PEGASO
Università Telematica

TEMPLATE
PROJECT WORK

Corso di Studio	INFORMATICA PER LE AZIENDE DIGITALI (L-31)
Dimensione dell'elaborato	Minimo 6.000 – Massimo 10.000 parole *(pari a circa Minimo 12 – Massimo 20 pagine)*
Formato del file da caricare in piattaforma	PDF
Nome e Cognome	Fabio Vitaterna
Numero di matricola	0312300511
Tema n. (Indicare il numero del tema scelto):	1
Titolo del tema (Indicare il titolo del tema scelto):	La digitalizzazione dell'impresa
Traccia del PW n. (Indicare il numero della traccia scelta):	4
Titolo della traccia (Indicare il titolo della traccia scelta):	Progettazione dello schema di persistenza dei dati a supporto dei servizi di un'azienda nel settore dei trasporti
Titolo dell'elaborato (Attribuire un titolo al proprio elaborato progettuale):	Progettazione dello schema di persistenza a supporto del servizio di prenotazione di ITA Airways

PARTE PRIMA – DESCRIZIONE DEL PROCESSO

Utilizzo delle conoscenze e abilità derivate dal percorso di studio
(Descrivere quali conoscenze e abilità apprese durante il percorso di studio sono state utilizzate per la redazione dell'elaborato, facendo eventualmente riferimento agli insegnamenti che hanno contribuito a maturarle):

Il dominio applicativo scelto per il presente project work è frutto delle conoscenze acquisite tramite il corso di laurea triennale in Informatica per le aziende digitali (L-31), nonché delle esperienze pratiche maturate in contesti lavorativi.
Le conoscenze necessarie al raggiungimento dell'obiettivo prescelto possono essere classificate tramite le seguenti aree:

- **Fondamenti teorici della progettazione di basi di dati.**
  - Dal punto di vista accademico, l'insegnamento fondamentale è **Basi di Dati**, che fornisce gli strumenti teorici minimi necessarie per la progettazione e modellazione tramite diagrammi E-R volti a favorire l'adozione logico relazione più opportuna rispetto ai requisiti identificati. In particolare, dato che il focus del progetto verte sul modello di persistenza, è stato necessaria una modellazione, tramite forma normali, che garantisca quanto più possibile l'integrità referenziale dei dati da persistere, con la conseguente salvaguardia della consistenza delle informazioni relative a voli, passeggeri, aeromobili e al ciclo di vita della prenotazione, ciclo che culmina con la generazione e vendita dei biglietti e alla conseguente fruizione dei servizi offerti da ITA Airways da parte degli acquirenti.
  - L'esperienza lavorativa, dal 1995 in poi, prima come sviluppatore, poi come analista ed infine come project manager, ha permesso la mediazione tra le conoscenze teoriche acquisite e l'approccio pratico sul campo ottenuto in progetti in cui sono stati nel tempo coinvolti i seguenti DBMS:
    - Sybase.
    - Sql Server e SqlAzure.
    - mySQL.
    - PostgreSQL.
    - MongoDB.
- Metodologie di ingegneria del software a supporto della corretta modellazione dei dati.
  - Dal punto di vista accademico gli insegnamenti di **Algoritmi e strutture dati**, **Ingegneria del software**, **Tecnologie Web** e **Programmazione distribuita e cloud computing** sono stati di supporto per:

        - L'approccio alla progettazione in relazione alla visione dell'intero sistema applicativo che, pur non essendo parte del project work, hanno permesso di stabilire le linee guida della modellazione: un database non deve infatti essere considerato un elemento avulso dal sistema applicativo totale, ma la sua progettazione deve sempre tenere in conto i vincoli e le esigenze dei livelli applicativi a lui superiori; una scarsa considerazione di questo aspetto potrebbe produrre effetti indesiderati al momento di modellare e realizzare tali livelli.
        - L'applicazione dei fondamenti del ciclo del software ha permesso la gestione rigorosa dell'intero processo di realizzazione del progetto.
    - Anche in questo caso, l'esperienza lavorativa è stata di aiuto nella razionalizzazione di tutto il processo.
- Requisiti funzionali derivanti dal settore della logistica e dei trasporti, nonché dall'organizzazione di un'impresa operante in tali ambiti. Sia l'esperienza lavorativa che le nozioni teorici acquisite tramite insegnamenti come **Strategia, organizzazione e marketing** e **Corporate planning e valore d'impresa** hanno permesso la raccolta dei requisiti tramite l'analisi delle dinamiche operative del settore aereo, con particolare riferimento agli standard di ITA Airways.

Fasi di lavoro e relativi tempi di implementazione per la predisposizione dell'elaborato
(Descrivere le attività svolte in corrispondenza di ciascuna fase di redazione dell'elaborato. Indicare il tempo dedicato alla realizzazione di ciascuna fase, le difficoltà incontrate e come sono state superate):

Fase 1: Analisi del dominio e Raccolta dei requisiti

- **Obiettivi:**
    - Studio e analisi del dominio applicativo
    - Definizione dei requisiti
- **Attività svolte**. Sono state ricercate ed analizzate le fonti atte a permettere la comprensione approfondita del settore aereo e, in particolare, della sua applicazione concreta all'interno del core business di ITA Airways: letteratura specialistica del settore aeroportuale, normative da applicare, analisi del portale di ITA Airways e di altre risorse connesse con le funzionalità di tale portale, come ad esempio le strategie di tariffazioni più diffuse, con particolare riferimento a quelle del ticketing dinamico, le informazioni tecniche e logistiche della flotta di aeromobili di ITA Airways, con particolare attenzione alle configurazioni di seating utilizzate. Il deliverable di tale attività è stata la definizione dei requisiti funzionali da utilizzare per le successive fasi progettuali.
- **Durata:** 3 settimane

Fase 2: Progettazione

- **Obiettivi:**
    - **Definizione ad alto livello dell'architettura da utilizzare per il sistema applicativo globale**. Anche se si tratta di un'analisi di alto livello, questo obiettivo non esula dallo scope del project work, anche se non ne è il nucleo fondamentale. Ha lo scopo di supportare la modellazione del layer dati all'interno delle necessarie interazioni architetturali con i layer superiori (logica di business, presentazione dati, etc.). Permette inoltre di focalizzare problematiche che in prima istanza potrebbero sembrare non correlate direttamente con la persistenza, ma che, se trascurate, potrebbero indurre criticità nella fruizione dei servizi messi a disposizione dal sistema applicativo nella sua interezza. Come si vedrà nel proseguo, si consiglierà il disaccoppiamento delle strategie di persistenza (write-model) e di lettura (read-model): una scelta di questo tipo è frutto delle attività legate alla definizione, anche se solo di alto livello, dell'architettura globale. Rientrano in questo obiettivo:
        - L'identificazione del modello architetturale del sistema applicativo di gestione delle prenotazioni di cui per il quale il modello da realizzare fornisce i servizi di persistenza dati.
        - L'identificazione dei pattern di sviluppo, in modo particolare quelli legati alla persistenza, che meglio garantiscono l'implementazione del sistema applicativo nella sua interezza.
    - **Identificazione del DBMS, o dei DBMS, che ospiteranno l'implementazione del modello progettato.** Tipicamente, mentre per i servizi di persistenza (write-model) la scelta cade, a meno di esigenze particolari, su di un DBMS relazionale che per sua natura permette una migliore implementazione dell'integrità e della consistenza dei dati salvati, per i servizi di fruizione dei dati (read-model), la scelta deve orientarsi, soprattutto nel caso di un sistema enterprise quale è un portale di prenotazione di biglietti aerei, su soluzioni più denormalizzate.
    - **Creazione del diagramma E-R**. Il diagramma E-R è un diagramma logico-concettuale delle caratteristiche fondamentali delle entità coinvolte nel modello di persistenza (campi, chiavi, vincoli, indici, etc.) e delle relazioni che ne descrivono e definiscono il mutuo comportamento; una delle caratteristiche salienti di tale modello è l'indipendenza dall'implementazione.
- **Attività svolte.**
    - Per l'identificazione dell'architettura applicativa di alto livello, è stata svolta un'analisi dei requisiti volta ad identificare gli use case più comuni e le problematiche o criticità che potrebbero essere collegate a tali casi. Per non perdere di vista il necessario dimensionamento di un project work

accademico, soprattutto a livello di complessità, è stato molto utile definire, ad alto livello, un MVP cui il modello di persistenza faccia riferimento. Per *Minimun Viable Product* (*MVP*) si intende la versione più semplice e funzionale di un prodotto in grado di poter essere messa sul mercato. Tale concetto è stato introdotto da Franck Robinson nel 2001 e successivamente diffuso da Eric Ries, soprattutto tramite il libro *The Lean Startup.*Tale MVP consta di:
        - Una gestione di un semplice ciclo di vita del processo di prenotazione, con particolare attenzione alle problematiche delle prenotazioni multi-scalo.
        - La possibilità di definire funzionalità di tariffazione dinamica
        - La possibilità di gestire la scelta dei posti relativi ad una prenotazione.
    - Per la scelta dei DBMS da utilizzare, si è fatto riferimento soprattutto all'esperienza pratica acquisita in campo lavorativo.
    - Per arrivare alla creazione del modello E-R si è proceduto, a partire dai requisiti identificati in Fase 1, con l'identificazione e la modellazione:
        - Delle entità e delle loro proprietà.
        - Delle relazioni e delle forme normali necessarie a garantire l'integrità e la consistenza del dato da persistere.
        - Degli indici e chiavi da implementare nella fase successiva
- **Durata:** 3 settimane.

Fase 3: Implementazione

- **Obiettivi:**
    - Implementazione fisica degli oggetti definiti nel modello E-R.
    - Popolamento dati necessari a eseguire dei test di persistenza e di lettura basati sull'MVP previamente indentificato
- **Attività svolte**. Creato i necessari database, sono stati implementati gli script DDL completi sia per la creazione da zero degli oggetti, sia del popolamento necessario per i casi di test. Tali script sono il deliverable di questa fase.
- **Durata**: 1 settimana

Fase 4: Valutazione e documentazione finale

- **Obiettivi:**
    - Valutazione complessiva del risultato delle attività del project work
    - Documentazione
- **Attività svolte:**
    - È stata effettuata un'analisi retrospettiva delle attività svolte, in particolare verificando il livello di soddisfacimento dei requisiti identificati in fase 1.
    - La documentazione consta:
        - Del presente elaborato.
        - Dell'accesso al repository GitHub contenente il codice sorgente.
        - Dell'accesso alla possibilità di interrogare i modelli fisici sviluppati
- **Durata:** 1 settimana.

La durata totale delle attività è stata di 8 settimane.

