



# SDD

# System Design

# Document

Foundly

<b>Riferimento</b>	C07_SDD_ver.1.0
<b>Versione</b>	0.1
<b>Data</b>	03/11/2025
<b>Destinatario</b>	Prof. Gravino Carmine
<b>Presentato da</b>	CN05 Team



## Revision History

Data	Versio ne	Descrizione	Autori
3/11/2025	0.1	Prima stesura	SL, SDP, NN
17/12/2025	0.2	Revisione	SL, SDP, NN



## Team members

---

Nome	Acronimo	Informazioni di contatto
Salvador Davide Passarelli	SDP	s.passarelli2@studenti.unisa.it
Salvatore Lepore	SL	s.lepore11@studenti.unisa.it
Natale Nappi	NN	n.nappi8@studenti.unisa.it



## Sommario

Revision History.....	2
Team members.....	3
<b>1 Introduzione.....</b>	<b>4</b>
1.1 Scopo del sistema.....	4
1.2 Obiettivi di Design (Design Goals).....	6
1.3 Definizioni, acronimi, e abbreviazioni.....	11
1.4 Riferimenti.....	11
1.5 Organizzazione del documento.....	12
<b>2 Architettura del sistema corrente.....</b>	<b>13</b>
<b>3 Architettura del sistema proposto.....</b>	<b>13</b>
3.1 Panoramica sulla sezione.....	13
3.2 Decomposizione in sottosistemi.....	14
Mapping hardware/software.....	18
Gestione dei dati persistenti.....	19
Controllo degli accessi e sicurezza.....	27
Controllo globale del software.....	28
Condizioni limite.....	29
<b>4 Servizi dei sottosistemi.....</b>	<b>32</b>
<b>5 Design Pattern.....</b>	<b>36</b>
5.1. Pattern Strutturale: Facade.....	36
5.2. Pattern Comportamentale: Observer.....	37
5.3. Pattern Creazionale: Abstract Factory.....	38
<b>6 Glossario.....</b>	<b>39</b>



# 1 Introduzione

---

## 1.1 Scopo del sistema

Foundly semplifica segnalazione, ricerca e restituzione di oggetti e animali smarriti, costruendo una rete di cittadini e negozi “Drop-Point” sotto la supervisione di uno o più amministratori. Il sistema accetta la registrazione di utenti e attività commerciali (Drop-Point), permette tramite dei reclami (Secure Claim) la restituzione e premia i comportamenti virtuosi con una scoreboard pubblica.

### Componenti principali del sistema:

1. **Gestione Utenti e Autenticazione:** registrazione, login, profilo, recupero password.
2. **Bacheca Segnalazioni e Ricerca:** pubblicazione di oggetti/animali ritrovati con foto, luogo, domande di verifica; lista e filtri di ricerca.
3. **Secure Claim e Restituzione:** invio del claim con risposte alle domande; contatto diretto tra le parti; chiusura con doppia conferma per consegna diretta.
4. **Consegna tramite Drop-Point:** quando il claim è accettato per un oggetto, il sistema genera un **Codice di Consegna** usato dall'operatore per deposito e ritiro; tracciamento degli eventi di deposito/ritiro.
5. **Gestione Amministrativa:** approvazione/sospensione Drop-Point, moderazione segnalazioni e gestione utenti.
6. **Community/Scoreboard:** punteggio automatico alla chiusura delle restituzioni e classifica pubblica.



## 1.2 Obiettivi di Design (Design Goals)

In questa sezione si presentano i **Design Goals** di *Foundly*, ovvero le qualità fondamentali su cui il sistema deve focalizzarsi.

Essi vengono formalizzati esplicitamente per garantire che ogni decisione di progettazione sia coerente con lo stesso insieme di obiettivi di qualità.

Seguendo le linee guida di **Bernd Bruegge & Allen Dutoit – Object-Oriented Software Engineering: Conquering Complex and Changing Systems** e le raccomandazioni del portale **ISO/IEC 25010:2011 – Systems and Software Quality Models** (pubblicato da ISO e consultabile online su [iso.org](http://iso.org)), i design goals sono classificati nelle seguenti categorie:

- **Performance** – Riguarda i vincoli di tempo di risposta e uso delle risorse nelle operazioni principali (pubblicazione segnalazione, ricerca, invio Secure Claim, gestione Drop-Point).
- **Dependability**: Comprende le proprietà di **affidabilità, sicurezza e robustezza** volte a prevenire guasti, perdita di dati o accessi non autorizzati.
- **Maintenance**: Indica la **facilità di manutenzione, estensione e aggiornamento** del sistema dopo il rilascio, promuovendo modularità e separazione dei livelli architetturali.
- **End User**: Include le qualità percepite dagli utenti finali, come **usabilità, accessibilità e coerenza dell'interfaccia**, essenziali per un'interazione semplice e intuitiva.

