Università degli Studi di Salerno
Corso di Ingegneria del Software

ResolveIT
System Design Document
Versione 1.1

Data: 24/11/2025

Partecipanti:

Nome	Matricola
Samuele Nacchia	0512119128
Andrea Generale	0512119134
Mario Di Feo	0512119320

Scritto da:	MDF, AG, SN

Revision History

Data	Versione	Descrizione	Autore
18/11/2025	0.5	Introduction e Current software architecture	AG, SN
23/11/2025	1.0	Proposed Software Architecture	MDF, AG, SN
24/11/2025	1.1	Subsystem Services e revisione documento	MDF, AG, SN

Indice

1. Introduction

1.1 Purpose of the system

L'obiettivo del sistema è quello di creare una piattaforma web che permetta agli utenti di un sistema o di una piattaforma di richiedere assistenza agli operatori tecnici tramite dei ticket. Per i contesti che lo adotteranno si vuole quindi creare un ambiente generale e personalizzabile, in modo da poter fornire supporto specifico al numero di utenti maggiore possibile. In particolare, un gestore (definito in fase contrattuale) si occuperà della gestione e della personalizzazione. Le principali interazioni avverranno tra i clienti, che presenteranno delle problematiche riscontrate, via ticketing, e gli operatori che li risolveranno.

1.2 Design goals

1. **Reliability**
L'architettura sarà progettata per garantire la consistenza dei dati e la sicurezza delle transazioni. Implementeremo meccanismi di validazione server-side rigorosi per prevenire stati inconsistenti del database causati da malfunzionamenti o input malevoli (come SQL-injection).

2. **Modifiability**
Il sistema adotterà un'architettura modulare MVC.

3. **Maintainability**
Il codice sarà scritto seguendo gli standard di 'Clean Code' e sarà ampiamente commentato. L'uso di log informativi permetterà agli sviluppatori di diagnosticare rapidamente errori senza esporre dettagli tecnici agli utenti finali.

4. **Understandability**
Verranno utilizzate convenzioni di denominazione chiare per variabili e funzioni (es. createTicket, closeSession), affinché la logica di gestione dei ticket sia intuitiva per qualsiasi sviluppatore prenda in carico il progetto.

5. **Adaptability**
Il sistema utilizzerà file di configurazione esterni per gestire parametri come connessioni al database o timeout delle sessioni, permettendo al software di essere installato in ambienti diversi.

6. **Reusability**
I componenti dell'interfaccia utente (come i form di input) e le librerie di validazione dei dati saranno sviluppati come moduli indipendenti per poter essere riutilizzati in future estensioni del portale aziendale.

7. **Efficiency**
Il design mira a minimizzare l'uso di risorse server durante le query al database, ottimizzando le interrogazioni per la ricerca dei ticket e la gestione degli utenti, garantendo tempi di risposta rapidi anche con molti utenti connessi.

8. **Portability**
L'interfaccia utente sarà sviluppata in modo responsive garantendo quindi il rendering corretto su desktop, tablet e smartphone.

9. **Traceability of requirements**
Ogni modulo del software sarà mappato direttamente a un requisito funzionale.

10. **Fault tolerance**
Il sistema gestirà le eccezioni in modo che un errore in una singola operazione non blocchi l'intera applicazione, ma restituisca un messaggio coerente all'utente mantenendo il servizio attivo.

11. **Backward-compatibility**
Se in futuro verranno aggiornate le API del sistema, verrà mantenuto il supporto per i formati di dati dei ticket storici, garantendo che le vecchie segnalazioni rimangano leggibili e gestibili.

12. **Cost-effectiveness**
Il progetto privilegerà l'uso di tecnologie Open Source consolidate per database e server web, riducendo i costi di licenza senza compromettere la sicurezza o le prestazioni.

13. **Robustness**
Il design pone massima priorità alla validazione degli input. Il sistema sarà 'robusto' contro dati errati o tentativi di manipolazione, sanificando ogni input nei form prima dell'elaborazione per prevenire crash e vulnerabilità.

14. **High-performance**
Per garantire la disponibilità continua, l'architettura sarà leggera e priva di colli di bottiglia bloccanti, assicurando che il caricamento della dashboard e l'invio dei ticket avvengano in tempi ridotti anche sotto carico.

1.3 Definitions, acronyms, and abbreviations

Quando si fa riferimento a un **ticket** si sta parlando di una richiesta, inviata digitalmente da un cliente tramite il nostro sistema web.

Se si fa riferimento ad un **Visitatore**, si sta parlando di un attore che non avendo un account registrato nel sistema, può registrarsi alla piattaforma come cliente o effettuare l'accesso alla piattaforma e diventare un utente.
Se si fa riferimento ad un **Utente**, senza specificare a quale categoria appartiene, allora si sta parlando di un generico utente registrato.
Se si fa riferimento ad un **Cliente** allora si sta parlando di un Utente cliente che può richiedere assistenza tramite l'apertura di ticket.
Se si fa riferimento ad un **Operatore** allora si sta parlando di un utente registrato che può gestire i ticket aperti dai clienti.
Se si fa riferimento ad un **Gestore** allora si sta parlando di un utente gestore che può gestire gli account di operatori e clienti e gestire le categorie dei ticket.

