|  |  |
| --- | --- |
| **Immagine che contiene Carattere, testo, bianco, logo  Il contenuto generato dall'IA potrebbe non essere corretto.** | |
|  |  |
|  |  |
|  |  |
|  | Fabio Vitaterna |
|  | 0312300511 |
| Immagine che contiene testo, Carattere, bianco, tipografia  Il contenuto generato dall'IA potrebbe non essere corretto. | 1 |
| Immagine che contiene testo, Carattere, bianco, tipografia  Il contenuto generato dall'IA potrebbe non essere corretto. | La digitalizzazione dell’impresa |
| Immagine che contiene testo, Carattere, bianco, tipografia  Il contenuto generato dall'IA potrebbe non essere corretto. | 4 |
| Immagine che contiene testo, Carattere, bianco, tipografia  Il contenuto generato dall'IA potrebbe non essere corretto. | Progettazione dello schema di persistenza dei dati a supporto dei servizi di un’azienda nel settore dei trasporti |
| **Immagine che contiene testo, Carattere, bianco  Il contenuto generato dall'IA potrebbe non essere corretto.** | Progettazione dello schema di persistenza a supporto del servizio di prenotazione di ITA Airways |
|  | |
|  | |
| 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[[1]](#footnote-1).     - 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. | |
|  | |
| **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 necesarie 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[[2]](#footnote-2) cui il modello di persistenza faccia riferimento. 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. | |
|  | |
|  | |
|  | |
|  | |
| 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 traffico aereo e di scalabilità sia del modello stesso sia della fruizione dei data service da parte dell’intero sistema applicativo.  Di seguito sono dettagliate le modalità di lavoro che hanno permesso il raggiungimento dell’obiettivo.  **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. | |
|  | |
| **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 |   **Scelte architetturali**  Il settore del traffico 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. | |
|  | |
| Inserisci qui il testo | |
|  | |
| 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. A screenshot of a computer      AI-generated content may be incorrect.   + 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. A screenshot of a computer      AI-generated content may be incorrect. A screenshot of a computer      AI-generated content may be incorrect. * 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[[3]](#footnote-3) 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** per il message management.   + **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. | |
|  | |
| **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. | |

1. Nelle citate esperienze lavorative sono stati i più frequentemente utilizzati. [↑](#footnote-ref-1)
2. Per Minimun Viable Product 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*. [↑](#footnote-ref-2)
3. Analoghe considerazioni possono essere fatte per Amazon AWS e Google Cloud Platform. [↑](#footnote-ref-3)