Risorse e strumenti impiegati

(Descrivere quali risorse - bibliografia, banche dati, ecc. - e strumenti - software, modelli teorici, ecc. - sono stati individuati ed utilizzati per la redazione dell'elaborato. Descrivere, inoltre, i motivi che hanno orientato la scelta delle risorse e degli strumenti, la modalità di individuazione e reperimento delle risorse e degli strumenti, le eventuali difficoltà affrontate nell'individuazione e nell'utilizzo di risorse e strumenti ed il modo in cui sono state superate)**:**

Strumenti impiegati

- **Repository pubblico GitHub.** Il repository di progetto può essere consultato al link https://github.com/fvitaterna-pegaso/schema_persistenza_dati.git. Il repository ha la seguente struttura:
    - **schema_persistenza_dati/code/json** contiene il file *full_reservations.json* utilizzato per popolare l'esempio di read model e altri documenti json utilizzati per vari test.
    - **schema_persistenza_dati/code/model** contiene il file dbml *write_model.dbml* utilizzato per la generazione del modello E-R.
    - **schema_persistenza_dati/code/sql** contiene il codice SQL
    - **schema_persistenza_dati/doc** contiene gli export in formato png e pfd del modello E-R
    - **schema_persistenza_dati/ schema_persistenza_dati/main_document** contiene il presente documento e il suo source word
    - **schema_persistenza_dati/ schema_persistenza_dati/main_document/support** contiene risorse di supporto al presente documento.

- **GitHub Desktop.** Client di HitHub. Sul GitHub di progetto è disponibile il pacchetto di installazione per Microsoft Windows, *GitHubDesktopSetup-x64.exe*, nella directory *schema_persistenza_dati/tools*.
- **dbdiagram.io.** Tool online (https://dbdiagram.io/home) per la modellazione ER. Mette a disposizione un linguaggio dichiarativo, il DBML, che consente la creazione di un modello ER grafico a partire dalla scrittura di codice. Il modello realizzato per il presente project work può essere consultato al link https://dbdiagram.io/d/Write-Model-68ac71d91e7a6119677e68e2
- **Microsoft Sql Server Management.** Client di gestione e sviluppo per SQL Server. Sul GitHub di progetto è disponibile il pacchetto di installazione per Microsoft Windows, *vs_SSMS.exe*, nella directory *schema_persistenza_dati/tools*. Di seguito i parametri di connessione all'istanza Sql Server utilizzata per il progetto:
    - **Server name:** SQL6030.site4now.net
    - **Authentication Type: Sql Server Authentication**
    - **Login:** db_abdf0e_fvitapegaso_admin
    - **Password:** ZdJku6bw
    - **Encryption:** Mandatory
- **Atlas MongoDB Compass.** Client di gestione e sviluppo per MongoDB. Sul GitHub di progetto è disponibile il pacchetto di installazione per Microsoft Windows, *mongodb-compass-1.46.10-win32-x64.msi*, nella directory *schema_persistenza_dati/tools*. La connection string per connettersi all'istanza MongoDB utilizzata per il progetto è *mongodb+srv://fabiovitaterna_db_user:ZdJku6bw@cluster0.nih5htg.mongodb.net/*
- L'AI **Microsoft Compass** è stata utilizzata per il reperimento dei dati geografici e di servizio (continenti, nazioni, città, aeroporti).

Risorse:

- **Dominio del trasporto aereo civile**
    - ENAC - Regolamento di Scalo
    - **Williams, K.** (2017). Dynamic Airline Pricing and Seat Availability. Yale University
    - Portale di ITA Airways
    - Portale di IATA
    - Portale di ICAO
    - Portale Aerolopa per la consultazione degli allestimenti degli aeromobili
- **Risorse tecniche:**
    - **Rise, E.** (2011). The Lean Startup. Portfolio Penguin
    - **Meyer, B.** (1988). Object-Oriented Software Construction. New York: Prentice Hall.
    - **Young, G.** (2010). CQRS Documents. Disponibile su: https://github.com/keyvanakbary/cqrs-documents
    - **Fowler, M.** (2011). CQRS.Martin Fowler's Blog .
    - **Vernon, V.** (2013). Implementing Domain-Driven Design. Boston: Addison-Wesley Professional.
    - **Richardson, C.** (2018). Microservices Patterns: With Examples in Java. Manning Publications.

PARTE SECONDA – PREDISPOSIZIONE DELL'ELABORATO

Obiettivi del progetto
(Descrivere gli obiettivi raggiunti dall'elaborato, indicando in che modo esso risponde a quanto richiesto dalla traccia):

Obiettivo del progetto è progettare nel modo più rigoroso e completo possibile lo schema di persistenza dati richiesto dalla traccia, con particolare riferimento al sistema di prenotazione di ITA Airways.
Il modello di persistenza da progettare deve garantire la possibilità di sviluppare un sistema applicativo di prenotazione nel campo del trasporto aereo civile; tale sistema dovrà basare la sua gestione dei dati sui servizi erogati dal database implementato in base al modello progettato. Il modello da progettare dovrà quindi rispondere alle esigenze di conformità con i requisiti operativi del mondo del trasporto aereo e di scalabilità sia del modello stesso sia della fruizione dei data service da parte dell'intero sistema applicativo.

Analisi dei requisiti.
Il dominio dei servizi in ambito trasporto aereo presenta un elevato grado di complessità funzionale; ciò è dovuto alla coesistenza di esigenze prioritarie di sostenibilità economica del business e di un quadro normativo fortemente vincolante.
Di qui la necessità di un'analisi dei flussi informativi che regolano l'erogazione di servizi di prenotazione e vendita da parte di una compagnia aerea; è stato poi necessario analizzare le interazioni di ITA Airways con altre compagnie, con i fruitori dei servizi e con gli attori che permettono la realizzazione dell'erogazione, in modo da indentificare le criticità, da punto di vista della gestione dei dati, dell'intero dominio funzionale.

Modellazione conforme agli scenari descritti dai requisiti

Identificati e analizzati i requisiti, il passo seguente è stato modellare la realtà da essi descritta, ovvero integrare i diversi aspetti tipici del dominio del trasporto aereo, come la gestione delle prenotazioni, il supporto alla vendita dei biglietti, la profilazione dei passeggeri, la possibilità di configurare, all'interno di una prenotazione, itinerari multi-scalo.

Progettazione architetturale

Si è reso poi necessario identificare l'architettura del sistema al fine di implementare quanto modellizzato. È stata effettuata quindi un'analisi comparativa tra varie opzioni di DBMS e pattern di sviluppo al fine di garantire quanto più possibile, anche se all'interno di un project work accademico, un'implementazione che soddisfacesse i requisiti identificati e gli scenari modellati, tenendo presente la natura enterprise di un sistema di prenotazione per il trasporto aereo. La soluzione individuata è stata infine vagliata per stabilirne il grado di idoneità rispetto alle esigenze puntuali di questo tipo di sistema applicativo come, ad esempio, la gestione di volumi transazionali rilevanti, l'esigenza di alta disponibilità e la necessità di performance appropriate, supponendo do voler arrivare a tempi di risposta dell'ordine di pochi secondi per le operazioni critiche.

Contestualizzazione

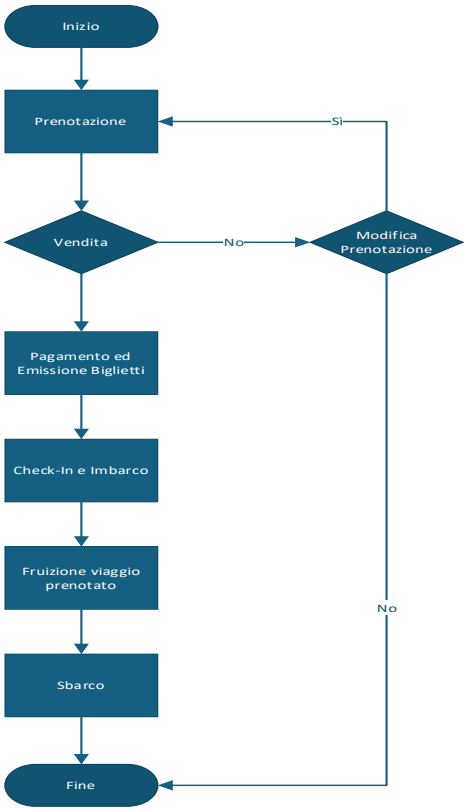
(Descrivere il contesto teorico e quello applicativo dell'elaborato realizzato):

Il dominio del trasporto aereo civile

La complessità della gestione operativa di una compagnia aerea risiede soprattutto nella molteplicità delle componenti di dominio da integrare e armonizzare.

Il core business dell'attività risiede nella **gestione passeggeri** in quanto fruitori dei servizi e fonte primaria delle condizioni di sostenibilità delle attività della compagnia. La gestione dei passeggeri opera all'interno di un percorso che sarà definito **ciclo di vita della prenotazione**.

Una versione semplificata di tale ciclo è raffigurata nel diagramma che segue.



Per gestire tale ciclo di vita si rende necessaria si rende necessaria l'interazione di numerose componenti tutte confluenti nell'obiettivo primario dell'ottimizzazione dei ricavi:

- Gestione delle risorse, materiali e fisiche
- Tariffazione, nella maggior parte dei casi, dinamica.
- Gestione della disponibilità in tempo reale dei posti.

Un vincolo ulteriore consiste nella necessità dell'interoperabilità tra le diverse compagne e tra le compagnie e le infrastrutture locali, come ad esempio gli scali; tale interoperabilità è garantita da standard internazionali, tra cui il

principale è quello legato a IATA.
Di seguito vengono ora dati cenni di alcuni standard di processo che saranno utilizzati nel modello implementato e di alcuni requisiti base che tale modello dovrà soddisfare.
Il **Passenger Name Record** (di qui in avanti **PNR**) è un codice alfanumerico di 6 caratteri che identifica una prenotazione aerea che può includere uno o più passeggeri che viaggiano insieme su di un itinerario che si realizza tramite n voli su n tratte. Il PNR è univoco nel contesto di un dato sistema di prenotazione, ma all'interno di tale contesto non gode di univocità rispetto al singolo passeggero o al singolo volo.
L'**Electronic Ticket** (di qui in avanti ETKE) è un documento digitale con significato di titolo di viaggio per i voli aerei; è un codice alfanumerico di 13 caratteri, di cui i primi 3 rappresentano univocamente a livello IATA la compagnia aerea emettitrice del biglietto (per esempio 055 per ITA Airways). È associato ad una prenotazione e ad un passeggero e al suo itinerario all'interno della prenotazione stessa.
Un **Flight Code** identifica univocamente un volo a livello di compagnia aerea per un dato giorno. È un codice alfanumerico la cui lunghezza può variare da un minimo di 3 caratteri ad un massimo di 6, così composto:

- I primi due caratteri identificano la compagnia aerea, attraverso il codice IATA (per esempio AZ per ITA Airways).
- I restanti caratteri identificano lo specifico volo.

