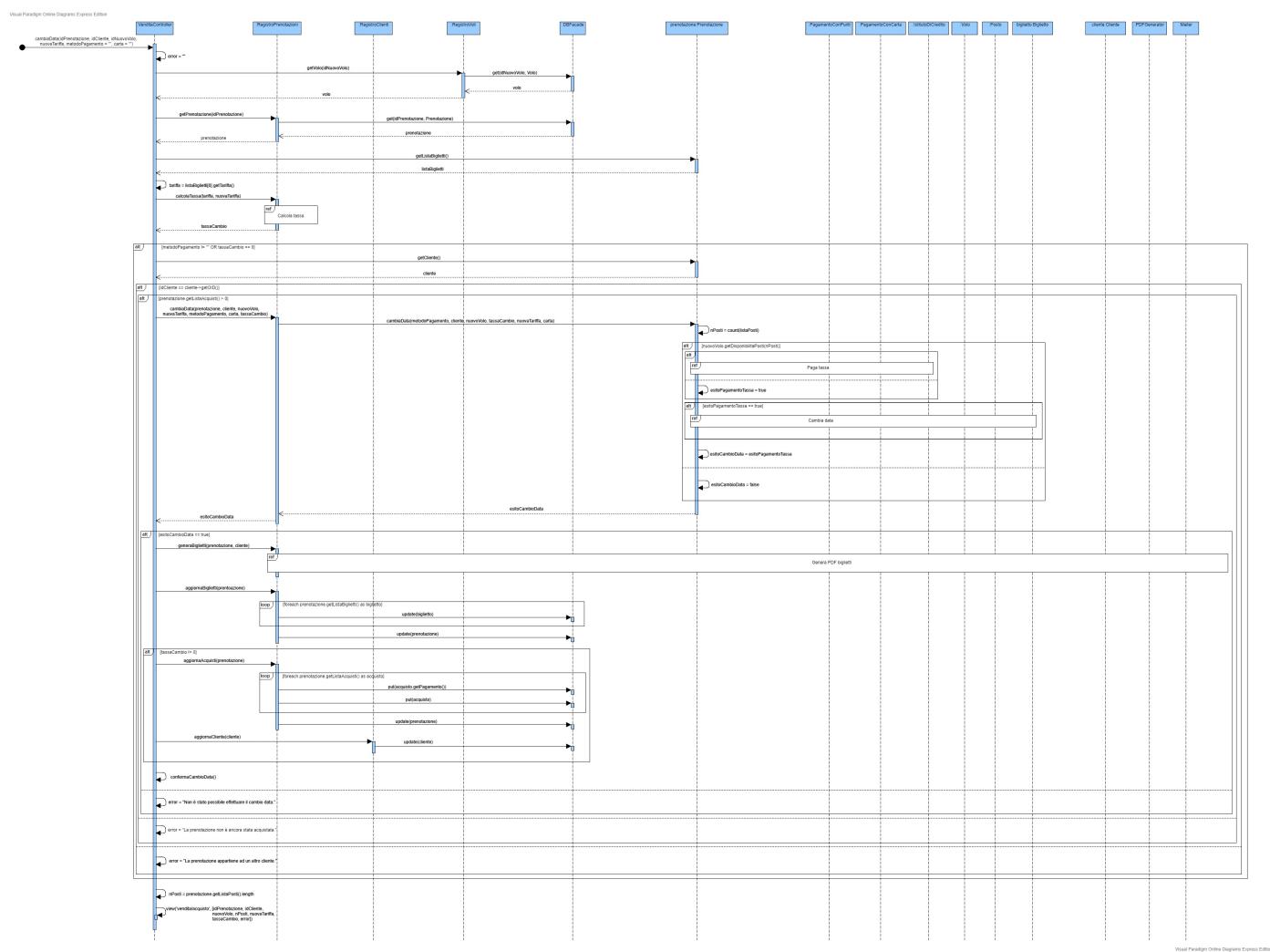


Figure 32: Cambiamento della data di una prenotazione

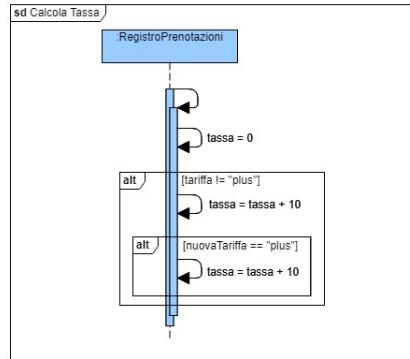