Ciascun design goal è descritto da:

- **Rank**, che ne specifica un valore di priorità compreso tra 1 e 16 (1 massima e 16 minima).
- **ID Design Goal**, un identificatore univoco e un nome esplicativo.
- **Descrizione**, una descrizione del design goal.
- **Categoria**, ovvero la categoria di appartenenza del design goal.
- **RNF di origine**, ovvero il requisito non funzionale che lo ha generato.



## Design goals

Rank	ID Design Goal	Descrizione	Categoria	RNF di origine
3	DG_1 Tempi di risposta	Le operazioni principali devono risultare rapide per l'utente; tempo di risposta $\leq 5$ s (95° percentile) per bachecca, dettaglio, creazione segnalazione e invio Secure Claim.	Performance	RNF_P_1
4	DG_2 Quantità di dati	Gestione affidabile di utenti, segnalazioni (oggetti/animali), Drop-Point, reclami e immagini $\leq 10$ MB per file	Performance	RNF_IN_1, RNF_IN_3
9	DG_3 Navigazione concorrente	Funzionamento regolare con almeno 50 sessioni concorrenti su singola istanza, senza errori funzionali percepiti.	Performance	RNF_P_1, RNF_IN_3
5	DG_4 Affidabilità delle operazioni	Le operazioni sono consistenti: o completate interamente o annullate con rollback su errore.	Dependability	RNF_A_1, RNF_IN_1
14	DG_5 Fallimento di sistema	In caso di errore, il Controller gestisce le eccezioni mostrando messaggi chiari senza esporre stacktrace; l'utente può riprovare senza perdita di dati persistenti.	Dependability	RNF_A_3
7	DG_6 Gestione permessi	Accesso alle funzioni limitato dal <b>ruolo</b> (User, Admin, Drop-Point) con controlli lato server su pagine e operazioni.	Dependability	RNF_A_5



15	DG_7 Disponibilità del sistema	Dopo interruzioni o riavvi, i dati già salvati restano consistenti e l'utente può riprendere le attività al nuovo accesso.	Dependability	RNF_A_4
1	DG_8 Sicurezza dei dati	Il Sistema deve proteggere i dati sensibili persistenti (es. password) cifrandoli tramite hashing	Dependability	RNF_SEC_1, RNF_SEC_2, RNF_SEC_3, RNF_LE_1
2	DG_9 Security by Design	Il sistema non deve contare sulla "Sicurezza tramite l'Oscurità" - tutte le tecnologie utilizzate per mantenere la sicurezza dei dati e del sistema devono essere sicure indipendentemente dal grado di conoscenza del funzionamento del sistema di un eventuale aggressore, per permettere la pubblicazione sicura dei codici sorgenti.	Dependability	RNF_LE_2, RNF_SEC_1, RNF_SEC_2, RNF_SEC_3
8	DG_10 Modularità	Il sistema deve garantire la separazione delle responsabilità tra View, Controller e Model per permettere la manutenzione o sostituzione di singoli componenti senza impatti globali.	Maintenance	RNF_S_1, RNF_IM_2
11	DG_11 Estendibilità	L'aggiunta di una nuova entità o funzionalità (es. un nuovo tipo di segnalazione) deve essere possibile estendendo le classi esistenti senza modificarne il codice sorgente (principio <b>Open/Closed</b> ), richiedendo modifiche a	Maintenance	RNF_S_2, RNF_S_3