Un sistema informatico di prenotazione in ambito civile, come quello per cui si fornisce la progettazione del modello di persistenza, è denominato **Global Distribution System** (di qui in avanti **GDS**). Come esposto nelle tabelle che seguono, è possibile identificare i principali requisiti cui un GDS deve sottostare; all'interno di tali requisiti è possibile poi identificare quelli che influiscono direttamente sulla progettazione del modello di persistenza.

Codice	Requisito	Descrizione	Rilevanza diretta per modello di persistenza
R1	Gestione prenotazioni	Creazione, modifica, cancellazione e conferma di prenotazioni multi-tratta	Sì
R2	Persistenza dati	Salvataggio permanente e sicuro di tutte le transazioni	Sì
R3	Integrazione con compagnie aeree	Interoperabilità tramite API o protocolli standard	No
R4	Supporto multi-segmento	Gestione di itinerari con tratte scale, cambi.	Sì
R5	Emissione biglietti elettronici (ETKT)	Generazione e gestione di biglietti elettronici con codice IATA	Sì
R6	Tracciamento PNR	Ogni prenotazione deve essere identificata tramite un PNR univoco	Sì
R7	Alta disponibilità	Il sistema deve essere operativo 24/7 con sistemi di tolleranza ai guasti e strategie di disaster recovery	No
R8	Scalabilità	Capacità di gestire rilevanti volumi di transazioni simultanee	Sì
R9	Sicurezza	Crittografia dei dati sensibili	Sì
R10	Performance	Tempi di risposta compatibili con la fruibilità enterprise richiesta dal prodotto	Sì
R11	Auditabilità	Tracciabilità delle operazioni	Sì
R12	Conformità	Rispetto delle normative IATA, PCI-DSS, GDPR	Sì

Infine, occorre dare alcuni dettagli circa le strategie di **tariffazione** dinamica, che negli ultimi anni ha progressivamente assunto grande importanza soprattutto con la comparsa delle cosiddette compagnie *low-cost*. Per tariffazione dinamiche si intendono le strategie messe in atto dalle compagnie aeree i cui obiettivi fondamentali sono:

- Garantire la vendita del maggior numero di biglietti entro la data di partenza del volo per minimizzare il numero di posti invenduti. Questo obiettivo viene raggiunto facendo variare le tariffe, dinamicamente e in tempo reale, in funzione ad esempio degli itinerari, delle disponibilità di posti sul volo, della data di acquisto e della sua relazione temporale con la data di partenza.
- Massimizzare il più possibile la redditività e la sostenibilità della vendita del volo. Questo obiettivo viene

raggiunto aggiungendo servizi a pagamento non inclusi nel semplice acquisto del volo. È possibile distinguere tra:

- Servizi legati ai tipi di tariffa. Questa strategia comporta la frammentazione delle tariffe base in una molteplicità di sotto-tariffe, ognuna della quali configura quali caratteristiche del volo siano gratuite o a pagamento. L'esempio più evidente è la possibilità per il passeggero di scegliere il posto: in alcune tariffazioni sarà richiesto al passeggero se pagare per tale scelta o accettare l'assegnazione automatica, gratuita, operata da un algoritmo specifico.
- Servizi addizionali, non necessariamente facenti parte del core business della compagnia aerea. Possono essere sviluppati tramite partnership con aziende di altri settori, ad esempio con provider di noleggio di autoveicoli.

Una possibile strategia aggiuntive, e trasversale rispetto alle precedenti, è la tariffazione attraverso la profilazione dei clienti. Da questo punto di vista riveste particolare importanza l'acquisizione dei dati dei passeggeri che supportano questo tipo di operazione.

Scelte architetturali

Il settore del trasporto aereo è caratterizzato da volumi transazionali elevati, con picchi di transito dati significativi in determinate finestre temporali; inoltre, tra le operazioni di lettura e scrittura esiste una relazione fortemente asimmetrica.

Tali considerazioni e quanto descritto finora delineano un'architettura decisamente orientata ai modelli enterprise tipici del mondo web.

Dato tale contesto, considerando anche le caratteristiche tipiche di un portale di prenotazione, si è optato per l'adozione del pattern **Command Query Responsability Segregation** (di qui in avanti **CQRS**).

CQRS si basa fondamentalmente sui principi del **Domain-Driven Design** (di qui in avanti **DDD**), formalizzato nel 2003 da Eric Evans. Alcune caratteristiche di DDD hanno notevoli punti di contatto con CQRS:

- La separazione delle responsabilità a livello di dominio, che viene implementata da CQRS attraverso la distinzione logica e fisica tra write model (persistenza) e read model (fruizione dei dati).
- Il concetto di aggregato di dominio.
- L'astrazione dell'accesso ai dati.
- Il concetto di servizi di dominio, che si riflette nei query services di CQRS.

D'altra parte, nel 1988, Bertrand Mayer aveva introdotto, nel contesto della programmazione orientata ai progetti, il **Command-Query Separation** (di qui in avanti **CQS**), stabilendo la distinzione tra:

- **Command**, ovvero metodi in grado di modificare lo stato di un oggetto.
- **Query,** metodi il cui unico scope è ritornare informazioni.

CQRS, formalizzato da Greg Young nel 2010 estende questi principii applicandoli alla separazione fisica tra i modelli di persistenza (write models) e i servizi di fruizione dei dati, opportunamente aggregati (read models).

Ulteriori contributi all'implementazione definitiva di CQRS sono stati poi dati da Martin Fowler negli articoli *CQRS* del 2011 ed *Event Sourcing* del 2015, nonché da altri specialisti del settore, come Udi Dahan, Vaughn Vernon e Chris Richardson.

La ricerca accademica ha evidenziato la validità del pattern proprio su domini caratterizzati da elevati volumi transazionali, asincronia tra volumi operazioni di lettura e scrittura e necessità di alti livelli performance. Le caratteristiche evidenziate dall'analisi del dominio del trasporto aereo condotta, giustificano l'utilizzo di CQRS:

- Le operazioni di scrittura, come ad esempio creazione e modifica e cancellazione di prenotazioni o aggiornamento dello stato dei voli, sono una percentuale piccola del carico totale, che è invece dominato dai servizi di lettura, come ad esempio la consultazione di una o più prenotazione già esistenti, la ricerca dei voli disponibili in base a un determinato itinerario richiesto, la visualizzazione della disponibilità di voli secondo determinate condizioni, come la tariffa o gli orari di partenza e arrivo.
- Si ha asimmetria tra scrittura e lettura anche dal punto di vista dei requisiti tecnici. Le operazioni di scrittura devono garantire l'integrità e la consistenza del dato persistito, devono essere sottoposte a vincoli dettati dalle regole di business, devono sottostare a un'implementazione forte del modello ACID a livello delle transazioni per garantirne strettamente l'affidabilità.
- Grazie alla segregazione introdotta da CQRS, è possibile progettare il modello di persistenza (oggetto del presente elaborato) privilegiando l'integrità transazionale e relazionale; per l'implementazione del read model (l'elaborato ne presenterà un piccolo esempio) sarà invece possibile mettere l'accento sulla denormalizzazione e sull'aggregazione del dato, caratteristiche che permettono l'ottimizzazione dei servizi di lettura dei dati sia dal punto di vista delle prestazioni che da quello della fruizione del dato stesso.

D'altra parte, l'utilizzo di un pattern classico privo di separazione, sia logica che fisica, tra scrittura e lettura costituirebbe un compromesso tra i due gruppi di requisiti esposti, compromesso che rischierebbe di non soddisfare pienamente né l'uno né l'altro.

I vantaggi derivanti dall'utilizzo di CQRS non sono solamente puramente applicativi:

- La segregazione fisica, oltre che logica, permette anche la gestione indipendente dei carichi in modo proporzionale alle reali esigenze della scrittura e della lettura, migliorando in questo modo le possibilità di scalabilità della soluzione, orientandola verso un modello orizzontale, ovvero non basata sull'aumento delle risorse hardware ma sulla possibilità di distribuire in modo flessibile le componenti applicative.
- La segregazione permette l'implementazione di strategie dedicate per ognuno dei due modelli. Un esempio è la possibilità di implementare per il read model strategie di caching sofisticate, di difficile utilizzo nel caso lettura e scrittura non fossero disaccoppiate; uno degli use case che possono chiarire questo punto è per esempio la ricerca di voli disponibili, una delle operazioni più frequenti e potenzialmente costose a livello di fruizione di dati.
- CQRS può migliorare anche la manutenibilità del sistema facilitando sia gli interventi correttivi sia quelli di manutenzione evolutiva. Questo perché, come del resto in tutti i moderni pattern orientati agli oggetti, l'architettura applicativa del sistema non è monolitica, ma distribuita, aprendo la possibilità di implementazioni dei livelli superiori allo strati dati, ad esempio, tramite microservizi.

La scelta del DBMS da utilizzare per l'implementazione del **write model**, ovvero il modello di persistenza, la parte core della traccia, è ricaduta su Microsoft Sql Server. Tra le varie caratteristiche che ne giustificano l'adozione, le più importanti sono:

- I meccanismi avanzati di gestione della concorrenza, attraverso le feature di **row-level locking** e **snaphot isolation**, che rivestono un ruolo di primo piano nella gestione di operazioni transazionali ad alta frequenza tipiche dei portali web di e-commerce e dei sistemi di prenotazione.
- L'esigenza di continuità operativa del sistema applicativo è supportata dal meccanismo di **Always On Availability Groups** garantisce l'alta disponibilità dei dati.
- Il **disaster recovery** è garantito dalle funzionalità di backup integrate di Sql Server.

Ovviamente il DBMS selezionato non era l'unico candidato appropriato; alternativa valide avrebbero potuto essere la versione enterprise di **mySQL**, **PostgreSQL** o **Oracle**, motori con funzionalità enterprise dello stesso livello di Sql Server. Tuttavia, la familiarità acquisita negli anni attraverso la pratica professionale, come detto precedentemente, è stato un ulteriore criterio che ha guidato la scelta, con l'obiettivo di ottimizzare i tempi di implementazione.

