Documentazione Architetturale

---

1. Introduzione

La presente documentazione descrive in modo completo l'architettura di un sistema e-commerce progettato per operare in un contesto ad alta variabilità di carico, con requisiti stringenti in termini di scalabilità, sicurezza, affidabilità e manutenibilità.

L'obiettivo del documento è fornire una descrizione architetturale strutturata, conforme ai principi dell'IEEE 1016 – Standard for Architectural Description, illustrando:

- il problema da affrontare,
- gli stakeholder coinvolti,
- i requisiti funzionali e non funzionali,
- i driver architetturali,
- le architetture candidate analizzate,
- la soluzione selezionata,
- i trade-off architetturali,
- i rischi identificati,
- la valutazione finale della soluzione proposta.

La documentazione integra descrizioni testuali e diagrammi UML per fornire una visione sia logica sia strutturale del sistema.

---

2. Contesto e Problema da Affrontare

L'azienda richiede la realizzazione di una piattaforma e-commerce moderna in grado di sostenere:

- elevati picchi di traffico durante promozioni o eventi stagionali,
- un numero elevato di utenti concorrenti,
- transazioni sicure e affidabili,
- integrazione con sistemi esterni (pagamenti e spedizioni),
- evoluzione continua delle funzionalità.

Il problema architetturale principale consiste nel bilanciare:

- performance elevate,
- disponibilità continua del servizio,
- sicurezza dei dati sensibili,
- capacità di crescita futura,
- controllo dei costi infrastrutturali.

Il sistema deve inoltre essere distribuito su infrastruttura cloud (AWS o Azure) e rispettare normative come GDPR e PCI-DSS.

---

3. Stakeholder e Obiettivi

Gli stakeholder principali sono:

End Users - Esperienza fluida - Risposte rapide - Sicurezza dei dati

Business Owners - Scalabilità - Riduzione dei costi operativi - Evoluzione modulare

Administrators - Monitoraggio - Logging centralizzato - Gestione dei servizi

Payment Providers - Integrazione sicura - Conformità normativa

Shipping Providers - Interfacce affidabili - Aggiornamenti in tempo reale

---

4. Requisiti del Sistema

4.1 Requisiti Funzionali

Il sistema deve consentire:

- Gestione del catalogo prodotti
- Ricerca e navigazione prodotti
- Gestione carrello
- Creazione e gestione ordini
- Integrazione con gateway di pagamento
- Integrazione con servizi di spedizione
- Gestione account utente
- Supporto multilingua

4.2 Requisiti Non Funzionali

Performance Tempo di risposta inferiore a 2 secondi per operazioni critiche.

Scalabilità Supporto ad almeno 50.000 utenti concorrenti.

Disponibilità Uptime minimo del 99,9%.

Affidabilità Nessuna perdita di ordini in caso di failure temporanei.

Sicurezza Cifratura dati in transito e a riposo. Conformità GDPR e PCI-DSS.

Manutenibilità Possibilità di introdurre nuove funzionalità con impatto minimo.

---

5. Driver Architetturali

I driver architetturali derivano direttamente dai requisiti non funzionali più critici.

FD-01 – Performance

Influenza caching, distribuzione dei carichi e progettazione database.

FD-02 – Scalabilità

Richiede architettura distribuibile e scalabile orizzontalmente.

FD-03 – Disponibilità

Impone ridondanza e fault tolerance.

FD-04 – Affidabilità

Richiede meccanismi di persistenza robusti e gestione transazioni.

FD-05 – Deployment Cloud

Influenza la scelta di servizi containerizzati e orchestrazione.

FD-06 – Manutenibilità

Favorisce modularità e basso accoppiamento.

---

6. Analisi delle Architetture Candidate

6.1 Space-Based Architecture

Vantaggi: - Alta scalabilità - Performance elevate

Svantaggi: - Complessità di gestione della consistenza - Costi di implementazione

---

6.2 Client-Server Architecture

Vantaggi: - Semplicità - Facilità di comprensione

Svantaggi: - Collo di bottiglia sul server - Limitata scalabilità orizzontale

---

6.3 Microservices Architecture

Vantaggi: - Modularità - Scalabilità indipendente dei servizi - Resilienza - Flessibilità evolutiva

Svantaggi: - Maggiore complessità operativa - Necessità di gestione comunicazioni distribuite - Problematiche di consistenza eventuale

---

7. Selezione della Soluzione

L'architettura Microservices è stata selezionata in quanto:

- supporta la scalabilità richiesta,
- consente isolamento dei failure,
- permette deploy indipendenti,
- favorisce evoluzione modulare del sistema.

---

8. Trade-Off Architetturali

Performance vs Complessità

Maggiore scalabilità implica maggiore complessità gestionale.

