Università degli Studi di Salerno
Corso di Ingegneria del Software

AniTour
System Design
Versione 1.0

Data: 23/11/2025

Progetto: AniTour	Versione: 1.0
Documento: System Design	Data: 23/11/2025

Indice

1. Introduction

1.1 Purpose of the system

Lo scopo del sistema AniTour è fornire una piattaforma di e-commerce specializzata nella vendita di tour tematici dedicati alla cultura nerd (anime, videogiochi, cultura pop giapponese). Il sistema permette agli utenti (Guest e Clienti) di scoprire e acquistare pacchetti viaggio, e agli Organizzatori di creare e gestire le proprie offerte turistiche. AniTour funge da intermediario centralizzato per semplificare la prenotazione, garantendo qualità e sicurezza nelle transazioni.

1.2 Design goals

Gli obiettivi del design derivano dai requisiti non funzionali identificati nel RAD. Sono stati prioritizzati come segue:

1. **Usability:** Il sistema deve offrire un'interfaccia intuitiva e responsive, accessibile da dispositivi desktop e mobile, per facilitare la navigazione e l'acquisto.
2. **Reliability:** Il sistema deve garantire un uptime annuo >= 99.5% e gestire il failover dei servizi critici.
3. **Performance:** I tempo di risposta devono essere inferiori a 2 secondi e il sistema deve supportare almeno 500 utenti simultanei senza degrado.
4. **Security:** Protezione dei dati personali (GDPR) e transazioni finanziarie sicure/certificate.
5. **Supportability:** L'architettura deve essere modulare (disaccoppiata) per permettere l'aggiunta di nuove funzionalità senza impattare sui componenti esistenti.

1.3 Definitions, acronyms and abbreviations

- **ECB:** Entity-Control-Boundary (Pattern architetturale utilizzato).
- **DAO:** Data Access Object (Pattern per l'accesso ai dati).
- **DBMS:** Database Management System.
- **API:** Application Programming Interface.
- **Voucher:** Documento PDF generato post-acquisto.

1.4 References

- Requirement Analysis Document – AniTour v1.2
- Problem Statement – AniTour v1.1

1.5 Overview

Questo documento descrive l'architettura software di AniTour. Il sistema è decomposto in sottosistemi logici basati sul pattern ECB, mappati su un'architettura fisica client-server a 3 livelli (3-tier). Vengono dettagliate le strategie per la gestione dei dati persistenti, il controllo degli accessi e la gestione del flusso globale.

2. Current software architecture

Attualmente non esiste un sistema software integrato (legacy system) che AniTour va a sostituire. L'offerta di tour nerd è frammentaria, gestita manualmente tramite forum,

social network o agenzie generaliste non coordinate. Il processo attuale è manuale, decentralizzato e privo di standard di verifica di qualità. AniTour viene sviluppato come soluzione *greenfield* per centralizzare e automatizzare questi processi.