1.4 References

Come idea di riferimento è stata presa in considerazione la piattaforma helpdesk utilizzata da UNISA.

1.5 Overview

I contenuti del System Design Document sono stati così strutturati:

- Situazione attuale del sistema proposto / esistenti affini
- Sistema proposto: introduzione, decomposizione in sottosistemi, hardware-software mapping, lato persistenza, autenticazione/sicurezza, funzionamento dei sottosistemi, boundary conditions
- Servizi offerti dai sottosistemi

2. Current software architecture

Molte aziende fornitrici di beni e servizi si avvalgono di un sistema di assistenza remota molto spesso sviluppati in proprio o molto rudimentali. Eventualmente fornendo poca uniformità di utilizzo dei vari sistemi di assistenza per gli utenti che, realisticamente, riscontrano problemi su piattaforme disconnesse tra loro, risultando caotiche per gli utenti e creando disagi nella richiesta di assistenza.

3. Proposed software architecture

3.1 Overview

Il sistema proposto è basato sullo stile architetturale Three Tier combinato con il design pattern
Model View Controller, implementato utilizzando Spring MVC. Nello sviluppo del sistema verranno usati HTML5, CSS3 per la parte di front-end. Per la logica applicativa e quindi il back-end sarà utilizzato Java SPRING.

Per la gestione del database saranno usati:

- Spring JPA per il collegamento al database.
- MySQL e SQL per il database

3.2 Subsystem decomposition

I sottosistemi individuati sono:

- **Registrazione**: si occupa di gestire le registrazioni degli utenti clienti e operatori.

- **Autenticazione**: è responsabile del login e del logout.

- **Gestione ticket**: si occupa della gestione dei ticket, creazione, eliminazione, assegnazione e risoluzione.

- **Gestione categorie**: è responsabile della creazione, modifica ed eliminazione delle categorie dei ticket.

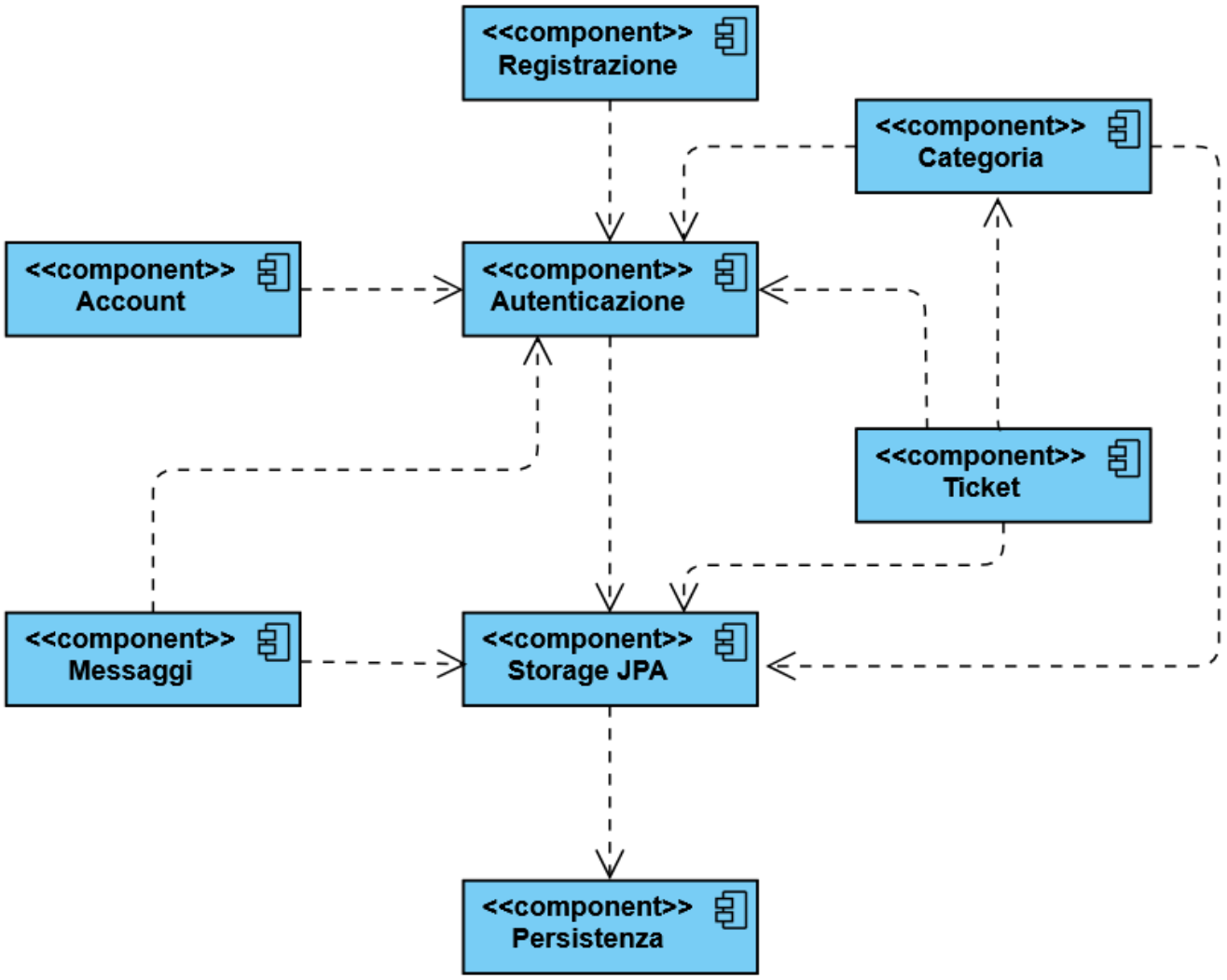
- **Gestione account**: è responsabile dell'eliminazione degli account, della visualizzazione e modifica dei dati personali e del ripristino password.