		non più di 2 file di configurazione.		
6	DG_12 Apprendibilità (UX)	Un nuovo utente (profilo User), senza formazione precedente, deve essere in grado di completare il flusso di 'creazione segnalazione' in meno di 180 secondi e con un tasso di successo al primo tentativo superiore all'90% (misurato su un campione di test di almeno 5 utenti).	End User	RNF_U_1, RNF_U_4
10	DG_13 Interfaccia intuitiva	L'interfaccia utente della piattaforma deve permettere di eseguire azioni in modo chiaro e semplice, rendendo ben esplicita la funzionalità di ogni elemento visuale.	End User	RNF_U_2
12	DG_14 Feedback esplicito	Ogni azione all'interno della piattaforma in seguito ad un'interazione dell'utente deve comunicare un chiaro feedback allo stesso	End User	RNF_U_3
16	DG_16 Interfaccia con le Utility di Sistema per l'Amministrazione	Il Sistema deve interagire correttamente con le utility del sistema operativo sul quale viene eseguito il server.	Maintenance	RNF_IN_1, RNF_IN_2, RNF_SEC_1



## Trade-off

Trade-off	Descrizione
Performance del Database Cloud vs costi	L'uso di un database in Cloud lega le prestazioni direttamente ai costi. Per avere velocità elevata e risorse dedicate servirebbero piani di abbonamento costosi. Per rispettare il budget limitato, il sistema utilizza un piano 'Basic': questo implica che, se ci sono troppe richieste contemporanee, il database potrebbe rallentare a causa dei limiti imposti dal provider.
Tempi di risposta vs sicurezza	L'implementazione di protocolli di sicurezza (es. hashing complesso delle password) introduce un overhead computazionale. Questo può incrementare leggermente i tempi di latenza rispetto a un sistema non protetto, rendendo necessario un margine di tolleranza nei tempi di risposta



## 1.3 Definizioni, acronimi, e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento corrente:

- **Sottosistema:** un sottoinsieme dei servizi del dominio applicativo, formato da servizi legati da una relazione funzionale.
- **Design Goal:** le qualità sulle quali il sistema deve essere focalizzato.
- **Dati Persistenti:** dati che sopravvivono all'esecuzione del programma che li ha creati e che dunque vengono salvati.
- **Mapping Hardware/Software:** studio della connessione tra parti fisiche e logiche di cui si compongono il sistema.
- **SDD:** System Design Document
- **RAD:** Requirements Analysis Document

## 1.4 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- [Statement Of Work](#);
- [Business Case](#);
- [Requirements Analysis Document](#);
- [System Design Document](#);
- [Object Design Document](#);
- [Test Plan](#);
- [Matrice di tracciabilità](#);
- [Manuale di installazione](#);
- [Manuale utente](#);



## 1.5 Organizzazione del documento

Il presente documento di System Design consta di quattro sezioni:

**Introduzione:** Viene descritto in generale lo scopo del sistema, gli obiettivi di design che il sistema propone di raggiungere.

**Architettura software corrente:** Viene descritto lo stato attuale dell'architettura del software già presente.

**Architettura software proposta:** Viene descritto come il sistema sarà definito e partizionato in sottosistemi, il loro mapping Hardware/Software, la gestione dei dati persistenti. Verranno poi presentate la struttura dei singoli sottosistemi e le boundary conditions riguardanti l'intero sistema.

**Glossario:** Contiene la lista dei termini usati nel documento con annessa spiegazione.



## 2 Architettura del sistema corrente

---

Al momento non esiste una piattaforma unica che concentri tutte le funzionalità di Foundly (segnalazioni, secure claim con domande di verifica, consegna diretta/Drop-Point con codice, scoreboard) in un solo servizio. Il mercato è frammentato tra post sui social, gruppi locali, bacheche fisiche, moduli comunali/polizia e servizi verticali per animali smarriti; queste soluzioni non sono integrate né tracciabili in modo uniforme, quindi non esiste una reale architettura di riferimento con cui confrontare ragionevolmente il sistema.

## 3 Architettura del sistema proposto

---

### 3.1 Panoramica sulla sezione

Il sistema proposto adotta l'architettura Three-Tier combinata con il pattern Model-View-Controller (MVC). La separazione tra presentazione, logica di business e accesso ai dati migliora leggibilità, manutenzione e riuso.

- Presentazione (View): HTML5, CSS3, JavaScript, JSP/JSTL per la generazione delle pagine e chiamate AJAX/JSON verso i controller.
- Logica applicativa (Controller/Service): Java 17 su Jakarta Servlet in esecuzione su Apache Tomcat 11. Controller come Servlet, servizi applicativi come classi Java dedicate.
- Persistenza (Model/DAO): JDBC con MySQL managed (Aiven); connection pool, transazioni e vincoli di integrità (PK/FK/UNIQUE).