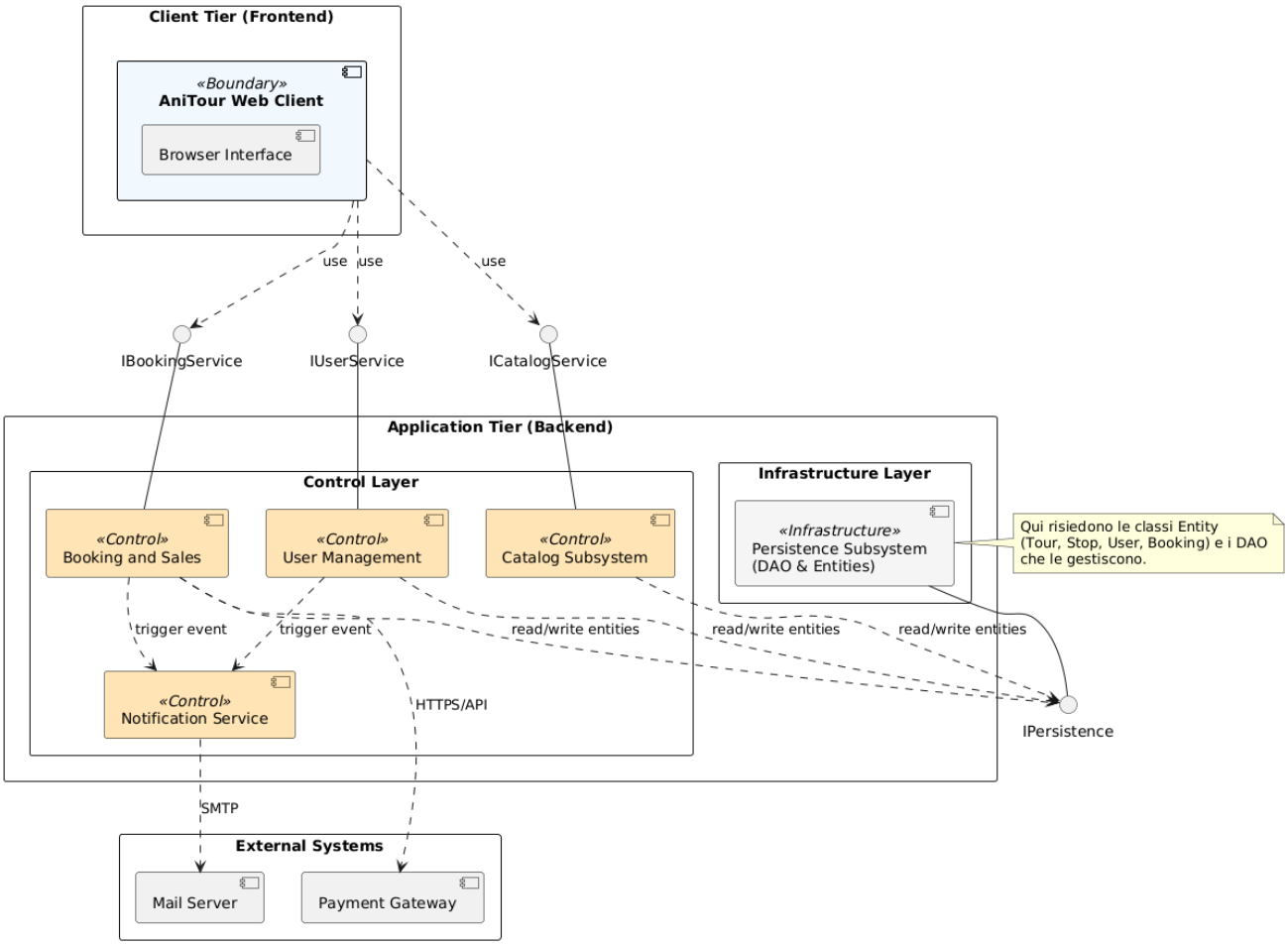
3. Proposed software architecture

3.1. Overview

L'architettura proposta è basata sullo stile Client-Server a 3 livelli (3-Tier). Logicamente, l'applicazione segue il pattern Entity-Control-Boundary (ECB) per garantire una chiara separazione delle responsabilità:

- **Boundary:** Gestisce l'interazione con l'utente.
- **Control:** Gestisce la logica applicativa e i casi d'uso.
- **Entity:** Rappresenta i dati di dominio.