- **Messaggistica**: si occupa della gestione dei messaggi associati ad un determinato ticket, in particolare, creazione e visualizzazione dei messaggi.

- **Persistenza**: gestisce la persistenza dei dati con un database.

- **Storage JPA**: si interpone tra i vari sottosistemi ed il sottosistema di persistenza.

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un component diagram UML.



• Storage JPA verrà gestito da Spring Data JPA

• Persistenza sarà gestita attraverso un DBMS relazionale MySQL.

Di seguito una vista dettagliata di ciascun sottosistema evidenziando le componenti principale:

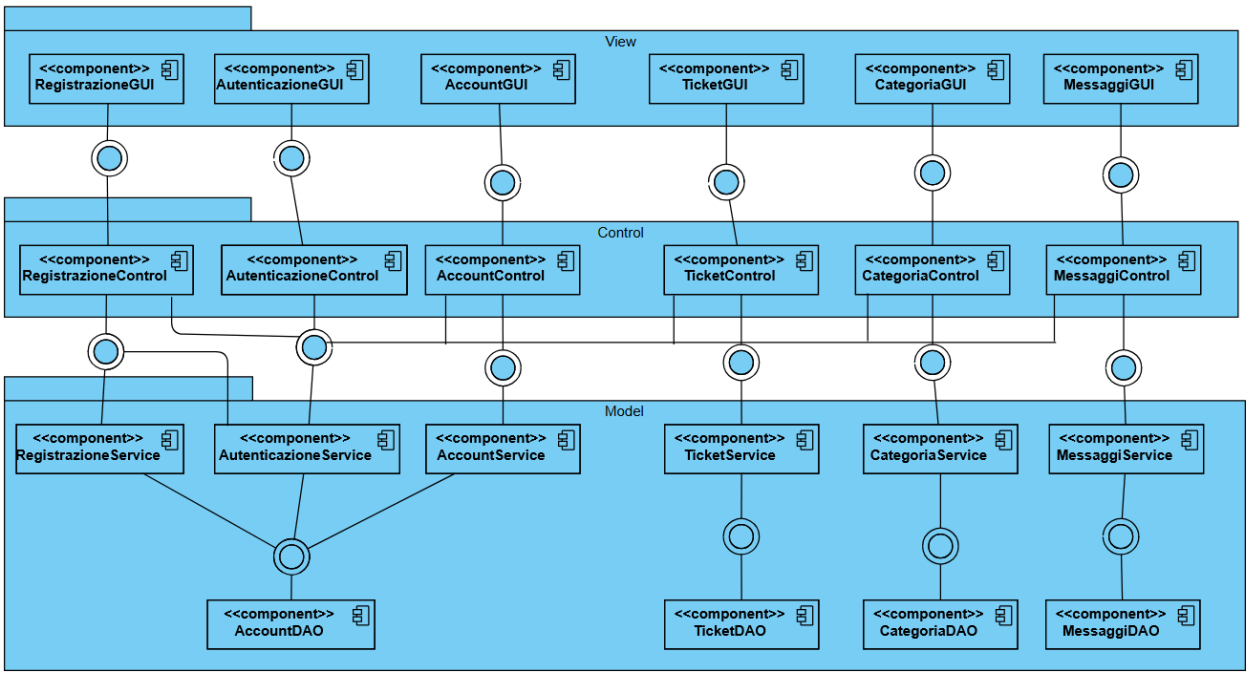
• GUI: Graphic User Interface, che contiene le varie view che saranno renderizzate per creare le pagine web da mostrare agli utenti.

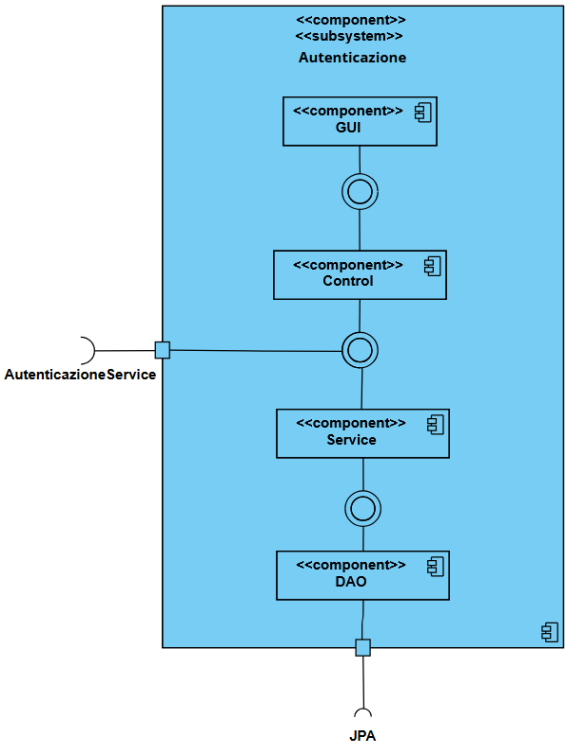
• Controller: si occupa della logica per il controllo del sistema.