Il **read model** deve garantire alte prestazioni di lettura e flessibilità, data la complessità delle strutture dati da rendere disponibili. Si è ritenuto che un approccio relazionale non sarebbe stata la scelta ottimale per rispettare tali requisiti: la scelta è caduta sui **DBMS NoSql** e in particolare su **MongoDB**; la struttura a documenti e l'uso di JSON permettono una fruizione migliore di strutture dati complesse, come ad esempio la prenotazione di viaggi multi-tratta, uno degli use case più importanti di tutto il progetto. I principali vantaggi di tale scelta sono:

- Eliminazioni di operazioni di join potenzialmente costose.
- Rappresentazione JSON nativa per strutture gerarchiche.
- Horizontal sharding per scalabilità enterprise.
- Flessibilità schema per evoluzione senza downtime.
- Indicizzazione flessibile.
- Ricerca full-text avanzata

L'alimentazione del Read Model potrà essere implementata, a livello di strato applicativo di business logic, mediante un l'implementazione di un sistema di event streaming basato su un bus, ad esempio RabbitMQ. Di seguito una delle possibili strategie di implementazione.
Stabilite le strutture dei documenti JSON che saranno ospitati da MongoDB, in corrispondenza di ogni operazione, o di ogni gruppo di operazioni, di persistenza sarà scatenato un evento che sarà gestito da due code di RabbitMQ, una utilizzata per la trasmissione dei dati associati all'evento e l'altra per la gestione di eventuali errori. Sarà poi compito di un microservizio che sarà sottoscritto alla fruizione di tali code, implementare una cosiddetta projection che modificherà i documenti JSON contenenti i dati del read model che sono stati modificati dal command di persistenza.

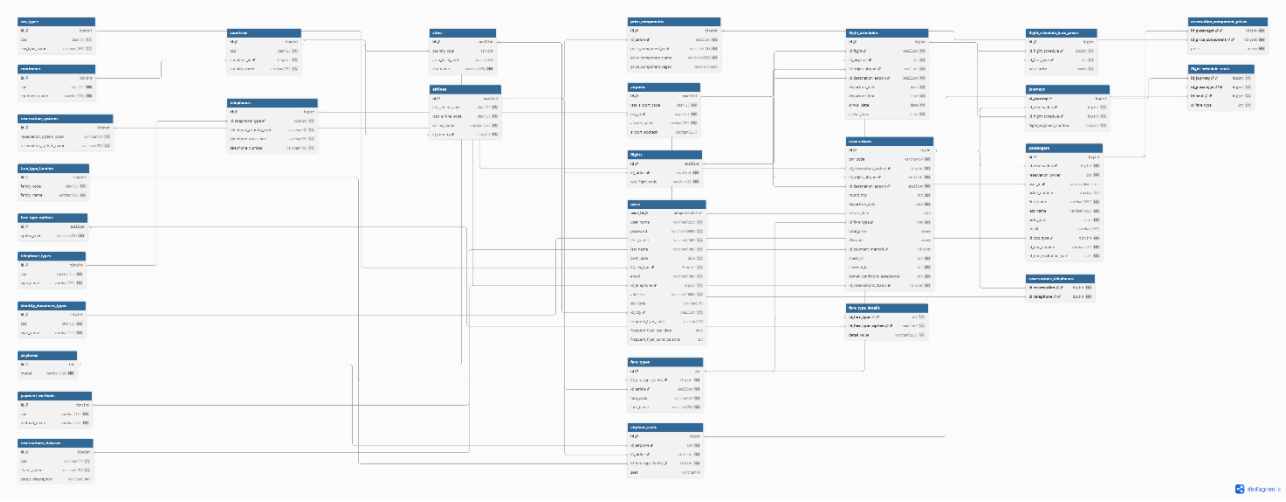
Descrizione dei principali aspetti progettuali
(Sviluppare l'elaborato richiesto dalla traccia prescelta):

Modello E-R della persistenza dati (write model)
Nell'immagine che segue è mostrato il modello E-R progettato. Per una migliore visualizzazione, è possibile consultare il modello nel repository GitHub di progetto, come detto precedentemente nella sezione 1.c, nella directory *schema_persistenza_dati/doc* in cui vengono messi a disposizione i seguenti file:

- *Write Model.png*
- *Write Model.pdf*

Inoltre, nella directory *schema_persistenza_dati/code/model* è messo a disposizione il file *write_model.dbml* contenente

il codice dichiarativo dbml esportato da *dbdiagram.io*, il tool utilizzato per la modellazione.



Categorizzazione delle entità in relazione al dominio del trasporto aereo.
Di seguito una possibile categorizzazione delle entità modellate:

- **Entità Geografiche**, che gestiscono informazioni geografiche e di localizzazione:
  - **continents** - Gestione continenti
  - **countries** - Gestione nazioni con riferimento ai continenti
  - **cities** - Gestione città con riferimento alle nazioni
  - **airports** - Gestione aeroporti con localizzazione nelle città e codice IATA degli stessi.

- **Entità Operative**, che gestiscono le operazioni core del trasporto aereo:
  - **airlines** - Compagnie aeree
  - **flights** - Definizione voli (codici IATA)
  - **airplanes** - Aeromobili della flotta
  - **airplane_seats** - Configurazione posti per aeromobile
  - **flight_schedules** - Programmazione operativa dei voli
  - **journeys** - Tratte specifiche di ogni prenotazione
  - **flight_schedule_seats** - Assegnazione posti per tratta specifica
  - **passengers** - Gestione passeggeri e documenti
  - **users** - Utenti registrati del sistema

- **Entità di Servizio**, a supporto di funzionalità tecniche e di sistema:
  - **sex_types** - Tipologie di genere
  - **telephone_types** - Tipologie di numeri telefonici
  - **telephones** - Numeri di telefono
  - **identity_document_types** - Tipologie documenti di identità
  - **payment_methods** - Metodi di pagamento disponibili
  - **reservation_systems** - Sistemi di prenotazione
  - **reservations_statuses** - Stati del ciclo di vita delle prenotazioni
  - **reservations_telephones** - Associazione prenotazioni-telefoni

- **Entità Commerciali**, che gestiscono aspetti commerciali:
  - **fare_type_families** - Famiglie di tariffe (Economy, Business, etc.)
  - **fare_types** - Tipologie tariffarie specifiche per compagnia
  - **fare_type_options** - Opzioni configurabili nelle tariffe
  - **fare_type_details** - Configurazione delle tariffe
  - **price_components** - Componenti del prezzo del biglietto (tasse, fuel surcharge, etc.)
  - **flight_schedule_base_prices** - Prezzi base per volo e tariffa
  - **reservations** - Prenotazioni
  - **reservation_component_prices** - Scomposizione prezzi delle prenotazioni

Considerazioni su alcune entità di particolare importanza

L'entità **airports** modella l'infrastruttura aeroportuale per permettere la creazione di itinerari di volo associati a una o più prenotazione. È collegata gerarchicamente alle entità geografiche tramite relazione 1 a molti con l'entità **cities**; **airports** introduce inoltre nel modello il vincolo di univocità dell'aeroporto, tramite l'attributo *iata_airport_code* che tiene traccia del codice aeroportuale secondo lo standard IATA.

L'entità **airlines** modella il concetto di compagnia aerea. È collegata gerarchicamente alle entità geografiche tramite relazione 1 a molti con l'entità **countries**, rendendo tra l'altro possibile lo sviluppo di report per analisi di mercato a livello sia di singola nazione che internazionale. Inoltre, introduce nel modello il vincolo di univocità della compagnia aerea con i seguenti attributi, ciascuno dei quali può essere utilizzato autonomamente per l'implementazione del vincolo:

- *iata_airline_code*, secondo lo standard IATA
- *icao_airline_code*, secondo lo standard ICAO

I voli sono modellati tramite l'entità **flights** che definisce un volo astratto introducendone il codice IATA e l'entità **flight_schedules** che concretizza i voli che fanno riferimento ad un dato codice IATA, modellando l'univocità tramite tramite il suddetto codice e la data di partenza. L'entità **flight_schedules** contiene le informazioni necessarie per aggiungere un volo all'itinerario di una prenotazione.

L'entità **reservations** modella le informazioni di prenotazione comunica a tutti i passeggeri appartenenti ad essa. Attributi rilevanti dell'entità sono:

- *pnr_code,* che modella il codice univoco di prenotazione.
- *round_trip*, responsabile dell'identificazione del tipo di viaggio (andata/ritorno o solo andata).
- *departure_date* e *return_date,* le date di riferimento della prenotazione
- *id_origin_airport* e *id_destination_airport*, che, implementando relazioni con l'entità **airports,** modellano l'inizio e la fine geografica della prenotazione.
- *id_reservations_systems* tiene traccia del riferimento al sistema di prenotazione utilizzato.
- *id_fare_type* modella la relazione con l'entità **fare_types**. Come si dettaglierà nel proseguo, questa relazione è importante ai fini di permettere l'eventuale configurazione, da parte della compagnia aerea, di strategie di tariffazione dinamica.
- *id_reservation_status*, in relazione con **reservation_statuses**, *check_in* e *checked_in* tengono traccia dello stato della prenotazione nel ciclo di vita della stessa.

L'entità **passengers** modella il dettaglio della prenotazione dal punto di vista dei dati dei passeggeri coinvolti nella prenotazione. La relazione con **reservations** è modellata tramite l'attributo *reservation_id*. L'acquisizione dei dati del passeggero è fondamentale sia per soddisfare i requisiti normativi di dominio che per implementare la profilazione cliente da parte della compagnia aerea. Si noti infine che la modellazione di passengers supporta, tramite l'attributo *user_id*, sia utenti registrati, in modo da permettere il possibile accesso a strategie di fidelizzazione e di profilazioni avanzati, che di tipo *ospite*, interessati solamente a transazioni rapide, ma che vengono comunque profilati.

L'entità **journeys** è di particolare importanza perché supporta l'implementazione di itinerari multi-tratta associati alla prenotazione.

- L'attributo *id_reservation* modella una relazione 1 a molti con **reservations** (journeys nel lato molti). In questo modo ad una prenotazione è possibile associare n tappe di un itinerario multi-scalo.
- L'attributo *id_flight_schedule* modella una relazione 1 a molti con **flight_schedules** (journeys nel lato molti), collegando ogni tappa dell'itinerario associato alla prenotazione con un volo pianificato.
- L'attributo *flight_segment_number* modella l'ordine di svolgimento della singola tappa all'interno dell'itinerario. L'attributo varrà 1 in corrispondenza del volo con partenza dall'aeroporto di origine della prenotazione e n, dove n è l'ordinale dell'ultima tappa del viaggio, in corrispondenza del volo con arrivo nell'aeroporto di destinazione della prenotazione.
- Si noti che l'entità non ha relazione diretta con **passengers**, in quanto uno dei requisiti è che per tutti i passeggeri associati alla stessa prenotazione ci sia univocità di itinerario.