3.2. Subsystem decomposition

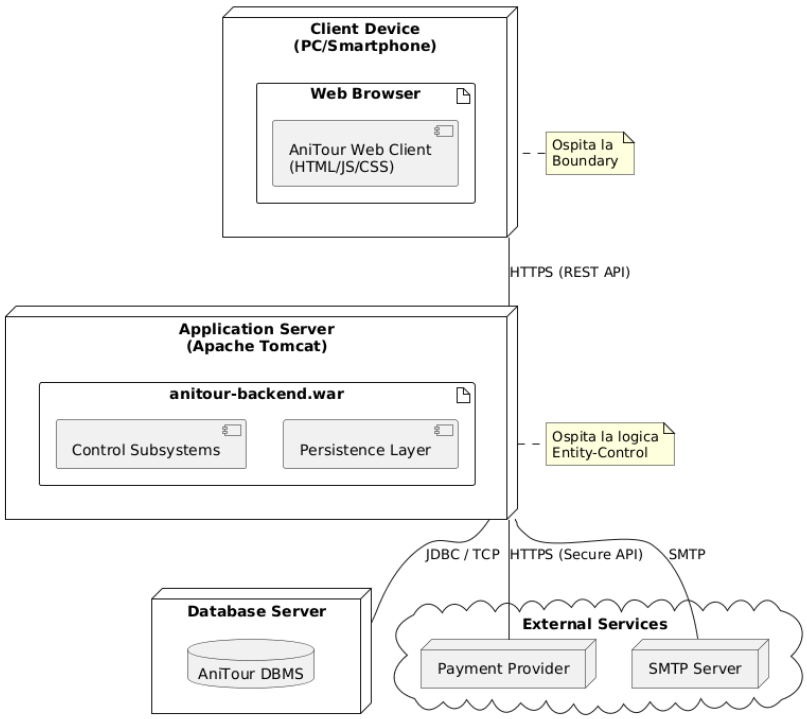


1. **AniTour Web Client (Boundary):**
   - Responsabile della presentazione delle informazioni all'utente e della cattura degli input.
   - Contiene le classi Boundary (es. TourPage, CartPage, LoginForm)

- Comunica esclusivamente con il livello *Control*

2. **User Management Subsystem (Control):**
   - Gestisce la logica relativa agli utenti: autenticazione, registrazione, gestione profili.
   - Classi chiave; AuthManager, UserManager.
3. **Catalog Subsystem (Control):**
   - Gestisce la ricerca, il filtraggio e la visualizzazione dei dettagli dei tour.
   - Permette agli organizzatori di creare/modificare tour e agli operatori di approvarli.
   - Classi chiave: CatalogManager, TourManager.
4. **Booking and Sales Subsystem (Control):**
   - Gestisce il carrello (sessione), il processo di checkout e la creazione degli ordini.
   - Classi chiave: CartManager, OrderManager.
5. **Payment Subsystem (External Interface):**
   - Interfaccia verso gateway di pagamento esterni
   - Gestisce la transazione finanziaria sicura
6. **Notification Subsystem (Service):**
   - Gestisce l'invio di e-mail transazionali (conferma registrazione, invio voucher, notifiche stato tour)
7. **Persistence Subsystem (Infrastructure/Data):**
   - Contiene i *DAO (Data Access Object)* e gestisce la comunicazione con il DBMS.
   - Le classi *Control* utilizzano le interfacce fornite da questo subsystem per salvare o recuperare le *Entity.*

3.3. Hardware/software mapping



L’architettura fisica prevede una distribuzione su nodi distinti per garantire scalabilità e sicurezza.

- **Client Node:** Browser web (Chrome, Firefox, Edge) su PC o mobile. Esegue il codice front-end (HTML/CSS/JS)..
- **Application Server Node:** Server che ospita la logica di business (Java/Spring). Gestisce i sottosistemi di Controllo e le Entity.Il *cliente* deve essere in grado di navigare il sito, cercare tour e finalizzarne l’acquisto.
- **Database Server Node:** Server dedicato per il DBMS Relazionale (es. MySQL/PostgreSQL).
- **Mail Server (Esterno):** Servizio SMTP per l’invio di notifiche.

3.4. Persistent data management

La persistenza è gestita tramite un **Database Relazionale**, scelto per garantire integrità referenziale e consistenza nelle transazioni finanziarie (ACID).

- **Mapping:** Le classi *Entity* (User, Tour, Booking, ecc. descritte nel RAD) vengono mappate su tabelle relazionali. Il sito deve essere pienamente operativo su browser moderni (Chrome, Firefox, Edge, ecc…).
- **Ruolo dei DAO:** I metodi di accesso ai dati (CRUD) risiedono nelle classi DAO all'interno del *Persistence Subsystem*. Le classi *Control* invocano i DAO per rendere persistenti le *Entity*. Non vi è logica SQL all'interno delle Entity o delle Boundary.
- **Sessione:** Il carrello (Cart) è mantenuto volatile nella sessione HTTP o in uno store temporaneo (es. Redis) finché non viene convertito in ordine (Booking).