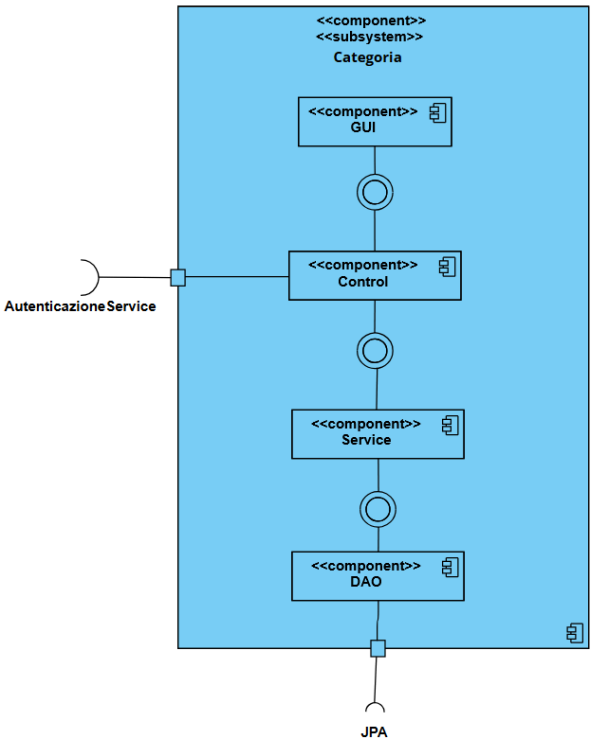
• Service: si occupa della logica di business.

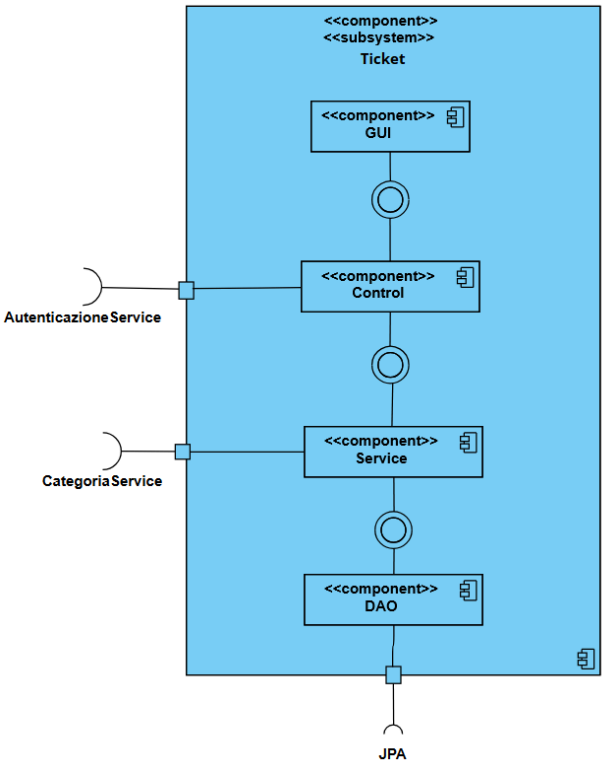
• DAO: Data Access Object, che si occupa di fornire accesso ai dati persistenti.