Figure 33: Calcolo della tassa in caso di cambio tariffa

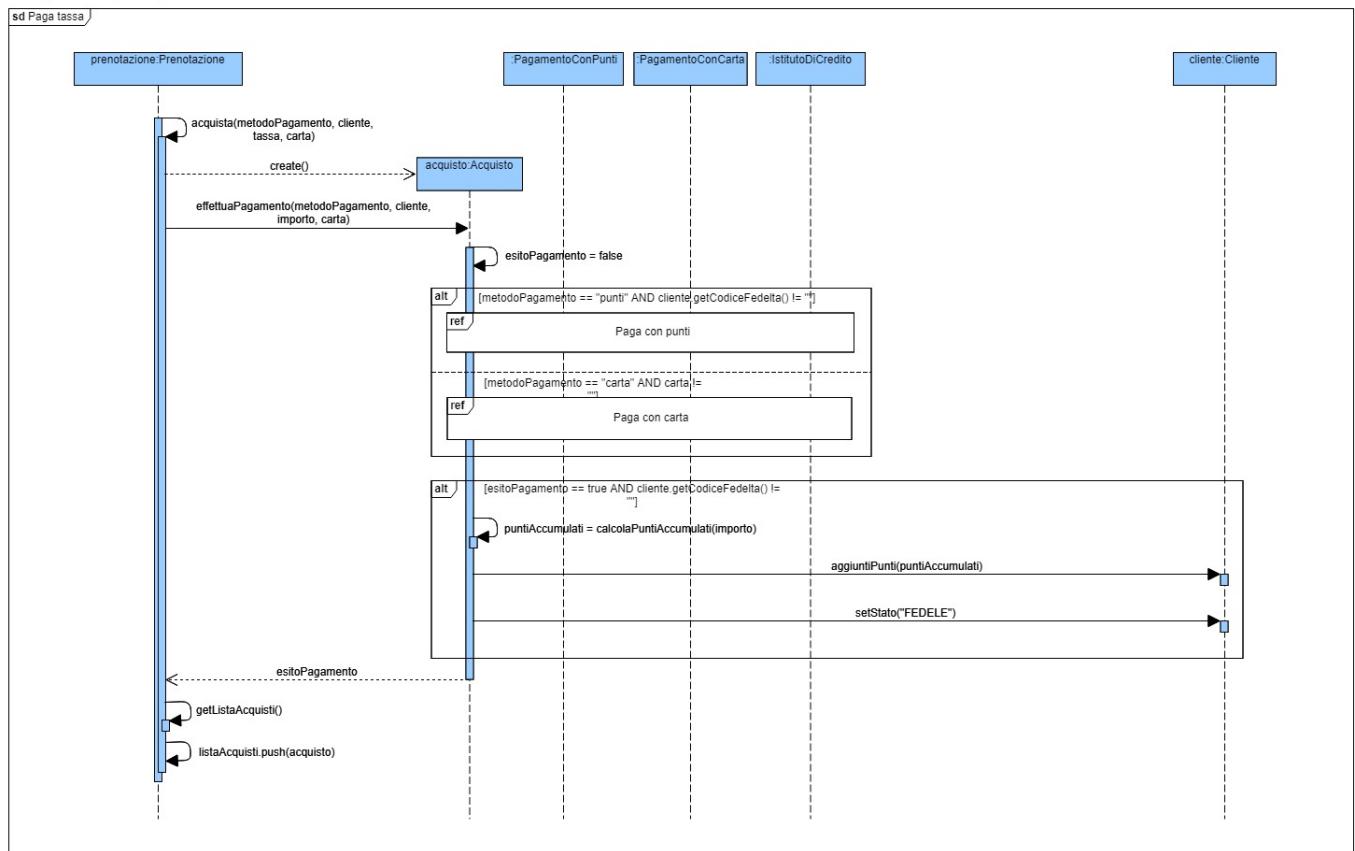


Figure 34: Pagamento della tassa

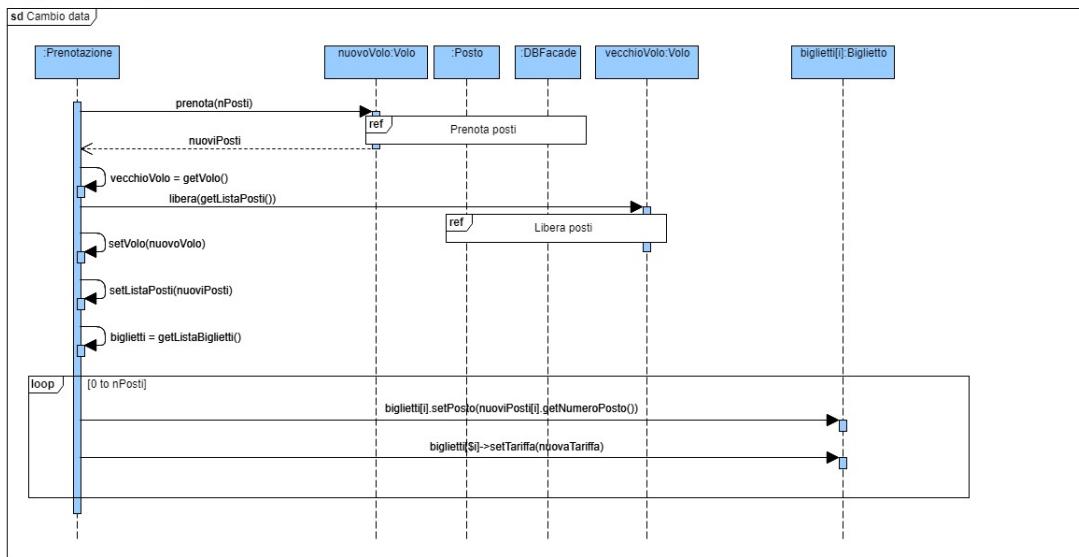


Figure 35:

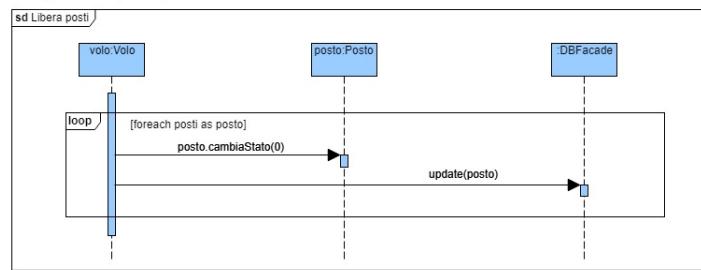


Figure 36: Liberazione dei posti prenotati precedentemente

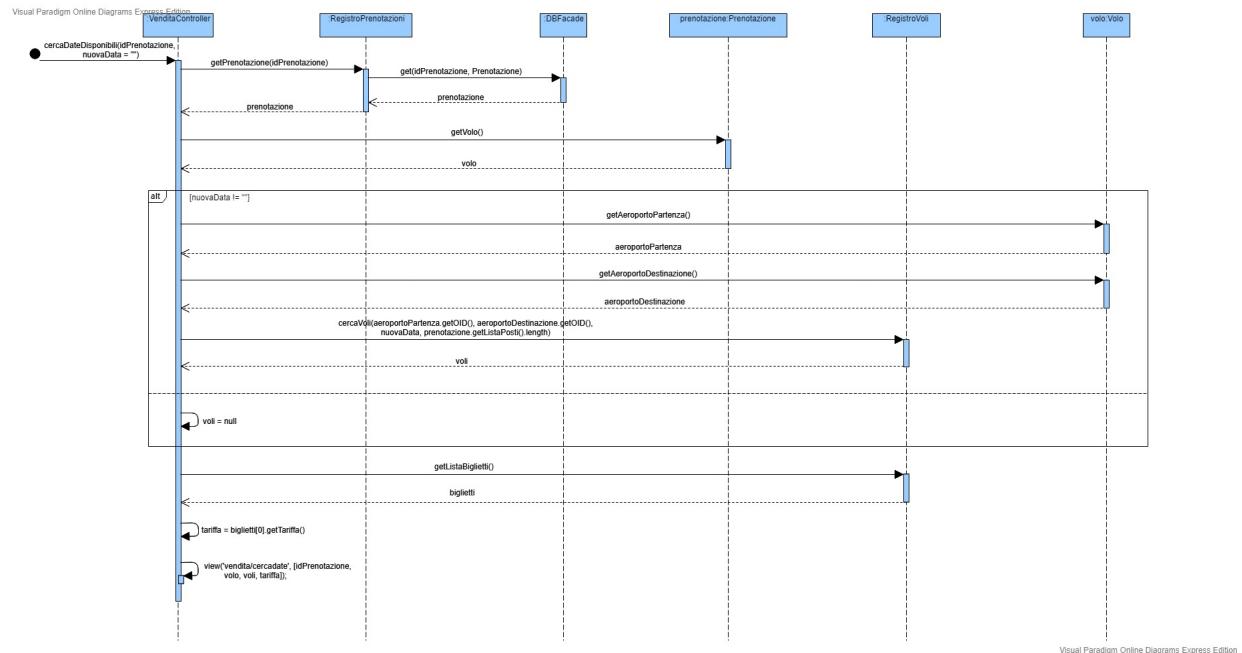


Figure 37: Ricerca delle date disponibili per il cambio data

2.6.6 Funzionamento DBFacade

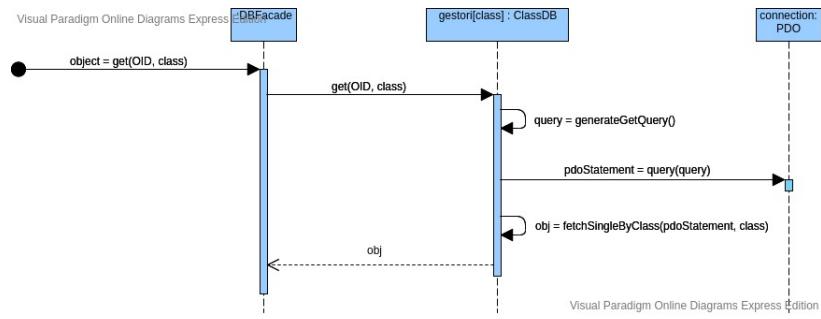


Figure 38: Funzionamento chiamata get

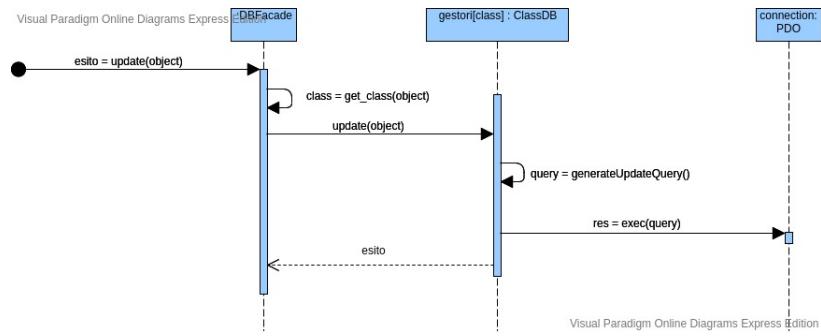


Figure 39: Funzionamento chiamata update

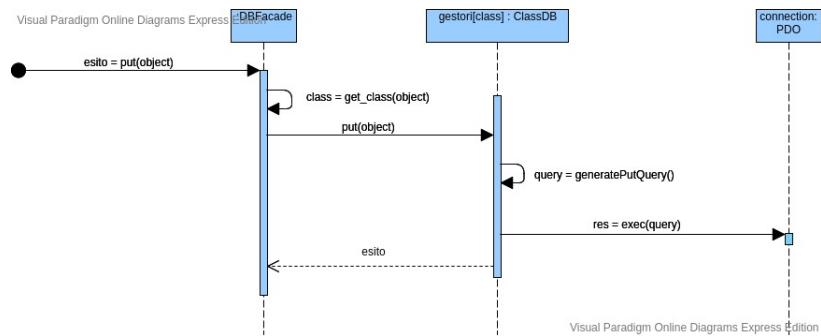


Figure 40: Funzionamento chiamata put

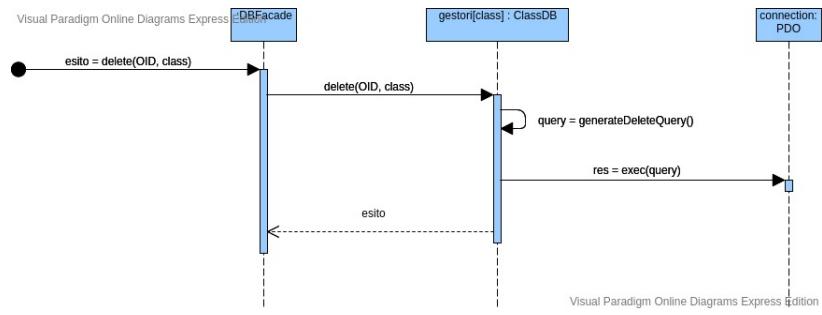


Figure 41: Funzionamento chiamata delete

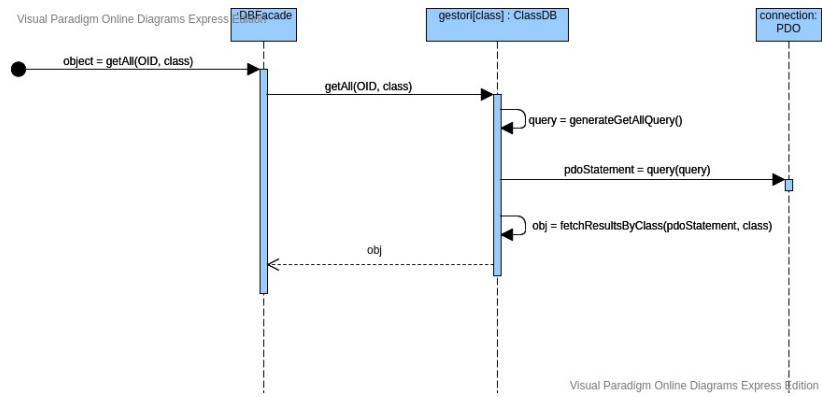


Figure 42: Funzionamento chiamata getAll

2.6.7 Gestione Promozioni

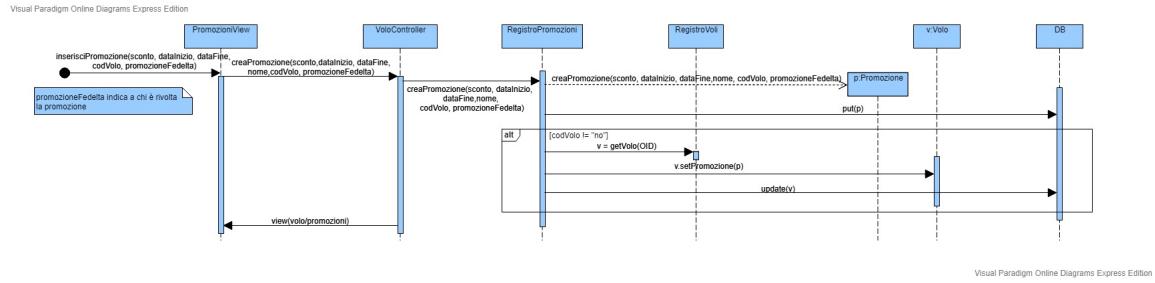


Figure 43: Inserimento nuova promozione

2.6.8 Generazione codici OID

Ogni oggetto ha assegnato un codice univoco, chiamato OID in tutte le classi. La lunghezza di un codice OID è 15 caratteri, divisi in:

- I primi 8 caratteri sono la data, in formato yyyyMMdd (es: 20200101)
- Gli altri 7 caratteri, rappresentano un numero di sequenza giornaliero

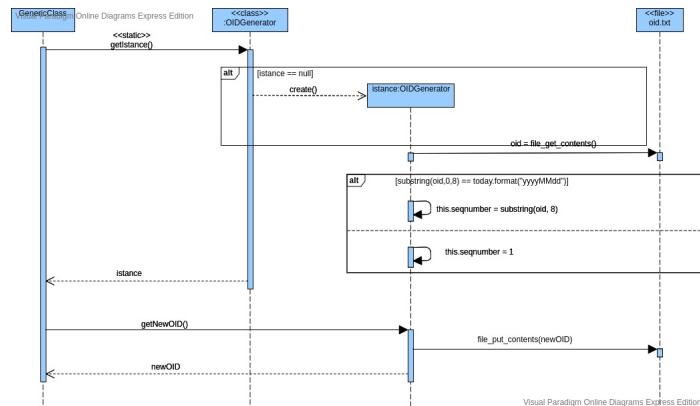


Figure 44: Generazione di un nuovo codice univoco per l'oggetto

2.7 Stati degli oggetti

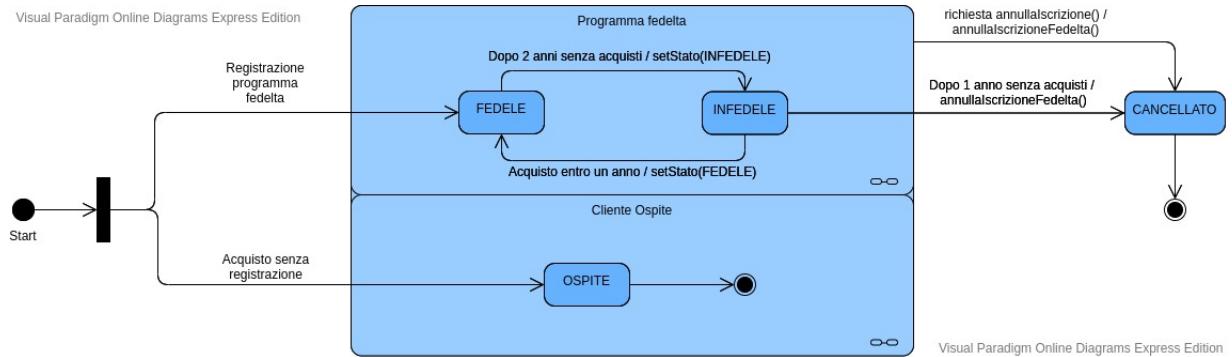


Figure 45: Rappresentazione degli stati di un cliente

2.8 Design Pattern considerati nello sviluppo

Durante la progettazione del sistema, abbiamo considerato i seguenti pattern GRASP:

- **Creator:** per decidere chi deve essere responsabile di creare cosa.
- **Information Expert:** per l'assegnazione delle responsabilità alle varie classi (Anche se ogni tanto è stato violato)
- **Low coupling e High cohesion:** anche questi due pattern sono stati considerati nelle decisioni delle assegnazioni delle responsabilità. Abbiamo cercato di sviluppare il codice pensando sempre alla coesione/accoppiamento delle classi, per avere classi focalizzate e debolmente accoppiate.

Inoltre, altri patterns che sono stati considerati nello sviluppo del sistema sono:

- **Strategy:** abbiamo applicato strategy nelle classi Pagamento, per permettere, con una sola chiamata effettua(..), di utilizzare due differenti algoritmi.
- **Singleton:** nel nostro progetto sono presenti due classi Singleton: OIDGenerator e DBFacade. Abbiamo optato per questa soluzione per garantire che ci sia un unico punto di accesso ai servizi forniti dal DB (per evitare di causare problemi con le letture/scritture..) e alla generazione degli OID, per garantire l'univocità dello stesso.
- **Facade:** siccome serviva un interfaccia comune per tutti i servizi servizi dal database, abbiamo deciso di sviluppare una Facade per fornire l'interfaccia di accesso allo stesso. Questa Facade poi comunica con diverse classi (quelle presenti nel package database) per fornire l'accesso ai vari metodi del database.

Per lo sviluppo del database invece, abbiamo seguito i seguenti design pattern:

- **Representing objects as tables:** per permettere di mappare direttamente le classi con le tabelle del database. Purtroppo per la classe Pagamento non è stato possibile applicare questo pattern a causa della difficoltà nella gestione dell'associazione con la classe Acquisto.
- **Object Identifier:** abbiamo utilizzato questo approccio per la generazione degli ID univoci agli oggetti.
- **Database mapper:** la DB facade collabora, per fornire i servizi di persistenza agli oggetti, con diversi Database Mapper. Non abbiamo però direttamente applicato questo pattern ma stato più una 'fonte di ispirazione'. La differenza sostanziale tra il pattern e il nostro codice è che la nostra soluzione prevede l'utilizzo di una classe astratta invece che di un'interfaccia, per favorire il riutilizzo del codice (esempio: la funzione fetchResultsByClass).
- **Template method:** i metodi get, put, update, delete e getAll sono tutti metodi template. Questi metodi contengono la logica di interrogazione del DB e fetch dei risultati nelle classi, utilizzando metodi hook per la generazione delle query (la parte variabile per ogni classe). Un esempio di metodo hook è generateGetQuery(), che restituisce la query per recuperare un'oggetto di quella determinata classe.

L'architettura del software invece segue il pattern MVC, scelto perchè si adatta molto bene alle esigenze delle applicazioni web.

Abbiamo inoltre considerato il pattern Strategy su IstitutoDiCredito per la gestione dei pagamenti con carta, in modo che ogni diversa sottoclasse (che rappresenta un istituto di credito) implementi i metodi per la comunicazione con un determinato sottosistema bancario.

Questa ipotesi è stata scartata poichè i pagamenti sono stati realizzati (ovviamente) in modo fittizio, cioè decidendo l'esito in base a un valore randomico.