Disponibilità vs Costi

Ridondanza aumenta costi infrastrutturali.

Sicurezza vs Performance

Crittografia introduce overhead computazionale.

SEZIONE DIAGRAMMI

---

9. Diagramma del Contesto

Il diagramma del contesto rappresenta il sistema come entità unica e mostra le interazioni con attori esterni. Evidenzia i confini del sistema e le integrazioni con provider esterni.

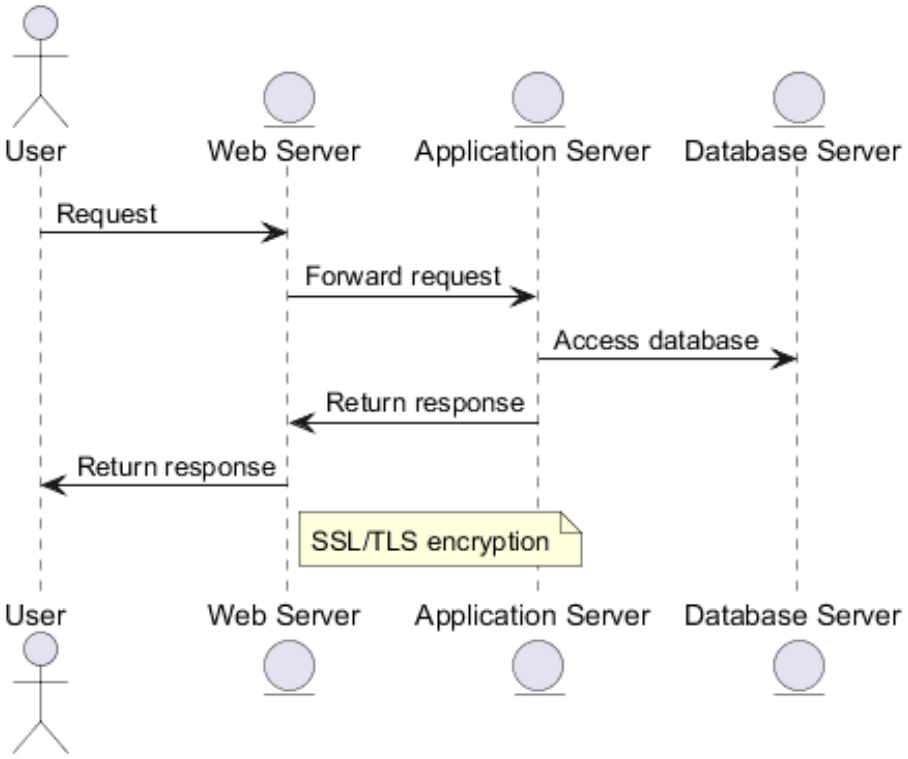

Figure 1: Diagramma del Contesto

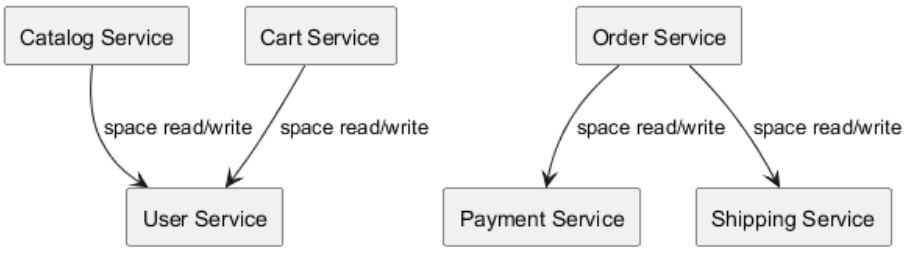

Figure 2: Diagramma dei Componenti

10. Diagramma dei Componenti

Il sistema è suddiviso in microservizi indipendenti:

- Catalog Service
- Cart Service
- Order Service
- User Service
- Payment Service
- Shipping Service

Ogni servizio espone API REST e comunica con gli altri tramite protocolli standard.

---

11. Diagramma del Deployment

Figure 3: Diagramma del Deployment

Il diagramma mostra la distribuzione dei servizi su nodi cloud:

- Web Server
- Application Server
- Database Server
- Message Queue Server

La separazione dei livelli garantisce scalabilità e fault isolation.