Il gruppo di entità relazionate gerarchicamente a **fare_types** è molto importante perché supporta la configurazione delle tariffe e soprattutto la possibilità di implementare tariffazioni dinamiche da parte della compagnia aerea.
L'entità **fare_type_families** supporta i macro-gruppi tariffari che la quasi totalità delle compagnie utilizza. Tipicamente sono:

- Economy
- Premium, normalmente utilizzata solamente per voli a lunga durata.

- Business

L'entità **fare_types** modella e identifica i piani tariffari: si tratta del set di tariffe proposte al cliente nel processo di prenotazione.
L'entità **fare_type_options** modella e identifica i possibili servizi, gratuiti o a pagamento, associabili alle tariffe. Esempi di queste opzioni, gratuite o a pagamento, sono la scelta del posto, le dimensioni e il numero di colli che è possibile inviare in stiva, la possibilità di modificare la prenotazione dopo l'acquisto.
fare_type_details è un'entità di tipo many-to-many che tramite le relazioni con **fare_types** e **fare_types_options** permette la personalizzazione flessibile di ogni piano tariffario.
L'entità **flight_schedule_prices**, tramite le relazioni con **flight_schedules** e **fare_types** permette di fissare il prezzo base di ogni volo schedulato: tale prezzo sarà il punto di partenza per le strategie di tariffazione dinamica; il modello esposto ne permette sia la dinamicizzazione del prezzo tramite le opzioni associate ad ogni piano tariffario, sia l'implementazione tramite parametri temporali e di disponibilità dei posti. Infine, le entità **passengers** e **users** permettono anche la dinamicizzazione della tariffa tramite la profilazione cliente.

Si noti che l'entità **fare_type_families** entità entra in gioco anche nella configurazione degli allestimenti dei diversi modelli di aeromobili facenti parte delle flotte delle compagnie aeree; ogni compagnia richiede ai fornitori un allestimento personalizzato per ogni modello appartenente alla sua flotta. L'entità **airplanes** tiene traccia dei vari modelli di aeromobili, per i quali ogni compagnia configura il suo allestimento personalizzato.
La personalizzazione dell'allestimento è modellata dall'entità **airplane_seats:**

- L'attributo seat identifica il singolo posto all'interno dell'allestimento. Normalmente è un codice alfanumerico composto da un numero che identifica la fila e da un carattere che identifica la posizione del posto all'interno della fila stessa.
- La relazione con **airplanes** tiene traccia che si sta configurando la personalizzazione del modello corrispondente all'attributo *id_airplane* di **airplane_seats**.
- La relazione con **airlines** tiene traccia che si sta configurando la personalizzazione del modello per la compagnia aerea corrispondente all'attributo *id_airlane* di **airplane_seats**.
- La relazione con **fare_type_families** identifica a quale macro-gruppo tariffario, identificato dall'attributo *id_fare_type_family* di **airplane_seats,** apparterrà il posto identificato da seat.

Implementazione fisica della persistenza dati (write model)
Il file *write_model.sql*, situato nella directory *schema_persistenza_dati/code/sql* del GitHub di progetto, contiene le DDL per la creazione del modello fisico del write model, basato sul modello E-R precedentemente commentato. L'esecuzione di tutti gli script, seguendo l'ordine presente nel file, permette la creazione da zero del modello senza errori anche in presenza di una precedente implementazione, che viene ovviamente eliminata.
La consultazione di *write_model.sql* consente di verificare che l'implementazione rispecchia il modello E-R. Nel proseguo saranno forniti degli esempi di codice e alcune considerazioni su quanto implementato.

Esempi di codice DDL
Vengono riportati alcuni esempi delle DDL di creazione degli oggetti fisici del database

Tabella reservations

```
--CREAZIONE TABELLE reservations
CREATE TABLE [reservations] (
   [id] bigint PRIMARY KEY IDENTITY(1,1),
   [pnr_code] varchar(6) NOT NULL,
   [id_reservation_system] tinyint NOT NULL,
   [id_origin_airport] smallint NOT NULL,
   [id_destination_airport] smallint NOT NULL,
   [round_trip] bit not null,
   [departure_date] date NOT NULL,
   [return_date] date,
   [id_fare_type] int NOT NULL,
   [total_price] money,
   [discount] money,
   [id_payment_method] tinyint,
   [check_in] bit NOT NULL DEFAULT (0),
   [checked_in] bit NOT NULL DEFAULT (0),
   [owner_conditions_acceptance] bit NOT NULL,
   [id_reservations_status] tinyint NOT NULL
)
GO

--CREAZIONE INDICE UNIQUE SUI CAMPI id_reservation_system e pnr_code di reservations
CREATE UNIQUE INDEX [reservations_index_1] ON [reservations] ("id_reservation_system", "pnr_code")

GO

--CREAZIONE FOREIGN KEY DELLA TABELLA reservations
ALTER TABLE [reservations] ADD CONSTRAINT [reservation_to_system_reservations] FOREIGN KEY ([id_reservation_system]) REFERENCES [reservation_systems] ([id])
GO

ALTER TABLE [reservations] ADD CONSTRAINT [origin_airport_to_reservations] FOREIGN KEY ([id_origin_airport]) REFERENCES [airports] ([id])
GO

ALTER TABLE [reservations] ADD CONSTRAINT [origin_destination_to_reservations] FOREIGN KEY ([id_destination_airport]) REFERENCES [airports] ([id])
GO

ALTER TABLE [reservations] ADD CONSTRAINT [fare_type_to_reservations] FOREIGN KEY ([id_fare_type]) REFERENCES [fare_types] ([id])
GO

ALTER TABLE [reservations] ADD CONSTRAINT [payment_method_to_reservations] FOREIGN KEY ([id_payment_method]) REFERENCES [payment_methods] ([id])
GO
```

Tabella passengers

```sql
--CREAZIONE TABELLA passengers
CREATE TABLE [passengers] (
   [id] bigint PRIMARY KEY IDENTITY(1, 1),
   [id_reservation] bigint NOT NULL,
   [reservation_owner] bit NOT NULL,
   [user_id] uniqueidentifier,
   [ticket_number] varchar(13),
   [first_name] varchar(100) NOT NULL,
   [last_name] varchar(100) NOT NULL,
   [birth_date] date NOT NULL,
   [email] varchar(100),
   [id_doc_type] tinyint NOT NULL,
   [doc_number] varchar(50) NOT NULL,
   [doc_expiration_date] date NOT NULL
)
GO

--CREAZIONE FOREIGN KEY TABELLA passengers
ALTER TABLE [passengers] ADD CONSTRAINT [reservation_to_reservation_passengers] FOREIGN KEY ([id_reservation]) REFERENCES [reservations] ([id])
GO

ALTER TABLE [passengers] ADD CONSTRAINT [user_to_reservation_passengers] FOREIGN KEY ([user_id]) REFERENCES [users] ([user_id])
GO

ALTER TABLE [passengers] ADD CONSTRAINT [identity_document_type_to_reservation_passengers] FOREIGN KEY ([id_doc_type]) REFERENCES [identity_document_types] ([id])
GO
```

Tabella journeys

```sql
--CREAZIONE TABELLA journeys
CREATE TABLE [journeys] (
   [id_journey] bigint PRIMARY KEY IDENTITY(1, 1),
   [id_reservation] bigint NOT NULL,
   [id_flight_schedule] bigint NOT NULL,
   [flight_segment_number] tinyint NOT NULL
)
GO

--CREAZIONE FOREIGN KEYS TABELLA journeys
ALTER TABLE [journeys] ADD CONSTRAINT [reservation_to_journeys] FOREIGN KEY ([id_reservation]) REFERENCES [reservations] ([id])
GO

ALTER TABLE [journeys] ADD CONSTRAINT [flight_schedule_to_journeys] FOREIGN KEY ([id_flight_schedule]) REFERENCES [flight_schedules] ([id])
GO
```

Tabella flight_schedules

```sql
--CREAZIONE TABELLA flight_schedules
CREATE TABLE [flight_schedules] (
   [id] bigint PRIMARY KEY IDENTITY(1, 1),
   [id_flight] smallint NOT NULL,
   [id_airplane] int NOT NULL,
   [id_origin_airport] smallint NOT NULL,
   [id_destination_airport] smallint NOT NULL,
   [departure_date] date NOT NULL,
   [departure_time] time NOT NULL,
   [arrival_date] date NOT NULL,
   [arrival_time] time NOT NULL
)
GO

--CREAZIONE INDICE UNIVOCO SUI CAMPI id_flight E departure_date DELLA TABELLA flight_schedules
CREATE UNIQUE INDEX [flight_schedules_index_0] ON [flight_schedules] ("id_flight", "departure_date")
GO

--CREAZIONE FOREIGN KEYS DELLA TABELLA flight_schedules
ALTER TABLE [flight_schedules] ADD CONSTRAINT [flight_to_flight_schedules] FOREIGN KEY ([id_flight]) REFERENCES [flights] ([id])
GO

ALTER TABLE [flight_schedules] ADD CONSTRAINT [airplanes_to_flight_schedules] FOREIGN KEY ([id_airplane]) REFERENCES [airplanes] ([id])
GO

ALTER TABLE [flight_schedules] ADD CONSTRAINT [origin_airport_to_flight_schedules] FOREIGN KEY ([id_origin_airport]) REFERENCES [airports] ([id])
GO

ALTER TABLE [flight_schedules] ADD CONSTRAINT [destination_airport_to_flight_schedules] FOREIGN KEY ([id_destination_airport]) REFERENCES [airports] ([id])
GO
```

Esempi di codice Query

Come richiesto dalla traccia, vengono ora riportati alcuni esempi di query sui dati del modello di persistenza che è stato previamente popolato con alcuni dati di test tramite gli script contenuti dal file *write_model_data_population.sql* della directory *schema_persistenza_dati/code/sql*.