Motivazione della scelta: è una soluzione semplice e adatta a web app come Foundly; consente di mantenere interfaccia e logica disaccoppiate, garantendo tempi di risposta adeguati e una base solida per evoluzioni future.

Deployment: packaging WAR su singola istanza Tomcat; configurazioni esterne (variabili d'ambiente); comunicazioni HTTPS.



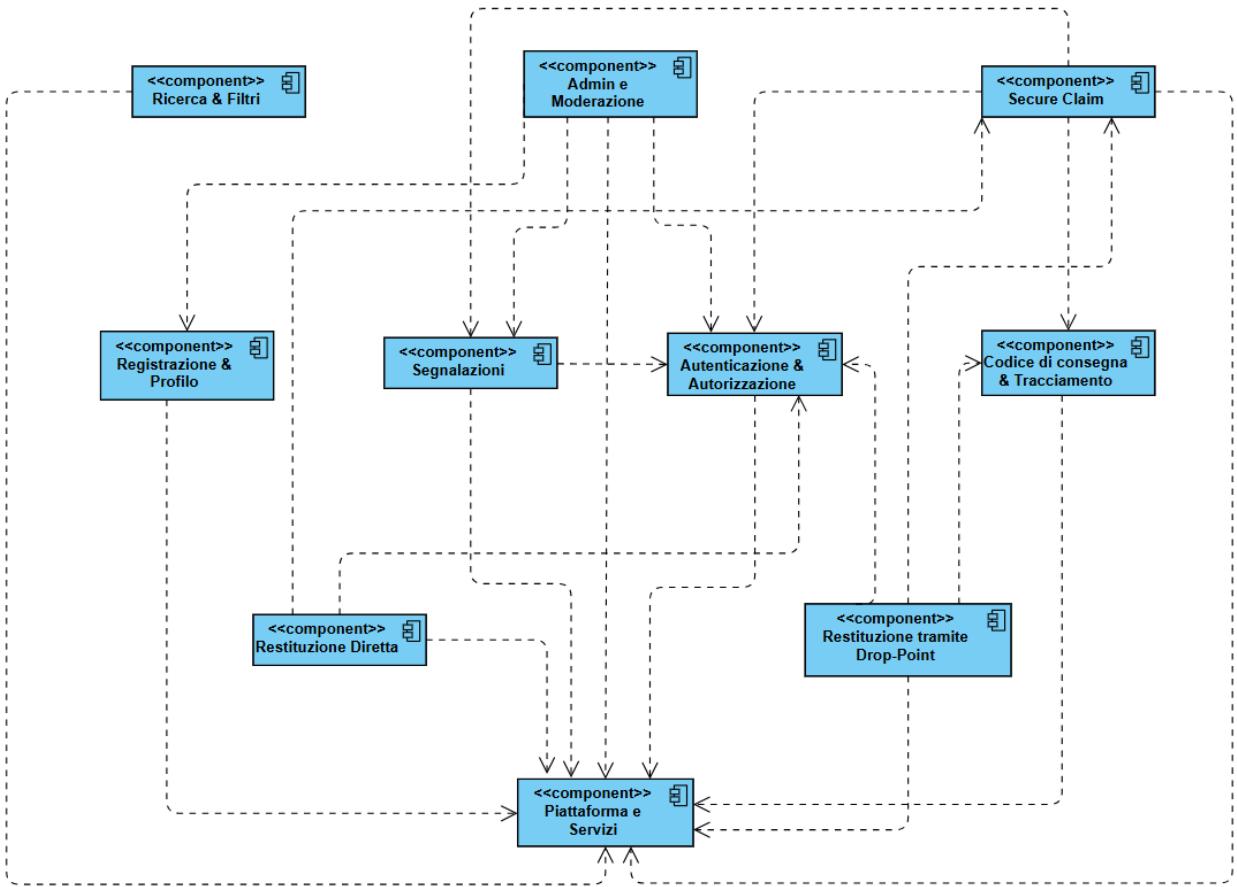
## 3.2 Decomposizione in sottosistemi

Il sistema è stato decomposto nei seguenti sottosistemi logici, in accordo con il *Component Diagram* dell'architettura:

- **Registrazione & Profilo** Gestisce il ciclo di vita degli account per utenti e attività commerciali (Drop-Point). Include la registrazione, la modifica dei dati personali, la visualizzazione del punteggio e la cancellazione.
- **Autenticazione e Autorizzazione** Gestisce la sicurezza degli accessi. Include le funzionalità di Login, Logout, Recupero Password e la gestione dei ruoli e dei permessi (User, Admin, Drop-Point) per l'accesso alle diverse aree riservate.
- **Segnalazioni** Gestisce l'intero ciclo di vita degli oggetti smarriti. Include la creazione, modifica, chiusura e cancellazione delle segnalazioni (oggetti o animali), con gestione delle foto, della geolocalizzazione e delle domande di verifica.
- **Ricerca & Filtri** Gestisce la consultazione pubblica della bacheca. Fornisce funzionalità di ricerca avanzata con filtri per categoria, area geografica e stato, oltre all'ordinamento dei risultati e alla visualizzazione di dettaglio.
- **Secure Claim** Gestisce il processo di rivendicazione. Include l'invio del reclamo con le risposte alle domande di verifica, il sistema di notifiche tra Finder e Owner e la gestione dello storico e degli stati del claim (In attesa, Accettato, Rifiutato).
- **Restituzione Diretta** Gestisce il flusso di restituzione per scambio a mano. Abilita la visualizzazione dei recapiti privati dopo l'accettazione del claim e gestisce il protocollo di "doppia conferma" (consegna e ricezione) per chiudere la segnalazione.
- **Restituzione tramite Drop-Point** Gestisce l'operatività logistica presso i punti di custodia. Include il pannello operatore per i Drop-Point e i flussi di lavoro per il deposito (Check-in) e il ritiro (Check-out) degli oggetti.
- **Codici di Consegna & Tracciamento** Modulo trasversale che gestisce la generazione sicura e la validazione dei codici univoci necessari per le operazioni presso i Drop-Point, monitorando gli stati (valido, usato, scaduto) e mantenendo il log delle operazioni.
- **Admin & Moderazione** Fornisce gli strumenti di supervisione per l'Amministratore. Include la valutazione delle richieste di registrazione dei Drop-Point (approvazione/rifiuto), la moderazione delle segnalazioni inappropriate, la gestione degli utenti e le configurazioni base della piattaforma.

- **Piattaforma & Servizi** Livello infrastrutturale che fornisce servizi comuni a tutti gli altri sottosistemi. Include la persistenza dei dati (DBMS MySQL via JDBC), il servizio di upload e gestione immagini, il servizio di invio email (notifiche/OTP) e l'audit basilare di sistema.

Sono mostrate di seguito le dipendenze tra i sottosistemi attraverso un Component Diagram UML.



Alcuni sottosistemi saranno gestiti da componenti COTS (Commercial Off-The-Shelf). In particolare:

- l'infrastruttura web sarà gestita dal **container servlet Apache Tomcat**;
- la persistenza sarà gestita da un **DBMS relazionale MySQL** ospitato in **cloud (Aiven)**;