3.5. Access control and security

L’accesso è controllato tramite una *Access Control List (ACL)* basata sui ruoli definiti nel RAD:

Attore	Diritti di Accesso
Guest	Visualizzazione catalogo, Ricerca, Registrazione, Login.
Cliente	Tutto ciò che fa il Guest + Aggiunta al carrello, Checkout, Visualizzazione storico ordini.
Organizzatore	Gestione profilo personale + Creazione Tour, Modifica propri Tour (Draft), Invio per approvazione.
Operatore	Approvazione/Rifiuto Tour, Gestione Utenti, Ban.

- **Autenticazione:** Username e password. Le password sono salvate tramite hashing sicuro.
- **Sicurezza:** Tutte le comunicazioni avvengono su protocollo HTTPS crittografato. I dati delle carte di credito non vengono salvati direttamente nel DB ma gestiti tramite token del Payment Gateway.

3.6. Global software control

Il flusso di controllo è *Event-Driven*, tipico delle web-app moderne.

1. L'utente interagisce con la *Boundary* (click su un pulsante).
2. La Boundary invia una richiesta HTTP al server.
3. Il *Control* appropriato intercetta la richiesta, elabora la logica di business, manipola

le *Entity* e usa i *DAO* per la persistenza.
4. Il *Control* restituisce i dati alla *Boundary* per la visualizzazione della risposta.

Eccezione: Il sottosistema di notifica opera in modo asincrono per non bloccare l'interfaccia utente durante l'invio delle e-mail.

3.7. Boundary conditions

- **Inizializzazione:** All'avvio del server (StartServer), il sistema verifica la connessione al db e carica le configurazioni globali.
- **Terminazione:** Lo ShutDownServer chiude le connessioni al db e completa le transazioni pendenti prima di spegnersi.
- **Failure:**
  - *Db Down:* Il sistema mostra una pagina "Manutenzione in corso" e logga l'errore critico.
  - *Pagamento fallito:* Il sistema gestisce l'eccezione restituita dal Payment gateway, non crea l'ordine e informa l'utente dell'errore permettendo il retry.

4. Subsystem services

Questa sezione descrive le interfacce (API) fornite dai principali sottosistemi.

4.1. User Management Service

- login(username, password): Restituisce token di sessione o errore.
- register(userData): Crea un nuovo Guest/Organizzatore.
- validateOrganizer(vatNumber): Verifica P.IVA organizzatore.

4.2. Catalog Service

- searchTours(keyword, filters): Restituisce lista di *Entity Tour.*
- getTourDetails(tourId): Restituisce dettagli complete e lista di *Stop.*
- createTour(tourData, organizerId): Crea un tour in stato "Draft".
- approveTour(tourId): Cambia stato tour in "Published".

4.3. Booking Service

- addToCart(sessionId, tourId, quantity): Aggiunge item al carrello temporaneo.
- checkout(sessionId, shippingDatam paymentData): Converte il carrello in un ordine persistente *Booking.*
- generateVoucher(bookingId): Crea il pdf del biglietto.

5. Glossary

- **Boundary:** Componente che interfaccia il sistema con un attore esterno (UI).
- **Control:** Componente che orchestra la logica di un caso d'uso.
- **Entity:** Oggetto che rappresenta dati persistenti significativi per il dominio.
- **Tour:** Pacchetto viaggio tematico (Entity principale).
- **Booking:** Oggetto che traccia sia il carrello temporaneo che l'ordine confermato.
- **Guest:** Utente non autenticato.
- **Organizzatore:** Utente fornitore che crea i tour.

Revision History

Data	Versione	Descrizione	Autore
25/11/2025	1.0	Creazione SDD	Vincenzo Chiocca

Partecipanti:

Nome	Matricola
Vincenzo Chiocca	**0512119182**
Salvatore Merola	**0512120979**