Query per l'estrazione dei dati generali di prenotazione

```sql
select  r.id,
        r.pnr_code,
        r.id_reservation_system,
        rSys.reservation_system_code,
        rSys.reservation_system_name,
        r.id_origin_airport,
        orig.iata_airport_code AS origin_iata_airport_code,
        orig.airport_name as origin_airport_name,
        cOrig.city_name as origin_city_name,
        r.id_destination_airport,
        dest.iata_airport_code as destination_iata_airport_code,
        dest.airport_name as detination_airport_name,
        cDest.city_name as detination_city_name,
        r.round_trip,
        case r.round_trip
            when 0 then 'One Way'
            when 1 then 'Round Trip'
        end as tipo_viaggio,
        r.departure_date,
        r.return_date,
        r.id_fare_type,
        fare.fare_code,
        fare.fare_name,
        r.total_price,
        r.discount,
        case r.total_price
            when null then null
            else r.total_price - ISNULL(r.discount,0)
        end as discounted_total_price,
        r.id_payment_method,
        pMeth.cod as payment_method_cod,
        pMeth.method_name,
        r.check_in,
        case r.check_in
            when 0 then 'Chek In is not available'
            when 1 then 'Check In is available'
        end as chek_in_availability,
        r.checked_in,
        case r.checked_in
            when 0 then 'Check In not executed'
            when 1 then 'Check In executed'
        end as check_in_status,
        r.owner_conditions_acceptance,
        r.id_reservations_status,
        rStat.cod as reservation_status_cod,
        rStat.status_name,
        rStat.status_description
from reservations r
inner join reservation_systems rSys on r.id_reservation_system = rSys.id
inner join airports orig on r.id_origin_airport = orig.id
inner join cities cOrig on orig.city_id = cOrig.id
inner join airports dest on r.id_destination_airport = dest.id
inner join cities cDest on dest.city_id = cDest.id
inner join fare_types fare on r.id_fare_type = fare.id
left join payment_methods pMeth on r.id_payment_method = pMeth.id
inner join reservations_statuses rStat on r.id_reservations_status = rStat.id;
```

Di seguito il risultato delle query, in forma testuale con la visione completa dei dati estratti, e in forma tabellare con una visione parziale, una visione tabellare totale sarebbe di difficile risoluzione con una immagine.

```
id                 pnr_code id_reservation_system reservation_system_code reservation_system_name  id_origin_airport origin_iata_airport_code origin_airport_name                          origin_city_name       id_destination_airport
destination_iata_airport_code detination_airport_name                          detination_city_name      round_trip tipo_viaggio departure_date return_date id_fare_type fare_code fare_name                total_price          discount
discounted_total_price id_payment_method payment_method_cod method_name                                    check_in chek_in_availability    checked_in check_in_status      owner_conditions_acceptance id_reservations_status reservation_status_cod
status_name     status_description
------------------ -------- --------------------- ----------------------- ------------------------ ----------------- ------------------------ -------------------------------------------- ---------------------- ----------------------
----------------------------- ------------------------------------------------ ------------------------- ---------- ------------ -------------- ----------- ------------ --------- ------------------------ -------------------- --------------------
---------------------- ----------------- ------------------ ---------------------------------------------- -------- ------------------------ ---------- -------------------- --------------------------- ---------------------- ----------------------
--------------- ----------------------------------------------------------------------------------------------------
1                  X7Y9ZQ   1                     ITA                     Prenotazioni ITA Airways 1                 FCO                      Aeroporto di Roma-Fiumicino                  Rome                   2
LIN                           Aeroporto di Milano-Linate                       Milan                     0          One Way      2025-11-24     NULL        8            BUSF      Business Flex            213,84               5,00
208,84                   1                CC                 Credit Card                                    0        Chek In is not available 0          Check In not executed 1                           3                      BHT
BOUGHT          Biglietti acquistati
2                  Y8Z0AR   1                     ITA                     Prenotazioni ITA Airways 9                 PMO                      Aeroporto di Palermo-Punta Raisi             Palermo                8
BOS                           Aeroporto di Boston Logan                        Boston                    1          Round Trip   2025-11-24     2025-12-01  4            ECOF      Economy Flex             3418,32              0,00
3418,32                  3                PP                 Paypal                                         0        Chek In is not available 0          Check In not executed 1                           3                      BHT
BOUGHT          Biglietti acquistati

(2 rows affected)


Completion time: 2025-09-20T16:55:24.6303874+02:00
```

Results | Messages

	id	pnr_code	id_reservation_system	reservation_system_code	reservation_system_name	id_origin_airport	origin_iata_airport_code	origin_airport_name	origin_city_name	id_destination_airport	destination_iata_airport_code	detination_airport_name	detination_city_name
1	1	X7Y9ZQ	1	ITA	Prenotazioni ITA Airways	1	FCO	Aeroporto di Roma-Fiumicino	Rome	2	LIN	Aeroporto di Milano-Linate	Milan
2	2	Y8Z0AR	1	ITA	Prenotazioni ITA Airways	9	PMO	Aeroporto di Palermo-Punta Raisi	Palermo	8	BOS	Aeroporto di Boston Logan	Boston

Query per l'estrazione degli itinerari delle prenotazioni

```sql
select  j.id_journey
        , j.id_reservation
        , r.pnr_code
        , j.flight_segment_number
        , j.id_flight_schedule
        , fSched.id_flight
        , a.airline_name
        , f.iata_flight_code
        , orig.iata_airport_code as origin_iata_flight_code
        , cOrig.city_name as origin_city_name
        , dest.iata_airport_code as destination_iata_flight_code
        , cDest.city_name as destnation_city_name
        , pl.model
        , fSched.departure_date
        , fSched.departure_time
        , fSched.arrival_date
        , fSched.arrival_time
from journeys j
inner join reservations r on j.id_reservation = r.id
inner join flight_schedules fSched on j.id_flight_schedule = fSched.id
inner join airports orig on fSched.id_origin_airport = orig.id
inner join cities cOrig on orig.city_id = cOrig.id
inner join airports dest on fSched.id_destination_airport = dest.id
inner join cities cDest on dest.city_id = cDest.id
inner join flights f on fSched.id_flight = f.id
inner join airlines a on f.id_airline = a.id
inner join airplanes pl on fSched.id_airplane = pl.id
order by j.id_reservation,j.flight_segment_number
```

No issues found

Results | Messages

id_journey	id_reservation	pnr_code	flight_segment_number	id_flight_schedule	id_flight	airline_name	iata_flight_code	origin_iata_flight_code	origin_city_name	destination_iata_flight_code	destnation_city_name	model	departure_date	departure_time	arrival_date	arrival_time
1	1	X7Y9ZQ	1	1	1	ITA Airways	AZ2010	FCO	Rome	LIN	Milan	AIRBUS A319	2025-11-24	07:00:00.0000000	2025-11-24	08:10:00.0000000
2	2	Y8Z0AR	1	13	21	ITA Airways	AZ1774	PMO	Palermo	FCO	Rome	AIRBUS A320	2025-11-24	07:20:00.0000000	2025-11-24	08:30:00.0000000
3	2	Y8Z0AR	2	14	22	ITA Airways	AZ614	FCO	Rome	BOS	Boston	AIRBUS A330-200	2025-11-24	10:25:00.0000000	2025-11-24	13:40:00.0000000
4	2	Y8Z0AR	3	15	23	ITA Airways	AZ615	BOS	Boston	FCO	Rome	AIRBUS A330-200	2025-12-01	17:05:00.0000000	2025-12-02	07:05:00.0000000
5	2	Y8Z0AR	4	16	24	ITA Airways	AZ1777	FCO	Rome	PMO	Palermo	AIRBUS A320NEO	2025-12-02	08:15:00.0000000	2025-12-02	09:20:00.0000000

Query per l'estrazione dei dati dei passeggeri associati alle prenotazioni

```sql
select  p.id
        , p.id_reservation
        , r.pnr_code
        , rStat.status_description as reservation_status
        , p.ticket_number
        , p.first_name
        , p.last_name
        , p.birth_date
        , p.email
        , idt.type_name as id_type
        , p.doc_number as id_number
        , p.doc_expiration_date as id_expiration_date
        , u.frequent_flyer_code
        , u.frequent_flyer_exp_date
        , u.frequent_flyer_point_balance
from passengers p
inner join reservations r on p.id_reservation = r.id
inner join reservations_statuses rStat on r.id_reservations_status = rStat.id
inner join identity_document_types idt on p.id_doc_type = idt.id
left join users u on p.user_id = u.user_id
```

No issues found

Results | Messages

id	id_reservation	pnr_code	reservation_status	ticket_number	first_name	last_name	birth_date	email	id_type	id_number	id_expiration_date	frequent_flyer_code	frequent_flyer_exp_date	frequent_flyer_point_balance
1	1	X7Y9ZQ	Biglietti acquistati	0552300662721	Fabio	Vitaterna	1968-09-13	fabio.vitaterna@studenti.unipegaso.it	ID Card	DB83707MB	2028-09-13	03770405	2025-12-31	1000
2	2	Y8Z0AR	Biglietti acquistati	0554117738322	Mario	Rossi	1990-05-10	mario.rossi@mokmail.com	Passport	YB4472738	2028-05-10	NULL	NULL	NULL
3	2	Y8Z0AR	Biglietti acquistati	0554117738330	Concetta	Montalbano	1997-11-15	cetta97@mokmail.com	Passport	YC5583849	2028-11-15	NULL	NULL	NULL
4	2	Y8Z0AR	Biglietti acquistati	0734117738349	Paolo	Rossi	2020-01-31	mario.rossi@mokmail.com	Passport	YC6683850	2026-01-31	NULL	NULL	NULL