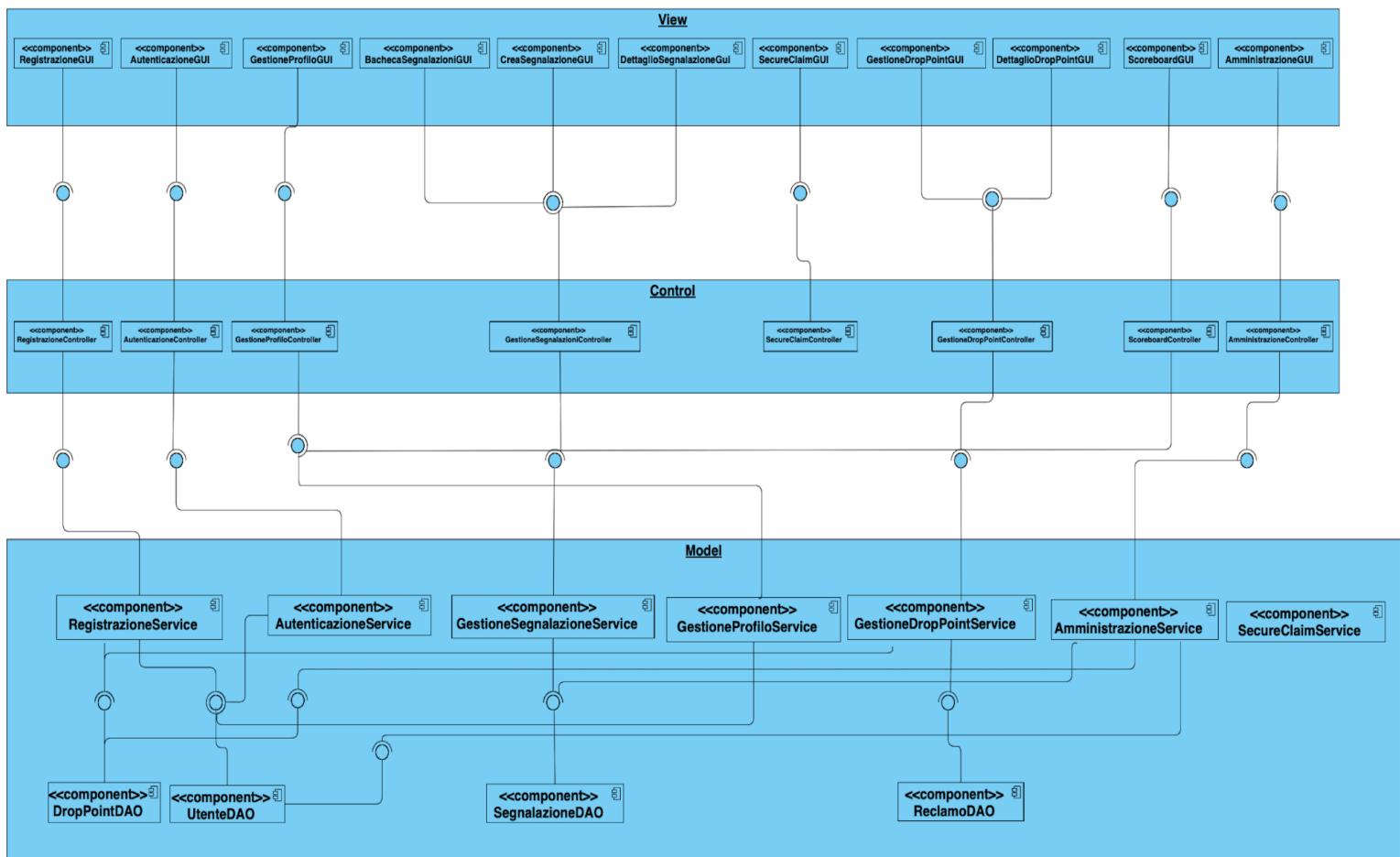


- l'accesso al database avverrà tramite **driver JDBC** e **connection pool** forniti da librerie standard del container.

Di seguito una vista dei principali sottosistemi interni al tier applicativo, in linea con l'architettura MVC adottata:

- **GUI:** contiene le view (pagine **JSP/CSS** e risorse statiche) che vengono renderizzate e inviate al browser dell'utente per la consultazione delle segnalazioni, la creazione dei reclami, la gestione dei Drop-Point e dell'area riservata.
- **Controller:** insieme di **Servlet / controller** che ricevono le richieste HTTP, invocano i servizi applicativi opportuni e selezionano la view da visualizzare.
- **Service:** classi di **logica di business** che encapsulano le operazioni del dominio (creazione segnalazione, invio reclamo, generazione codice di consegna, gestione punteggio utente, ecc.) coordinando i DAO e applicando le regole applicative.
- **DAO (Data Access Object):** componenti responsabili dell'**accesso ai dati persistenti** tramite JDBC, mappando le entità del dominio sulle corrispondenti tabelle del database MySQL.