---

12. Diagramma di Sequenza

Descrive il flusso temporale delle operazioni durante il checkout:

1. Aggiunta al carrello
2. Creazione ordine
3. Pagamento
4. Generazione spedizione

---

13. Diagramma di Sicurezza

Evidenzia trust boundaries e contromisure:

- TLS

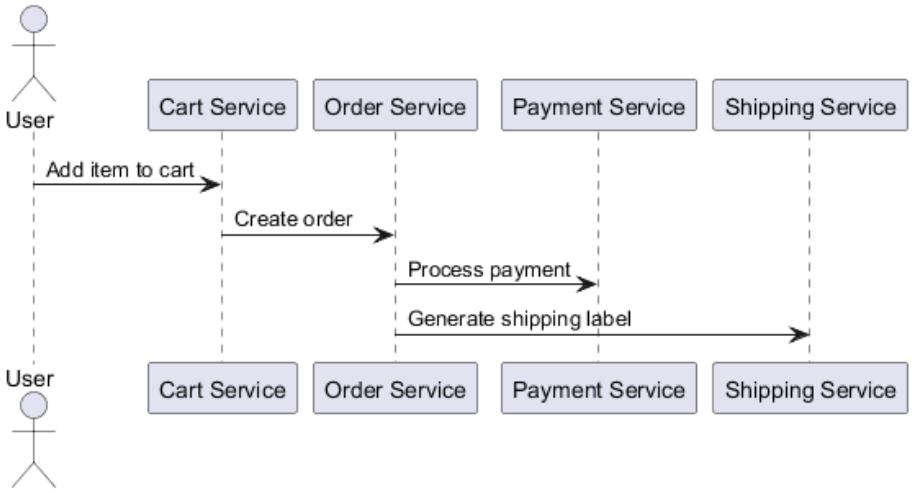

Figure 4: Diagramma di Sequenza

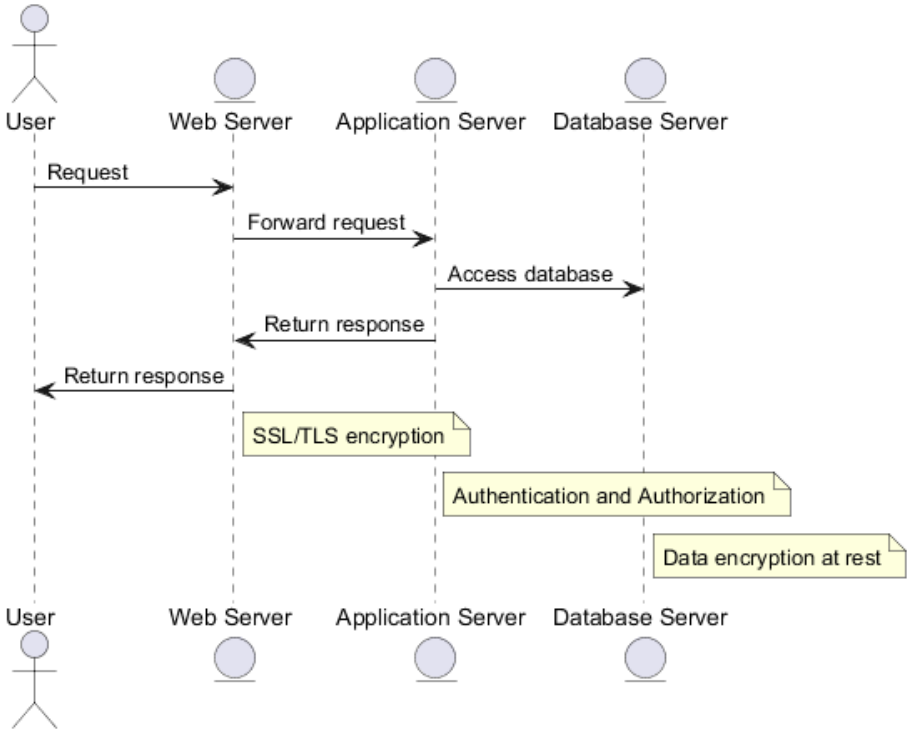

Figure 5: Diagramma di Sicurezza

- Autenticazione forte
- Controllo accessi
- Audit logging

---

14. Valutazione Architetturale

L'architettura soddisfa pienamente:

- FD-01 Performance
- FD-02 Scalabilità
- FD-05 Deployment Cloud
- FD-06 Manutenibilità

Richiede ulteriori raffinamenti per:

- FD-03 Disponibilità 99,9%
- Gestione esplicita consistenza distribuita

---

15. Rischi Identificati

- Complessità operativa
- Latenza di rete
- Consistenza eventuale
- Vendor lock-in cloud

---

16. Raccomandazioni

- Circuit breaker pattern
- Retry automatici
- Monitoring centralizzato
- Logging distribuito
- Auto-scaling
- Strategie di consistenza esplicite

---

17. Conclusioni

L'architettura microservizi rappresenta la soluzione più adeguata per il contesto descritto.

Essa consente di bilanciare scalabilità, sicurezza e manutenibilità, garantendo al tempo stesso resilienza e capacità evolutiva nel lungo periodo.

La documentazione fornisce una base strutturata per l'implementazione e l'evoluzione futura del sistema.