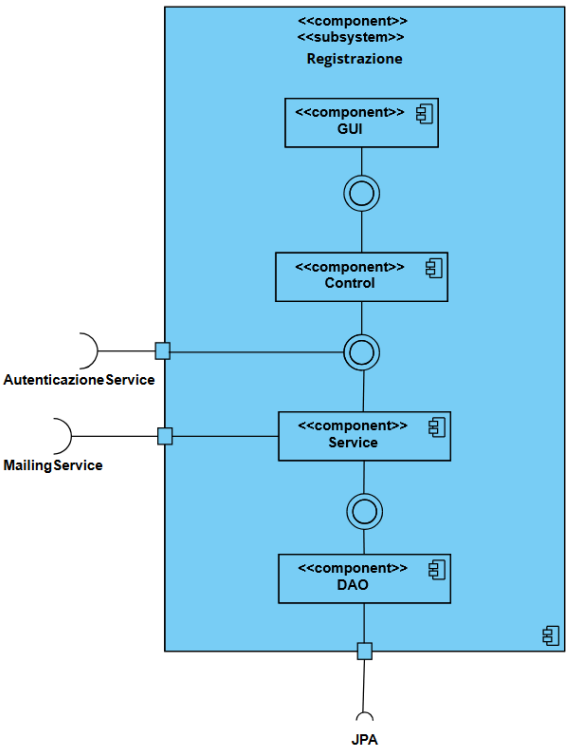
Diagramma architetturale

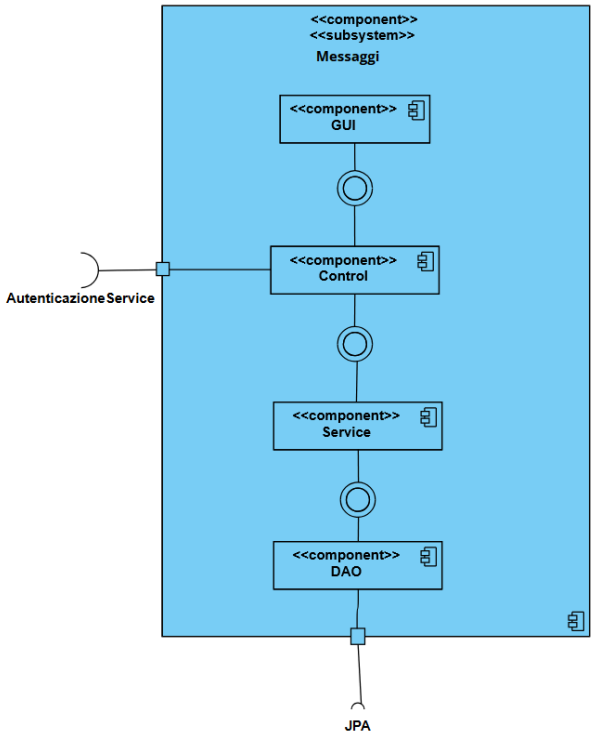


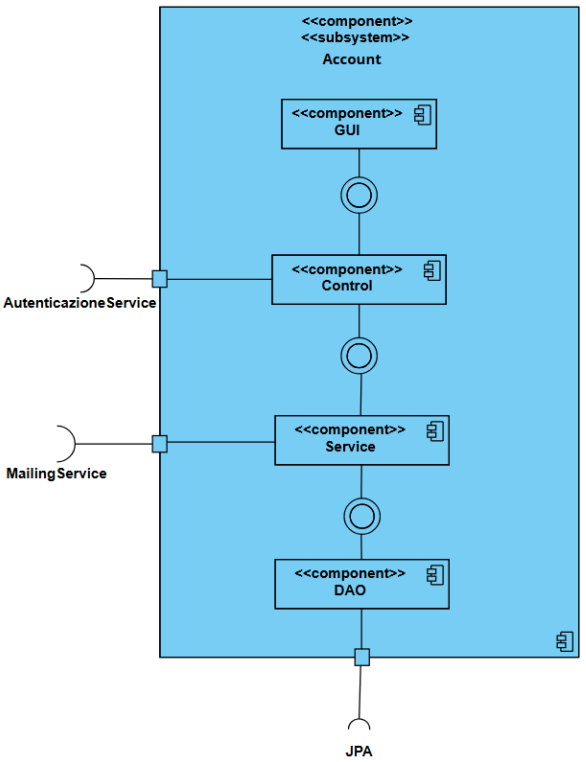





<<component>>
<<subsystem>>
Ticket
<<component>>
GUI
<<component>>
Control
<<component>>
Service
<<component>>
DAO
AutenticazioneService
CategoriaService
JPA


<<component>>
<<subsystem>>
Registrazione
<<component>>
GUI
<<component>>
Control
<<component>>
Service
<<component>>
DAO
AutenticazioneService
MailingService
JPA


<<component>>
<<subsystem>>
Messaggi
<<component>>
GUI
<<component>>
Control
<<component>>
Service
<<component>>
DAO
AutenticazioneService
JPA