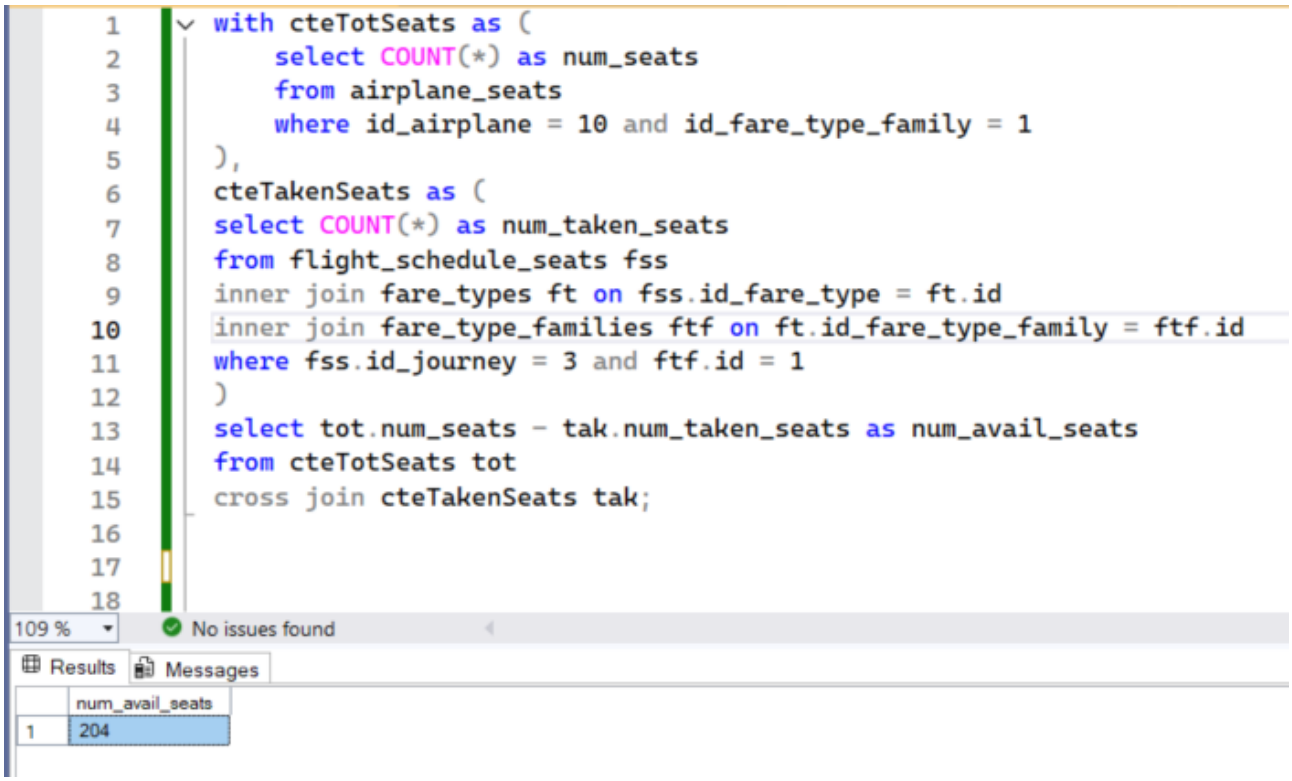
Query per l'estrazione dei dati di volo per passeggero

```sql
select  j.id_journey
        , j.id_reservation
        , r.pnr_code
        , p.first_name
        , p.last_name
        , j.flight_segment_number
        , j.id_flight_schedule
        , fSched.id_flight
        , a.airline_name
        , f.iata_flight_code
        , orig.iata_airport_code as origin_iata_flight_code
        , cOrig.city_name as origin_city_name
        , dest.iata_airport_code as destination_iata_flight_code
        , cDest.city_name as destnation_city_name
        , pl.model
        , fSched.departure_date
        , fSched.departure_time
        , fSched.arrival_date
        , fSched.arrival_time
        , airSeats.seat
        , fare.fare_code
        , fare.fare_name
from journeys j
inner join reservations r on j.id_reservation = r.id
inner join flight_schedules fSched on j.id_flight_schedule = fSched.id
inner join airports orig on fSched.id_origin_airport = orig.id
inner join cities cOrig on orig.city_id = cOrig.id
inner join airports dest on fSched.id_destination_airport = dest.id
inner join cities cDest on dest.city_id = cDest.id
inner join flights f on fSched.id_flight = f.id
inner join airlines a on f.id_airline = a.id
inner join airplanes pl on fSched.id_airplane = pl.id
left join flight_schedule_seats seats on seats.id_journey = j.id_journey
left join airplane_seats airSeats on seats.id_seat = airSeats.id
left join passengers p on seats.id_passenger = p.id
inner join fare_types fare on seats.id_fare_type = fare.id
order by j.id_reservation,j.flight_segment_number
```

Results Messages

	id_journey	id_reservation	pnr_code	first_name	last_name	flight_segment_number	id_flight_schedule	id_flight	airline_name	iata_flight_code	origin_iata_flight_code	origin_city_name	destination_iata_flight_code	destnation_city_name	model	departure_date	departure_time	arrival_date	arrival_time	seat	fare_code	fare_name
1	1	1	X7Y9ZQ	Fabio	Vitaterna	1	1	1	ITA Airways	AZ2010	FCO	Rome	LIN	Milan	AIRBUS A319	2025-11-24	07:00:00.0000000	2025-11-24	08:10:00.0000000	2A	BUSF	Business Flex
2	2	2	Y8Z0AR	Mario	Rossi	1	13	21	ITA Airways	AZ1774	PMO	Palermo	FCO	Rome	AIRBUS A320	2025-11-24	07:20:00.0000000	2025-11-24	08:30:00.0000000	10B	ECOF	Economy Flex
3	2	2	Y8Z0AR	Concetta	Montalbano	1	13	21	ITA Airways	AZ1774	PMO	Palermo	FCO	Rome	AIRBUS A320	2025-11-24	07:20:00.0000000	2025-11-24	08:30:00.0000000	10C	ECOF	Economy Flex
4	2	2	Y8Z0AR	Paolo	Rossi	1	13	21	ITA Airways	AZ1774	PMO	Palermo	FCO	Rome	AIRBUS A320	2025-11-24	07:20:00.0000000	2025-11-24	08:30:00.0000000	10A	ECOF	Economy Flex
5	3	2	Y8Z0AR	Mario	Rossi	2	14	22	ITA Airways	AZ614	FCO	Rome	BOS	Boston	AIRBUS A330-200	2025-11-24	10:25:00.0000000	2025-11-24	13:40:00.0000000	29D	ECOF	Economy Flex
6	3	2	Y8Z0AR	Concetta	Montalbano	2	14	22	ITA Airways	AZ614	FCO	Rome	BOS	Boston	AIRBUS A330-200	2025-11-24	10:25:00.0000000	2025-11-24	13:40:00.0000000	29G	ECOF	Economy Flex
7	3	2	Y8Z0AR	Paolo	Rossi	2	14	22	ITA Airways	AZ614	FCO	Rome	BOS	Boston	AIRBUS A330-200	2025-11-24	10:25:00.0000000	2025-11-24	13:40:00.0000000	29E	ECOF	Economy Flex
8	4	2	Y8Z0AR	Mario	Rossi	3	15	23	ITA Airways	AZ615	BOS	Boston	FCO	Rome	AIRBUS A330-200	2025-12-01	17:05:00.0000000	2025-12-02	07:05:00.0000000	29D	ECOF	Economy Flex
9	4	2	Y8Z0AR	Concetta	Montalbano	3	15	23	ITA Airways	AZ615	BOS	Boston	FCO	Rome	AIRBUS A330-200	2025-12-01	17:05:00.0000000	2025-12-02	07:05:00.0000000	29G	ECOF	Economy Flex
10	4	2	Y8Z0AR	Paolo	Rossi	3	15	23	ITA Airways	AZ615	BOS	Boston	FCO	Rome	AIRBUS A330-200	2025-12-01	17:05:00.0000000	2025-12-02	07:05:00.0000000	29E	ECOF	Economy Flex
11	5	2	Y8Z0AR	Mario	Rossi	4	16	24	ITA Airways	AZ1777	FCO	Rome	PMO	Palermo	AIRBUS A320NEO	2025-12-02	08:15:00.0000000	2025-12-02	09:20:00.0000000	10B	ECOF	Economy Flex
12	5	2	Y8Z0AR	Concetta	Montalbano	4	16	24	ITA Airways	AZ1777	FCO	Rome	PMO	Palermo	AIRBUS A320NEO	2025-12-02	08:15:00.0000000	2025-12-02	09:20:00.0000000	10C	ECOF	Economy Flex
13	5	2	Y8Z0AR	Paolo	Rossi	4	16	24	ITA Airways	AZ1777	FCO	Rome	PMO	Palermo	AIRBUS A320NEO	2025-12-02	08:15:00.0000000	2025-12-02	09:20:00.0000000	10A	ECOF	Economy Flex

Query per l'estrazione del numero di posti disponibili in classe Economy per un dato volo



```
with cteTotSeats as (
    select COUNT(*) as num_seats
    from airplane_seats
    where id_airplane = 10 and id_fare_type_family = 1
),
cteTakenSeats as (
select COUNT(*) as num_taken_seats
from flight_schedule_seats fss
inner join fare_types ft on fss.id_fare_type = ft.id
inner join fare_type_families ftf on ft.id_fare_type_family = ftf.id
where fss.id_journey = 3 and ftf.id = 1
)
select tot.num_seats - tak.num_taken_seats as num_avail_seats
from cteTotSeats tot
cross join cteTakenSeats tak;
```