3 Note tecniche

3.1 Database

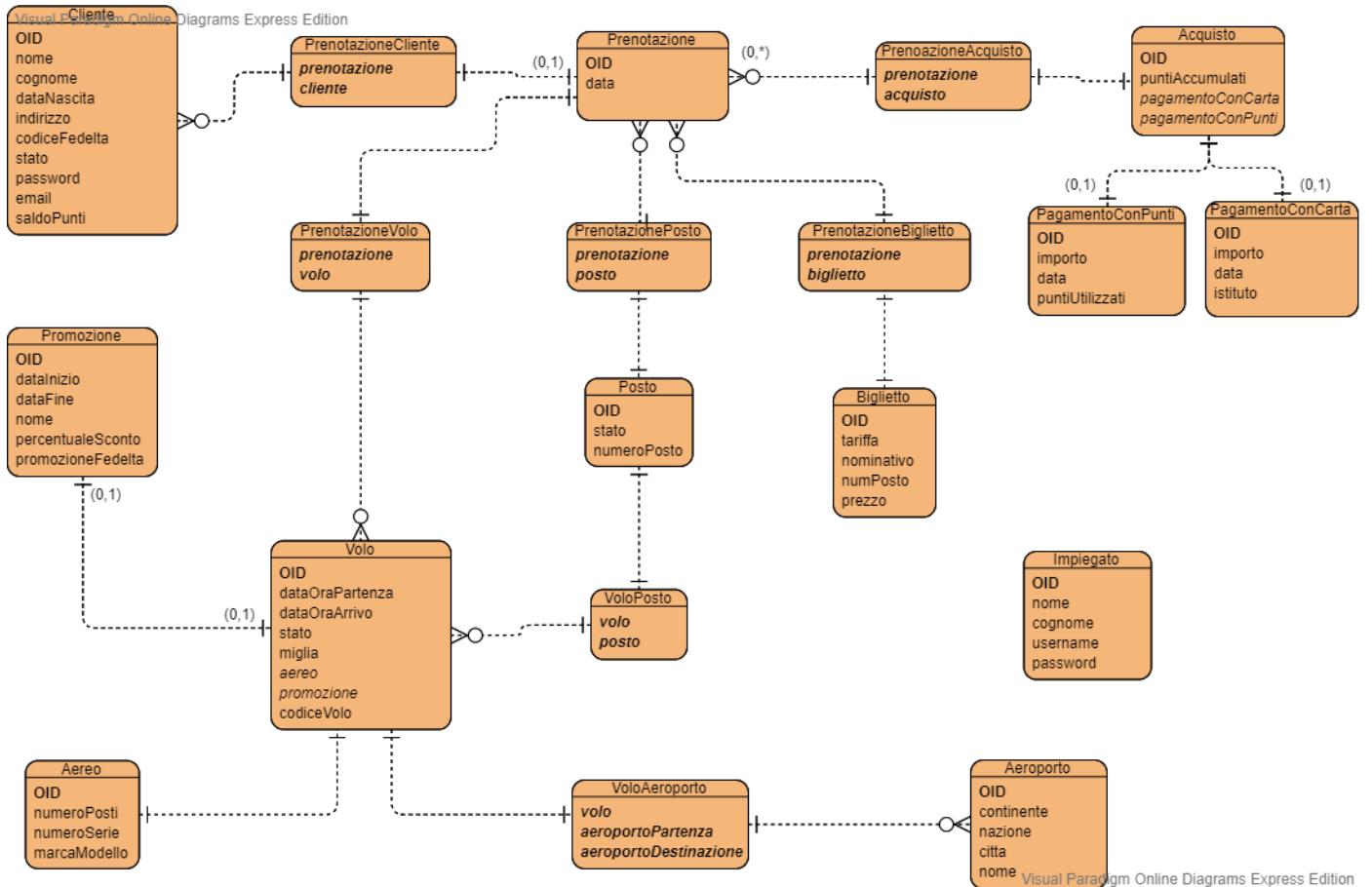


Figure 46: Schema Database

Per il salvataggio delle informazioni riguardanti voli, prenotazioni e clienti ci siamo appoggiati a un Database MySQL.

Per recuperare i dati da questo DB abbiamo utilizzato la materializzazione pigra. Questo tipo di gestione ci ha permesso di avere oggetti molto più snelli e un sistema molto più performante dato che la materializzazione dal db viene effettuata solo quando strettamente necessario.

3.2 Cronjobs

Abbiamo utilizzato dei cronjob per:

- Cercare clienti infedeli
- Controllare prenotazioni scadute
- Avvisare utenti fedeli delle promozioni

	Title, Address	Last execution	Next execution	
	Ricerca Clienti Infedeli https://gruppoaereo4.000webhostapp.com/...	-	-	History Edit
	Controlla Prenotazioni Scadute https://gruppoaereo4.000webhostapp.com/...	-	-	History Edit
	Avvisa Promozioni Fedeltà https://gruppoaereo4.000webhostapp.com/...	Today, 14:00:53 Successful (3,3 s)	16.03., 14:00	History Edit

Figure 47: Cronjobs

Siccome 000webhost mette a disposizione solo un cronjob e non volevamo scrivere uno script ad hoc contenente tutte le funzioni da eseguire periodicamente (al fine di mantenere tutto ben separato) abbiamo utilizzato le funzionalità messe a disposizione da cron-job.org. Qui sono stati creati tre cronjob che una volta al giorno o ogni ora eseguono, grazie ai metodi forniti dai controller, le 3 operazioni citate sopra.

4 Funzionalità

4.1 Per cominciare

Per poter usufruire dei servizi del progetto non servirà installare nulla; basterà visitare il sito:

<https://gruppoaereo4.000webhostapp.com/public/>

O in alternativa, se non dovesse funzionare, utilizzare il seguente:

<http://gruppoaereo4.altervista.org/public/>

4.2 Ricerca voli

Una volta sul sito, per poter cominciare a consultare voli si dovrà effettuare una ricerca tramite l'apposito form.