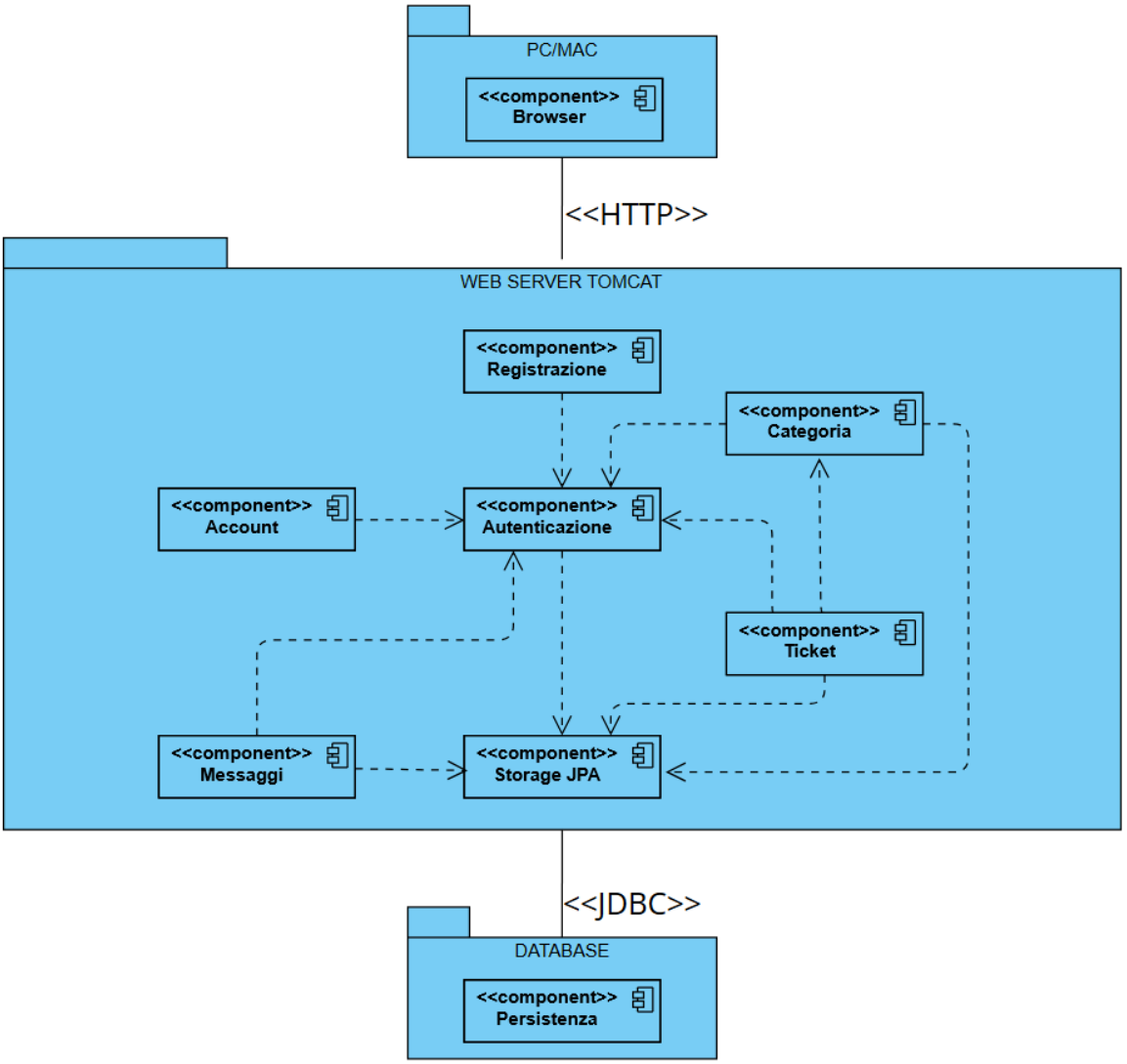

<<component>>
<<subsystem>>
Account
<<component>>
GUI
<<component>>
Control
<<component>>
Service
<<component>>
DAO
AutenticazioneService
MailingService
JPA

3.3 Hardware/software mapping

L’ applicazione web che verrà sviluppata si basa su una piattaforma hardware costituita da un server che risponde alle richieste effettuate dai clienti da una qualsiasi macchina con un browser ed una connessione ad Internet.

Il sistema da noi pensato è una applicazione web residente su un web server.

Di seguito un UML deployment diagram che descrive il mapping hardware/software.



3.4 Persistent data management

Per la gestione dei dati persistenti del sistema viene utilizzato un database relazionale, per garantire la consistenza e la coerenza dei dati con un accesso agevole, il tutto usando un DBMS basato su SQL.

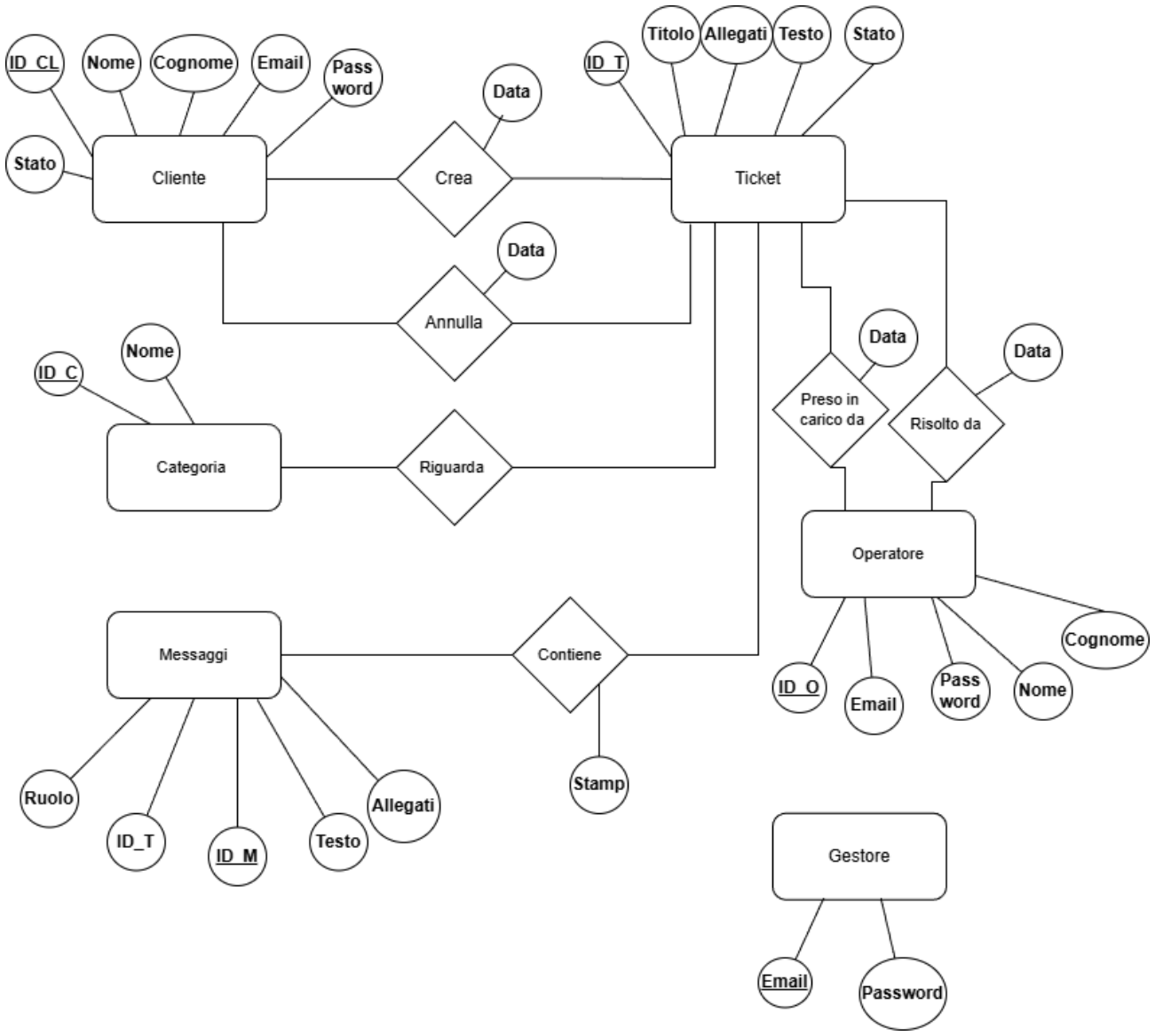
Tale scelta garantisce al sistema:

- **Integrità e consistenza dei dati**
  Il DBMS garantisce la consistenza e l'integrità dei dati tramite la possibilità di specificare dei vincoli, i quali verranno controllati e fatti rispettare dal DBMS.
- **Transazioni affidabili**
  Un DBMS basato su SQL ci garantisce il rispetto delle proprietà ACID sulle transazioni, ovvero Atomicità, Consistenza, Isolamento e Durabilità.
- **Concorrenza e controllo di accesso**
  Utilizzando un DBMS SQL più utenti possono accedere in modo concorrente al database, impedendo conflitti e usando accessi differenziati in base a ruoli e privilegi
- **Scalabilità e crescita dei dati**
  I DBMS SQL supportano la scalabilità del sistema tramite indicizzazione, caching, ottimizzazione delle query e sharding.
- **Backup e sicurezza**
  Il DBMS SQL offre Backup automatici, opzioni di ripristino in caso di crash, crittografia dei dati lato DB e audit log (registrazioni cronologiche degli eventi in un sistema).

L'analisi del Class Diagram ha permesso di identificare le entità persistenti: **Cliente, Ticket, Operatore, Categoria, Gestore e Messaggio**. Lo schema ER del database rappresenta le relazioni tra di esse.

L'accesso al database è effettuato tramite componenti **DAO** basati su **JDBC**, che incapsulano le operazioni CRUD e la gestione delle transazioni.

Schema ER del database



Dizionario dei Dati

Dal Class Diagram ricaviamo come entità persistenti sul database le seguenti:

Nome Entità	**Ticket**		
Descrizione	Contiene la richiesta di un utente ad un operatore in merito ad una problematica riscontrata		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_T	integer	PRIMARY KEY	
Titolo	varchar		NOT NULL
ID_C	integer	FOREIGN KEY	Riferimento a Categoria (ID_C)
ID_CL	integer	FOREIGN KEY	Riferimento a Cliente (ID_CL)
DataCreazione	date		NOT NULL
DataAnnullamento	date		Default Null

DataInCarico	date		Default Null
DataResolved	date		Default Null
Allegati	blob		
Testo	text		NOT NULL

Nome Entità	**Cliente**		
Descrizione	Contiene i dati relativi a un utente che apre i ticket		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_CL	integer	PRIMARY KEY	
email	varchar		NOT NULL, UNIQUE
nome	varchar		NOT NULL

cognome	varchar		NOT NULL
password_hash	varchar (255)		NOT NULL

Nome Entità	**Operatore**		
Descrizione	Contiene i dati relativi a un utente che prende in carico e risolve i ticket		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_O	integer	PRIMARY KEY	
email	varchar		NOT NULL, UNIQUE
nome	varchar		NOT NULL
cognome	varchar		NOT NULL
password_hash	char (255)		NOT NULL

Nome Entità	**Messaggio**		
Descrizione	Contiene i dati relativi ai messaggi scambiati tra cliente e operatore nel ticket		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_M	integer	PRIMARY KEY	
ID_T	integer	FOREIGN KEY	Riferimento aTicket (ID_T)
Stamp	datetime		NOT NULL
Ruolo	varchar		NOT NULL
Allegati	blob		Default Null
Testo	text		NOT NULL

Nome Entità	Categoria		
Descrizione	Contiene il nome della categoria a cui possono far parte i ticket		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
ID_C	integer	PRIMARY KEY	
Nome	varchar		NOT NULL, UNIQUE

Nome Entità	Gestore		
Descrizione	Contiene le credenziali del gestore del sistema		
Nome campo	**Tipo**	**Vincolo di chiave**	**Altri vincoli**
Email	varchar	PRIMARY KEY	
Password_Hash	varchar(255)		NOT NULL