109 % No issues found

Results Messages

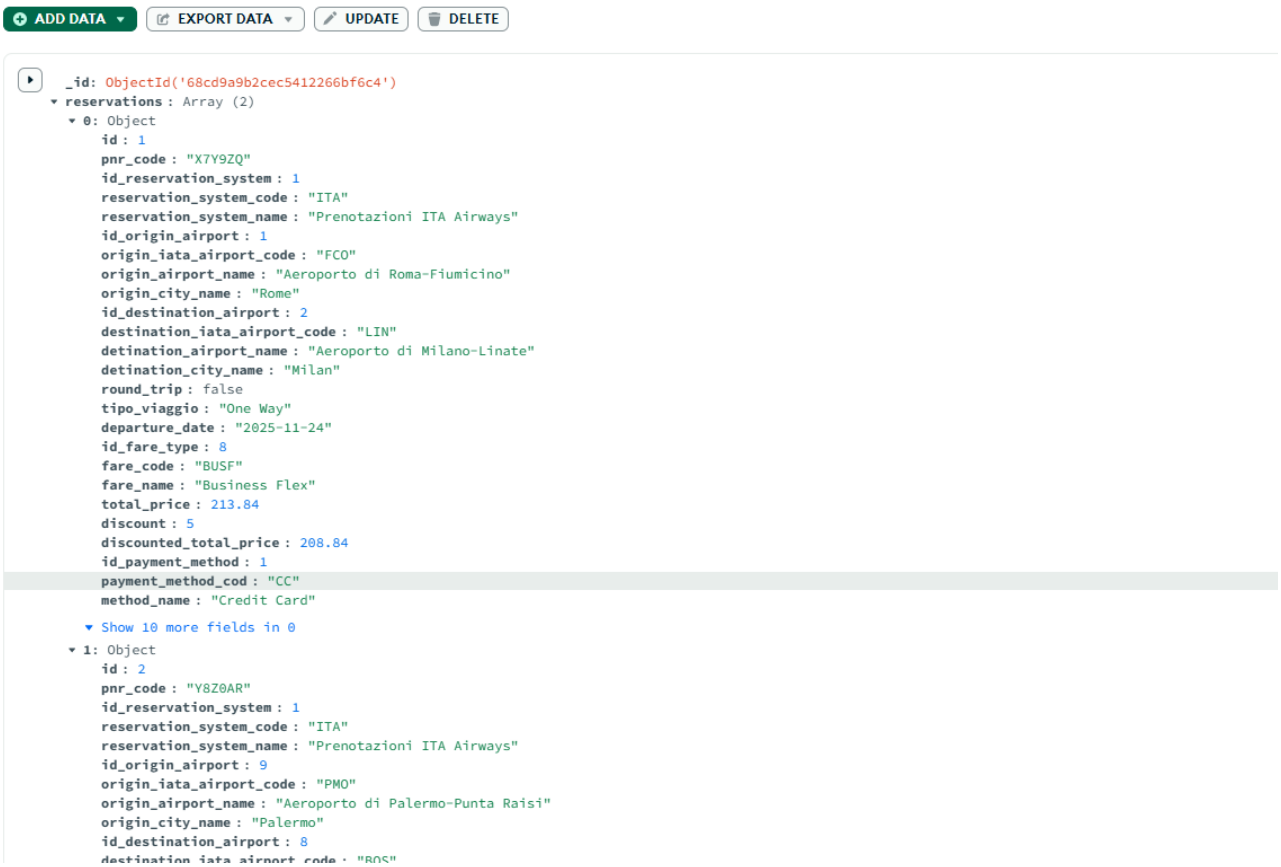
	num_avail_seats
1	204

Implementazione del Read Model

Come esposto precedentemente i dati del read model saranno contenuti collection MongoDB sotto forma di documenti JSON. A titolo di esempio viene riportato in figura parte di un possibile documento JSON contenenti i dati delle prenotazioni; il JSON può essere consultato nel suo insieme nel file *full_reservations.json* contenuto nella directory *schema_persistenza_dati/code/json* del GitHub di progetto.

```
{
    "reservations": [{
            "id": 1,
            "pnr_code": "X7Y9ZQ",
            "id_reservation_system": 1,
            "reservation_system_code": "ITA",
            "reservation_system_name": "Prenotazioni ITA Airways",
            "id_origin_airport": 1,
            "origin_iata_airport_code": "FCO",
            "origin_airport_name": "Aeroporto di Roma-Fiumicino",
            "origin_city_name": "Rome",
            "id_destination_airport": 2,
            "destination_iata_airport_code": "LIN",
            "detination_airport_name": "Aeroporto di Milano-Linate",
            "detination_city_name": "Milan",
            "round_trip": false,
            "tipo_viaggio": "One Way",
            "departure_date": "2025-11-24",
            "id_fare_type": 8,
            "fare_code": "BUSF",
            "fare_name": "Business Flex",
            "total_price": 213.8400,
            "discount": 5.0000,
            "discounted_total_price": 208.8400,
            "id_payment_method": 1,
            "payment_method_cod": "CC",
            "method_name": "Credit Card",
            "check_in": false,
            "chek_in_availability": "Chek In is not available",
            "checked_in": false,
            "check_in_status": "Check In not executed",
            "owner_conditions_acceptance": true,
            "id_reservations_status": 3,
            "reservation_status_cod": "BHT",
            "status_name": "BOUGHT",
            "status_description": "Biglietti acquistati",
            "passengers": [{
                    "id": 1,
                    "id_reservation": 1,
                    "pnr_code": "X7Y9ZQ",
                    "reservation_status": "Biglietti acquistati",
                    "ticket_number": "0552300662721",
                    "first_name": "Fabio",
                    "last_name": "Vitaterna",
                    "birth_date": "1968-09-13",
                    "email": "fabio.vitaterna@studenti.unipegaso.it",
                    "id_type": "ID Card",
                    "id_number": "DB83707MB",
                    "id_expiration_date": "2028-09-13",
                    "frequent_flyer_code": "03770405",
                    "frequent_flyer_exp_date": "2025-12-31",
                    "frequent_flyer_point_balance": 1000,
                    "journeys": [{
                            "id_journey": 1,
                            "id_reservation": 1,
                            "pnr_code": "X7Y9ZQ",
                            "first_name": "Fabio",
```

Di seguito un'immagine parziale dello stesso documento nel client MongoDB usato per l'implementazione.

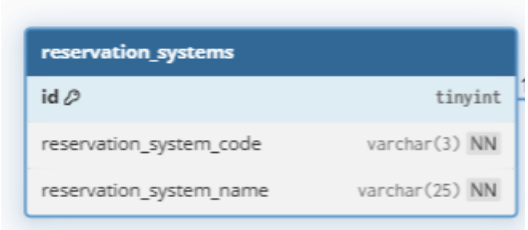


Come di può notare, l'uso di documenti MongoDB aumenta la fruibilità del dato aggregato grazie ad una migliore capacità di rappresentare strutture gerarchiche rispetto a query SQL.

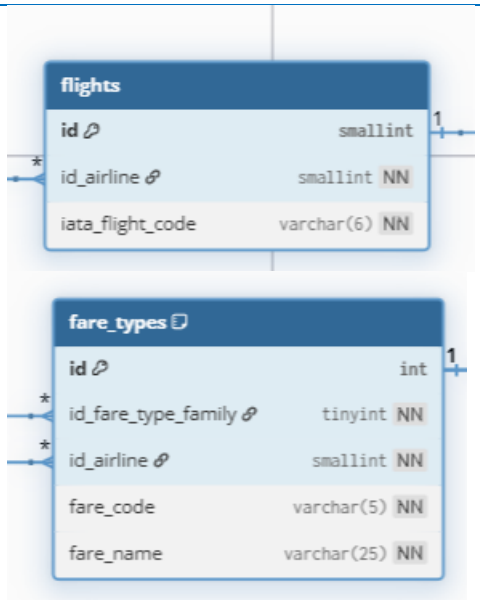
Campi di applicazione
(Descrivere gli ambiti di applicazione dell'elaborato progettuale e i vantaggi derivanti della sua applicazione):

Dal modello progettato e dalla relativa architettura applicativa di riferimento possono essere il punto di partenza per campi di applicazione che, pur spingendosi oltre il perimetro del presente elaborato, vale la pena accennare.

- Il punto di partenza del modello, oltre all'analisi generale del dominio del trasporto aereo civile, si è concretamente basato sullo studio del portale di ITA Airways. Nonostante ciò, il modello è predisposto, al netto di necessarie implementazioni evolutive, a supportare l'implementazione di sistemi di prenotazione diversi da quelli di ITA Airways e anche di andare oltre il dominio del trasporto aereo. A titolo di esempio molto superficiale, si faccia riferimento alle seguenti tabelle che sono già modellate per un utilizzo più generale:
    - La tabella reservation_systems predisposta per introdurre qualsiasi sistema di prenotazione diverso da quello discusso nell'elaborato.



    - Le tabelle **flights** e **fare_types**, tramite il campo *id_airline*, chiave esterna che implementa la relazione 1 a molti dalla tabella **airlines** verso le suddette tabelle permette l'inserimento di dati relativi a compagnie aeree diverse da ITA Airways.



- L'architettura CQRS progettata permette implementazioni a supporto di strategie commerciali avanzate e personalizzazioni dell'offerta.
- L'architettura CQRS progettata può essere di supporto per implementazioni di algoritmi di machine learning. Un esempio può essere l'applicazione di strategie predittive al dominio della manutenzione delle flotte.
- Attraverso l'implementazione di read model ad hoc, è possibile sviluppare sistemi di dashboard orientati alla business intelligence, come ad esempio metriche di KPI per monitorare gli andamenti della produttività aziendale.
- Come detto precedentemente, CQRS apre la porta allo sviluppo tramite microservizi. Uno delle possibili evoluzioni di un modello di questo tipo è l'esposizione di uno o più set di API per permettere a sistemi di prenotazione terzi di usufruire delle strutture dati implementate.
- L'architettura progettata è aperta allo sviluppo di un sistema completamente cloud based. Ad esempio, prendendo come riferimento Azure (ovviamente analoghe considerazioni possono essere fatte per Amazon AWS e Google Cloud Platform) sarebbe possibile, mantenendo l'architettura e i modelli dati progettati, utilizzare:
  - **Azure SQL Database** come DBMS per il write model.
  - **Azure Cosmos DB** come DBMS per il read model.
  - **Azure Service Bus** come bus
  - **Azure Functions** per l'implementazione dei microservizi responsabili della logica di business.
  - **Azure App Service**, in cooperazione di un framework come ad esempio **Angular** per il front end.

Valutazione dei risultati

(Descrivere le potenzialità e i limiti ai quali i risultati dell'elaborato sono potenzialmente esposti):

Potenzialità del modello

- Grazie alla segregazione imposta dal pattern CQRS, il modello progettato ottimizza la gestione delle prenotazioni.
- Il write model, fortemente normalizzato, garantisce l'integrità e la consistenza dei dati, anche per funzionalità complesse come le prenotazioni di itinerari multi-tratta e di gestione della tariffazione dinamica.
- Il write model progettato garantisce la persistenza transazionale ACID dei dati.
- Le proiezioni denormalizzate che popolano il read model NoSql facilitano la fruizione ad alta performance dei dati sia verso l'esterno (passeggeri) che verso l'interno (customer care, gestori di processo), soddisfacendo quindi i requisiti di ottimizzazione delle performance.
- Come già esposto precedentemente, CQRS permette che il sistema soddisfi i requisiti di scalabilità.
- Il modello soddisfa gli standard di dominio.
- L'architettura progettata è compatibile con lo sviluppo di integrazioni verso terzi, ad esempio altre compagnie aeree.

Criticità, limiti e possibili miglioramenti.

- Il modello di persistenza può essere migliorato. Di seguito alcuni esempi di modellazioni che hanno esulato dal presente project work, ma che dovrebbero essere modellati:
  - Supporto per l'interfaccia grafica dedicata alla scelta dei posti sulle aeromobili, in particolare per la creazione di mappe dinamica per ogni modello e per ogni personalizzazione degli allestimenti implementata da ITA Airways e, in generale, da ogni compagnia.
  - Migliorabile la modellazione dell'interfaccia verso i pagamenti on line.
  - Supporto per la gestione delle sessioni utente.

    - Supporto per i servizi crittografici dei dati sensibili: nel modello è solamente previsto, nella presente versione, che le lunghezze dei campi che si riferiscono a tali dati supportino una possibile persistenza di dati crittografati.
- L'esperienza lavorativa e la partecipazione a numerosi progetti in cui è stato applicato CQRS suggeriscono alcune possibili criticità legate all'architettura:
    - Lo sviluppo reale del modello e dell'architettura proposta possono soffrire di forte complessità implementativa, in modo particolare per problematiche come:
        - Il coordinamento tramite bus tra write e read model.
        - L'error handling distribuito.
        - L'implementazione del testing, sia a livello di unit che end-to-end test.
    - Data la complessità architetturale, il debito tecnico di un eventuale team di sviluppo deve essere il più possibile minimizzato.
    - Anche se in generale il sistema è progettato per alte prestazioni, è possibile il sorgere di colli di bottiglia legati a alti volumi di processamento di eventi. Un eventuale sviluppo reale deve tener conto di questa eventualità.