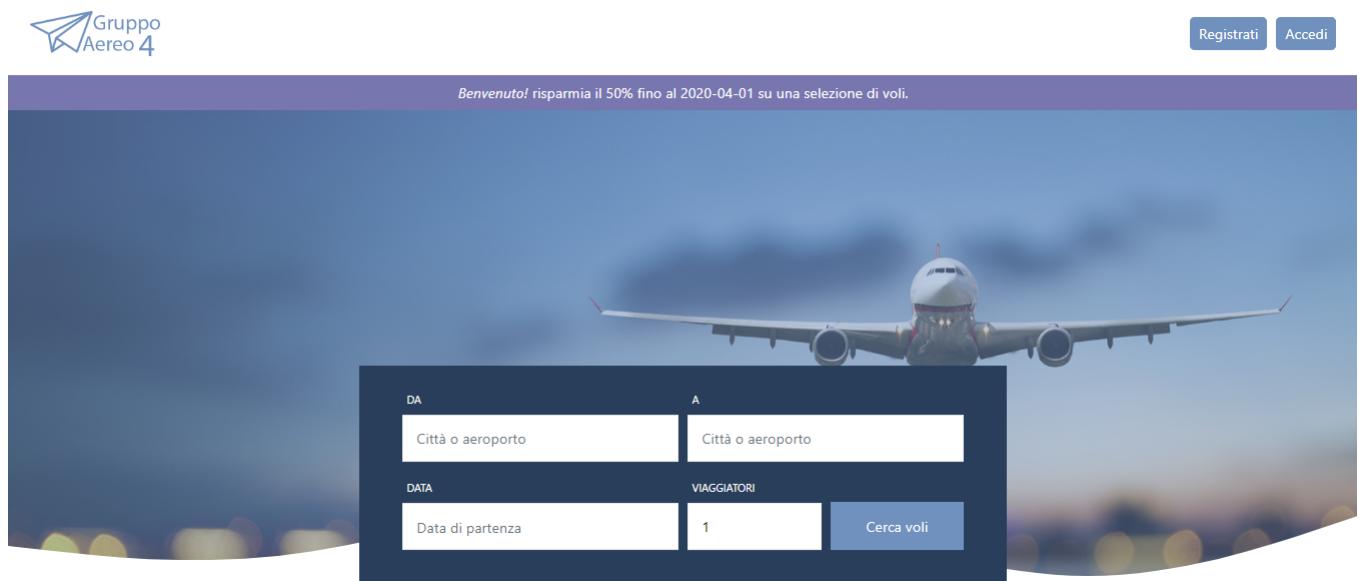


Figure 48: Homepage del sito

4.3 Consultazione voli

Una volta inseriti i dati del volo e aver premuto su "Cerca voli" se esistono voli corrispondenti alla nostra ricerca ci troveremo davanti a una schermata simile a questa: O questa se al volo è stata applicata una promozione:

4.4 Prenotazione volo

Cliccando su "Prenota questo volo" ci troveremo davanti alla schermata della prenotazione. In questa schermata (se si è loggati) è possibile non inserire i primi dati richiesti, ma solo i dati dei passeggeri. In caso contrario si dovranno inserire le informazioni di recapito dell'intestatario della prenotazione e i dati dei passeggeri (che serviranno per la generazione dei biglietti). Inoltre in fase di prenotazione è possibile scegliere il tipo di tariffa:

- Standard
- VoloPlus

La tariffa VoloPlus aggiunge la possibilità di effettuare il cambio data della prenotazione senza sovrapprezzo.



[Registrati](#) [Accedi](#)

Sconto del 20% fino al 20/03/2020 su una selezione di voli.

DA	A	DATA	VIAGGIATORI	
Milano Malpensa (MXP)	Roma Fiumicino (FCO)	2020-10-10	1	Cerca voli

10:30 MXP
2020-10-10



11:30 FCO
2020-10-10

13.48€ 24.50€
totale

[Prenota questo volo](#)

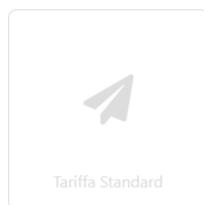
© Copyright 2020, Gruppo Aereo 4

Figure 49: Consultazione dei voli con promozioni

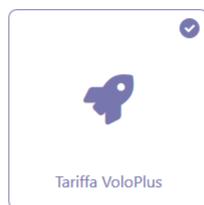


[Registrati](#) [Accedi](#)

Conferma la tua prenotazione



Tariffa Standard



Tariffa VoloPlus

Biglietti 149.00€

Milano Malpensa (MXP)
2020-04-10 09:30:00

Londra Stansted (STN)
2020-04-10 12:30:00

Viaggiatori
1

Supplementi

Powered by 000webhost 20€

Tariffa
VoloPlus

Totali 169.00€

NOME

COGNOME

E-MAIL

DATA DI NASCITA

Passeggero 1

NOME

COGNOME

[Prenota](#)

© Copyright 2020, Gruppo Aereo 4

Figure 50: Prenotazione volo tariffa VoloPlus

4.5 Acquisto

Una volta inseriti e dopo aver cliccato su "Prenota" ci ritroveremo davanti alla pagina di pagamento. Qui l'utente (ospite) può decidere se pagare ora o pagare in seguito. Nel caso decida di pagare in seguito verrà visualizzata una schermata di completamento di prenotazione. Nella sua casella di posta elettronica l'utente dovrebbe trovare una email con un link che servirà per la gestione della sua prenotazione. In questa pagina potrà trovare diverse opzioni:

- Acquista (se non ha ancora acquistato la prenotazione)
- Scarica Biglietti (non visibile fino all'acquisto)
- Cambia data (non visibile fino all'acquisto)

The screenshot shows a web-based travel booking system. At the top, there's a logo for 'Gruppo Aereo 4' featuring a stylized airplane icon. To the right are two buttons: 'Registrati' (Register) and 'Accedi' (Log in). Below the header, the main title 'Completa il tuo acquisto' is centered. On the left, there are input fields for 'NOME' (Name) and 'COGNOME' (Last Name), both currently empty. Below these are fields for 'NUMERO CARTA' (Card Number), 'CVV', and 'SCADENZA' (Expiration Date), also empty. There are three large blue buttons: a top one labeled 'Paga' (Pay), a middle one labeled 'oppure' (or), and a bottom one labeled 'Paga più tardi' (Pay later). On the right side, flight details are listed: 'Biglietti' (149.00€) from 'Milano Malpensa (MXP)' on '2020-04-10 09:30:00' to 'Londra Stansted (STN)' on '2020-04-10 12:30:00' for 'Viaggiatori 1'. Below this, 'Supplementi' (20€) for 'Tariffa VoloPlus' is shown. The total amount 'Totale' is '169.00€'. At the bottom, a copyright notice reads '© Copyright 2020, Gruppo Aereo 4' and a 'Powered by 000webhost' logo is visible.

Figure 51: Completamento della prenotazione

Gestione della prenotazione #202003150000017

18:00 FCO
2020-05-01  19:00 CDG
2020-05-01



© Copyright 2020, Gruppo Aereo 4

Figure 52: Gestione della prenotazione acquistata

Nel caso in cui l'utente decida di pagare subito dovrà immettere la carta di credito e pagare. Nel caso in cui l'utente sia un utente fedele avrà anche la possibilità di pagare attraverso i suoi punti. Quando l'utente avrà pagato, riceverà via email i suoi biglietti e verrà rendirizzato a una pagina di successo

Il tuo acquisto è stato confermato!



© Copyright 2020, Gruppo Aereo 4

Figure 53: Acquisto confermato

4.6 Cambia data

Per cambiare una data l'utente dovrà inserire la nuova data e potrà consultare i voli disponibili. Le operazioni da eseguire al fine di completare il cambio data sono le medesime della prenotazione.