3.5 Access control and security

Servizio	Utenti	
	Gestore	Visitatore
Registrazione		Registrazione cliente
Autenticazione	Log-in Log-out	
Gestione categorie	CreaCategoria ModificaCategoria EliminaCategoria	
Gestione account	CreaAccountOperatore EliminaAccountOperatore EliminaAccountCliente Visualizzazione lista account	

Servizio	Utenti	
	Cliente	Operatore
Registrazione		Conferma registrazione
Autenticazione	Log-in Log-out	Log-in Log-out
Gestione account	MyProfile ModificaDati Elimina Account Modifica dei dati Ripristino Password	-
Gestione Ticket	CreaTicket EliminaTicket Visualizzazione myTickets	AssignNewTicket PrendiInCarico Risolvi/assolvi
Messaggi	Crea messaggio Visualizza messaggio	Crea messaggio Visualizza messaggio

3.6 Global software control

ResolveIT, in quanto web-application interattiva, utilizza un meccanismo di controllo event-driven. Le azioni dell'utente sulla GUI scatenano eventi che vengono gestiti da handler dedicati. Questi handler delegano la logica di controllo al sottosistema di Controllo, il quale infine invoca i Servizi per l'esecuzione della logica applicativa

3.7 Boundary conditions

System start-up:

Il sistema è operativo quando: tutti i servizi applicativi sono attivi e connessi al database persistente, le sessioni utente sono inizialmente terminate. Tra le dipendenze esterne rispetto all'operatività abbiamo la connettività di rete stabile per una buona connessione al database e in generale un tempo di risposta ottimizzato.

System shut-down

Allo spegnimento il sistema esegue tutti i controlli possibili riguardo a sicurezza, transazioni e stato consistente e durevole dello strato persistente. Dopodiché non deve più accettare nuove connessioni in ingresso. Per quanto riguarda le transazioni si aspetta che tutte vengano completate con successo. Vengono poi chiuse le sessioni utente e rilasciate le risorse legate a memoria e database.

System failure

In caso di fallimento critico del sistema (causato da problemi di sicurezza, transazionali, di persistenza o hardware), l'obiettivo principale è duplice: garantire la consistenza dei dati e ripristinare la disponibilità del servizio. Alle transazioni rimaste in sospeso viene applicato un rollback automatico e le sessioni utente vengono ovviamente invalidate. Per il ripristino del sistema l'obiettivo è minimizzare il disservizio per gli utenti. Questo viene implementato attraverso il reindirizzamento ad un'istanza funzionante attiva o in generale verso nodi ridondanti (di backup).

4. Subsystem services

Sottosistema di Registrazione

Servizio	Descrizione	Interfaccia
Registrazione Cliente	Permette ai visitatori di registrarsi al sistema	RegistrazioneService
Conferma Account	Permette agli operatori di confermare la propria mail associata	RegistrazioneService

Sottosistema di Autenticazione

Servizio	Descrizione	Interfaccia
Login	Permette ad utenti ed operatori di effettuare l'accesso al sistema	AutenticazioneService
Logout	Permette ad utenti ed operatori di effettuare la disconnessione dal sistema	AutenticazioneService

Sottosistema di Messaggistica:

Servizio	Descrizione	Interfaccia
Crea Messaggio	Permette ad utenti ed operatori di inviare messaggi	MessaggiService
Visualizza Messaggio	Permette ad utenti ed operatori di visualizzare l'elenco di messaggi relativi a un ticket	MessaggiService

Sottosistema di Gestione categorie

Servizio	Descrizione	Interfaccia
Creazione Categoria	Permette al gestore di creare categorie	CategoriaService
Modifica Categoria	Permette al gestore di modificare le categorie	CategoriaService
Eliminazione Categoria	Permette al gestore di eliminare le categorie	CategoriaService

Sottosistema di Gestione account

Servizio	Descrizione	Interfaccia
Creazione Account Operatore	Permette al gestore del sistema di creare un nuovo account operatore	AccountService
Visualizzazione profilo	Permette agli utenti di visualizzare i propri dati utente	AccountService
Visualizzazione lista account	Permette al gestore di visualizzare una lista di tutti gli account registrati	AccountService
Eliminazione account	Permette al gestore di eliminare un account	AccountService
Modifica dei dati	Permette agli utenti di modificare i dati utente	AccountService
Ripristino Password	Permette agli utenti,che hanno dimenticato la propria password, di sceglierne una nuova	AccountService

Sottosistema di Gestione ticket

Servizio	Descrizione	Interfaccia
Creazione Ticket	Permette agli utenti di creare dei ticket di richiesta supporto tecnico	TicketService
Annullamento Ticket	Permette agli utenti di annullare i ticket da loro creati	TicketService
Presa in Carico Ticket	Permette agli operatori di selezionare i ticket sui quali lavorare	TicketService
Risoluzione Ticket	Permette agli operatori di contrassegnare i ticket su cui hanno lavorato come risolti	TicketService
Visualizzazione ticket aperti dall'utente	Permette agli utenti di visualizzare la lista dei ticket da loro aperti	TicketService
Visualizzazione ticket disponibili	Permette agli operatori di visualizzare la lista dei ticket non ancora presi in carico	TicketService
Visualizzazione ticket in carico	Permette agli operatori di visualizzare la lista dei ticket presi da loro in carico	TicketService