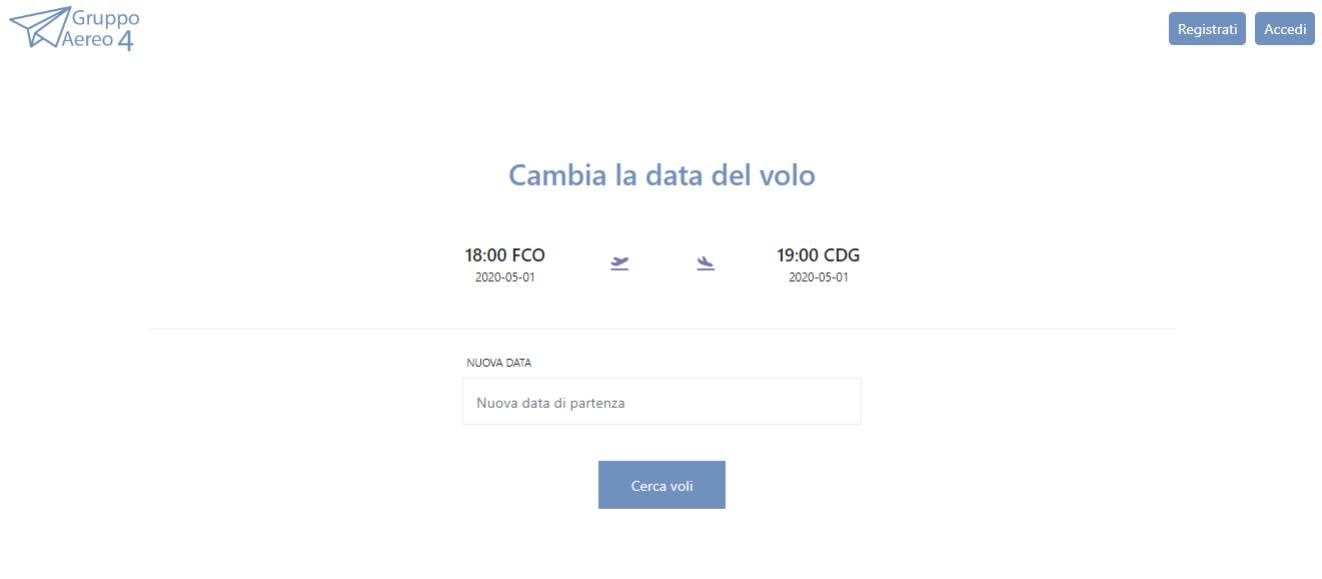


Figure 54: Cambia data

4.7 Area fedeltà

Dopo essersi registrato al programma fedeltà l'utente potrà accedere ad un'area privata nella quale potrà gestire le sue prenotazioni e controllare il saldo dei punti.

4.8 Backend

Abbiamo previsto anche un piccolo pannello di amministrazione per gli impiegati che vogliono inserire/modificare/-cancellare voli e inserire/cancellare promozioni. Questo è accessibile al link <https://gruppoaereo4.000webhostapp.com/public/volo/login> (loggando con user:admin e psw:admin).

#	Da	A	Data e ora partenza	Data e ora arrivo	Miglia	Stato del volo	Operazioni
2020031000204	Milano Malpensa (MXP)	Londra Stansted (STN)	2020-03-20 20:10:00	2020-03-20 10:10:00	1243	ATTIVO	Modifica Cancella
2020031000305	Milano Malpensa (MXP)	Londra Stansted (STN)	2020-02-10 09:20:00	2020-02-10 11:30:00	438	ATTIVO	Modifica Cancella
2020031000406	Milano Malpensa (MXP)	Londra Stansted (STN)	2020-04-04 04:30:00	2020-04-04 19:00:00	707	ATTIVO	Modifica Cancella

Figure 55: Gestione dei voli (view dell'impiegato)

5 Analisi del codice attraverso i tool

5.1 Analisi SonarCloud

Per analizzare il livello del codice abbiamo utilizzato SonarCloud ottenendo i seguenti risultati:

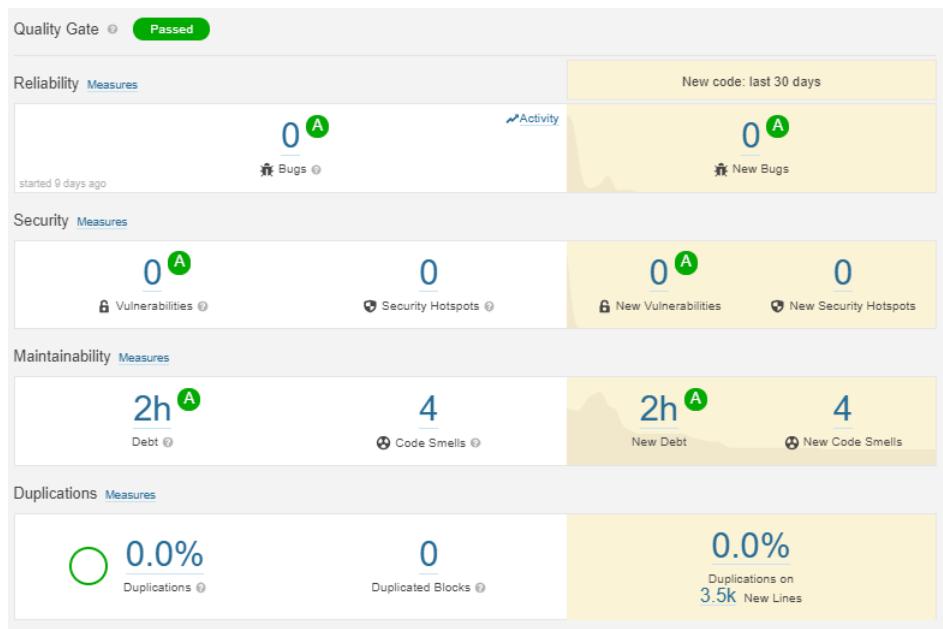


Figure 56: Analisi Sonar

Come si può vedere il codice è esente da bugs e da vulnerabilità e inoltre non contiene duplicati. Sono rimasti però 4 code smells:

- Due di questi indicano classi troppo grandi che non siamo riusciti a snellire.
- Uno indica che ci sono troppi parametri in un metodo che però non siamo riusciti a diminuire poichè sono dati necessari per la creazione di una prenotazione.
- L'ultimo indica che un metodo è troppo complicato da comprendere per via dei troppi if innestati.

5.2 Analisi Understand

Per verificare la presenza di antipattern strutturali nel nostro codice abbiamo utilizzato il tool Understand.

Abbiamo identificato External Hub: la classe DBFacade ha molti collegamenti all'esterno del suo package poichè rappresenta il punto d'accesso al DB.

Inoltre siamo riusciti a estrarre diverse statistiche riguardanti il nostro codice:

Metriche	
Files	68
Program Units	328
Lines	5394
Blank Lines	649
Code Lines	4281
Comment Lines	200
Statements	395

Per quanto riguarda la complessità ciclomatica abbiamo potuto constatare che la classe più grande (in termini di codice) e più complessa è `acquisto.php`:

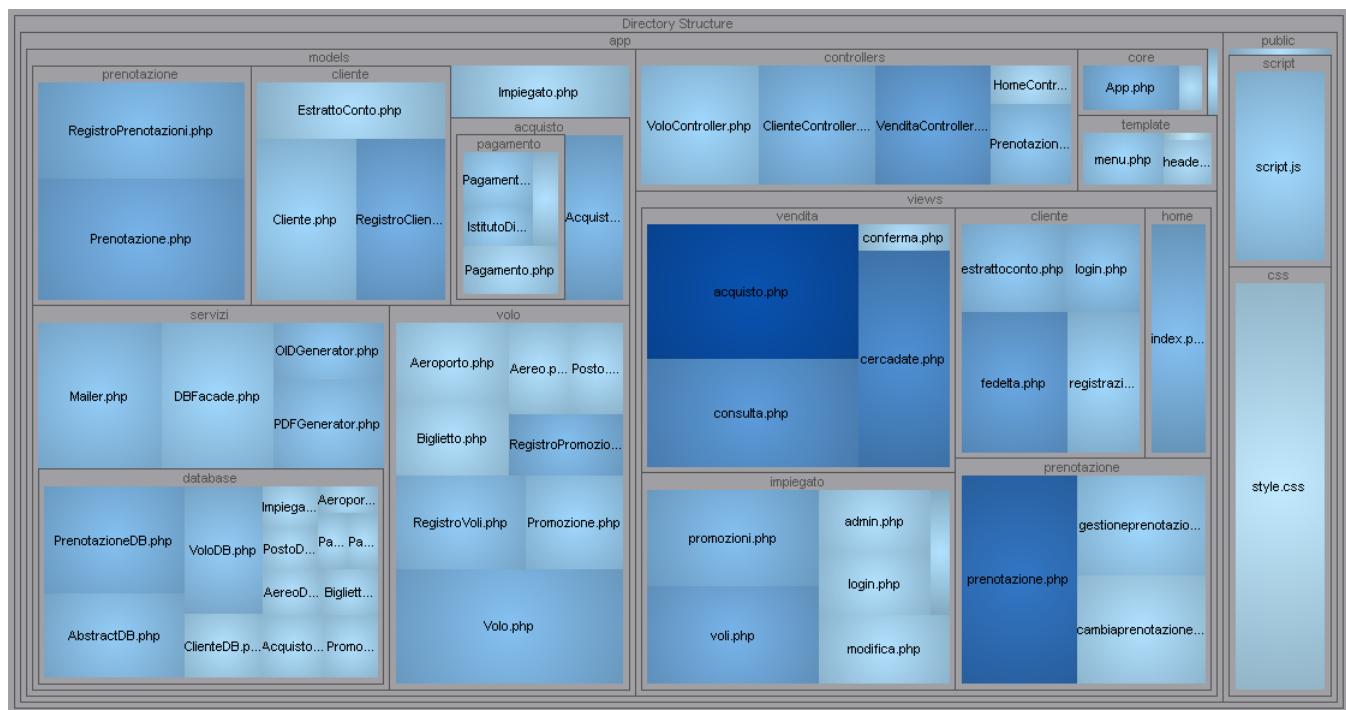


Figure 57: TreeMap complessità ciclomatica

Per quanto riguarda il numero di violazioni come possiamo vedere dalla foto seguente la classe volo.php prevale sulle altre.

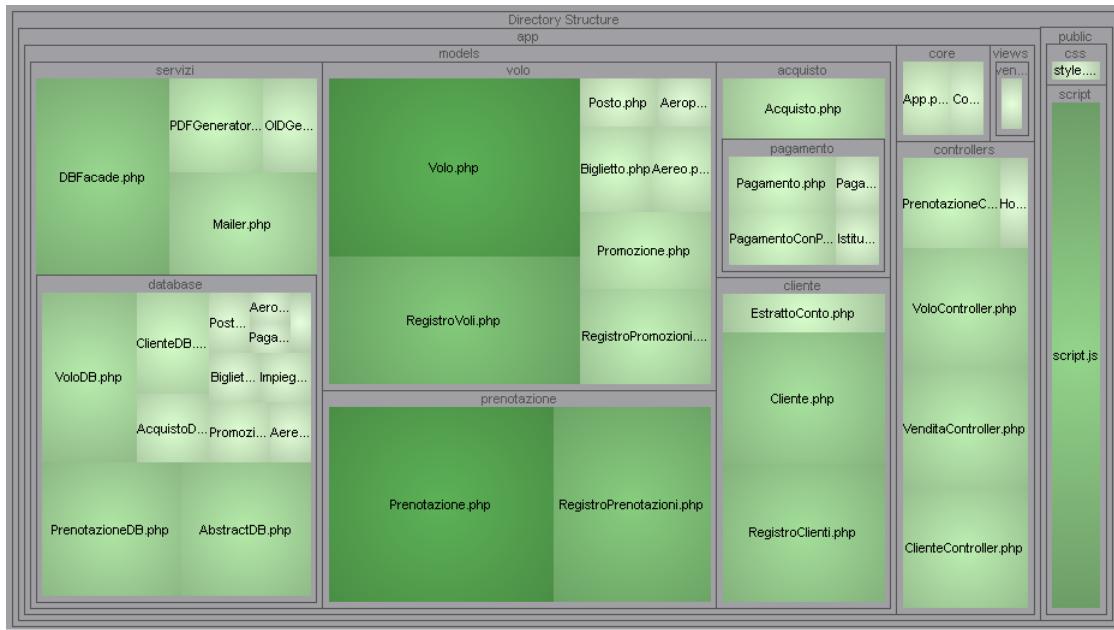


Figure 58: TreeMap numero di violazioni

Andando a controllare il tipo delle violazioni abbiamo notato che riguardano tutte la bassa densità di commenti nel codice.

6 Conclusioni

Grazie a questo progetto abbiamo avuto la possibilità di seguire lo sviluppo di un software dall'inizio alla fine: dalle prime fasi di progettazione, allo sviluppo, fino ad arrivare al testing che precede la release.

Ci riteniamo molto soddisfatti del risultato ottenuto, soprattutto per le conoscenze acquisite da questa esperienza.