## Diagramma architetturale

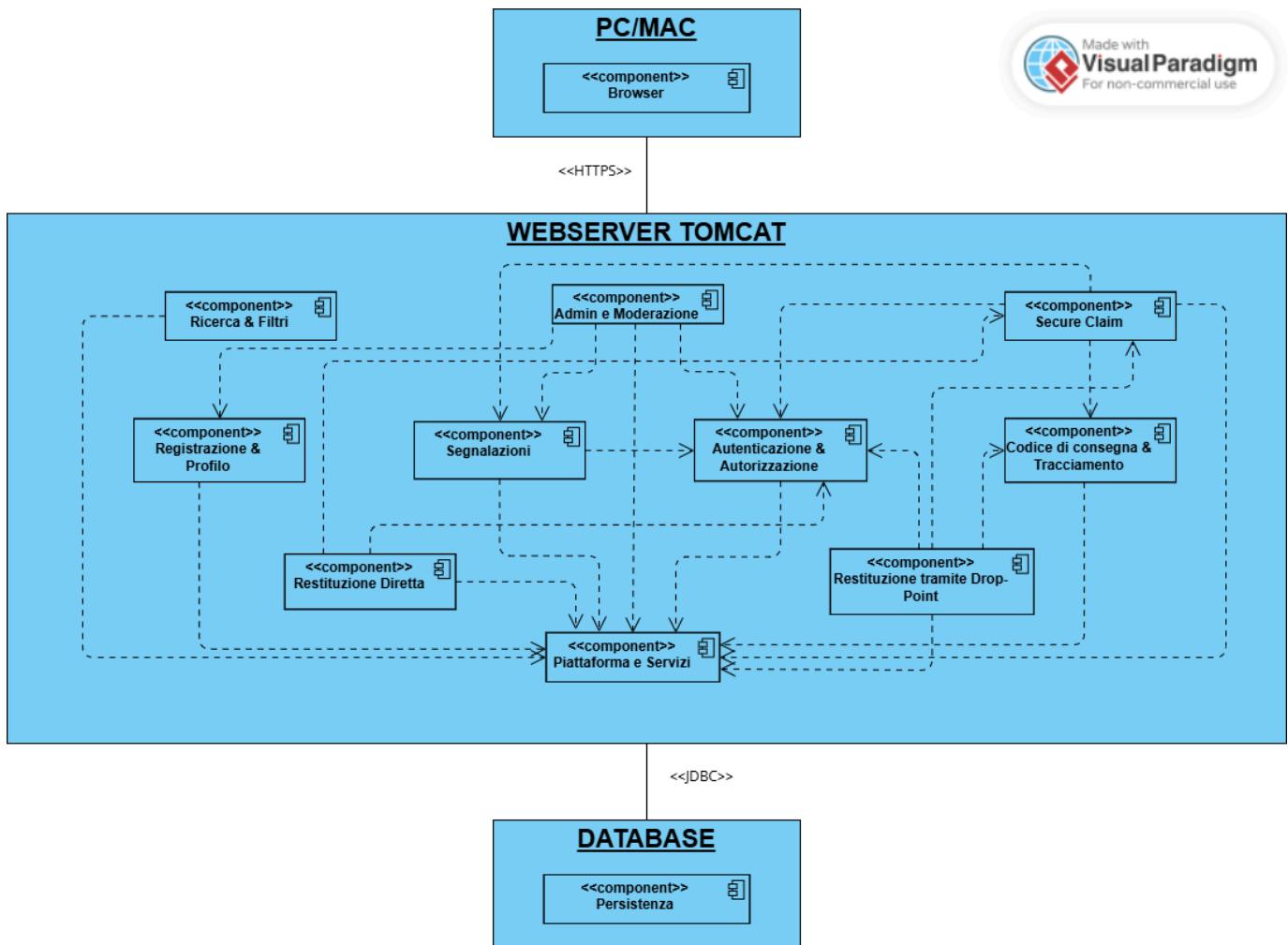


## Mapping hardware/software

L'applicazione web che verrà sviluppata si basa su una piattaforma hardware costituita da un unico server, che risponde alle richieste dei client provenienti da qualsiasi macchina dotata di browser e connessione a Internet.

Poiché il sistema è una web application non distribuita, esso risiede interamente su un solo nodo (web server).

Di seguito è riportato il deployment diagram UML che descrive il mapping tra componenti hardware e software.





## Gestione dei dati persistenti

### Introduzione

Per Foundly è stato scelto un **database relazionale MySQL in cloud (Aiven)**, in modo da garantire consistenza, sicurezza e collaborazione tra i membri del team.

L'uso di un DBMS permette di:

- applicare **vincoli di integrità** (chiavi primarie/esterne, unicità, ecc.) su utenti, segnalazioni, reclami e Drop-Point, mantenendo i dati coerenti;
- garantire **privatezza e controllo degli accessi** ai dati sensibili gestiti dal sistema;
- assicurare **affidabilità**, grazie a backup e meccanismi di ripristino in caso di guasti;
- gestire operazioni critiche (creazione segnalazioni, reclami, consegne) in modo **atomico**, evitando stati inconsistenti.

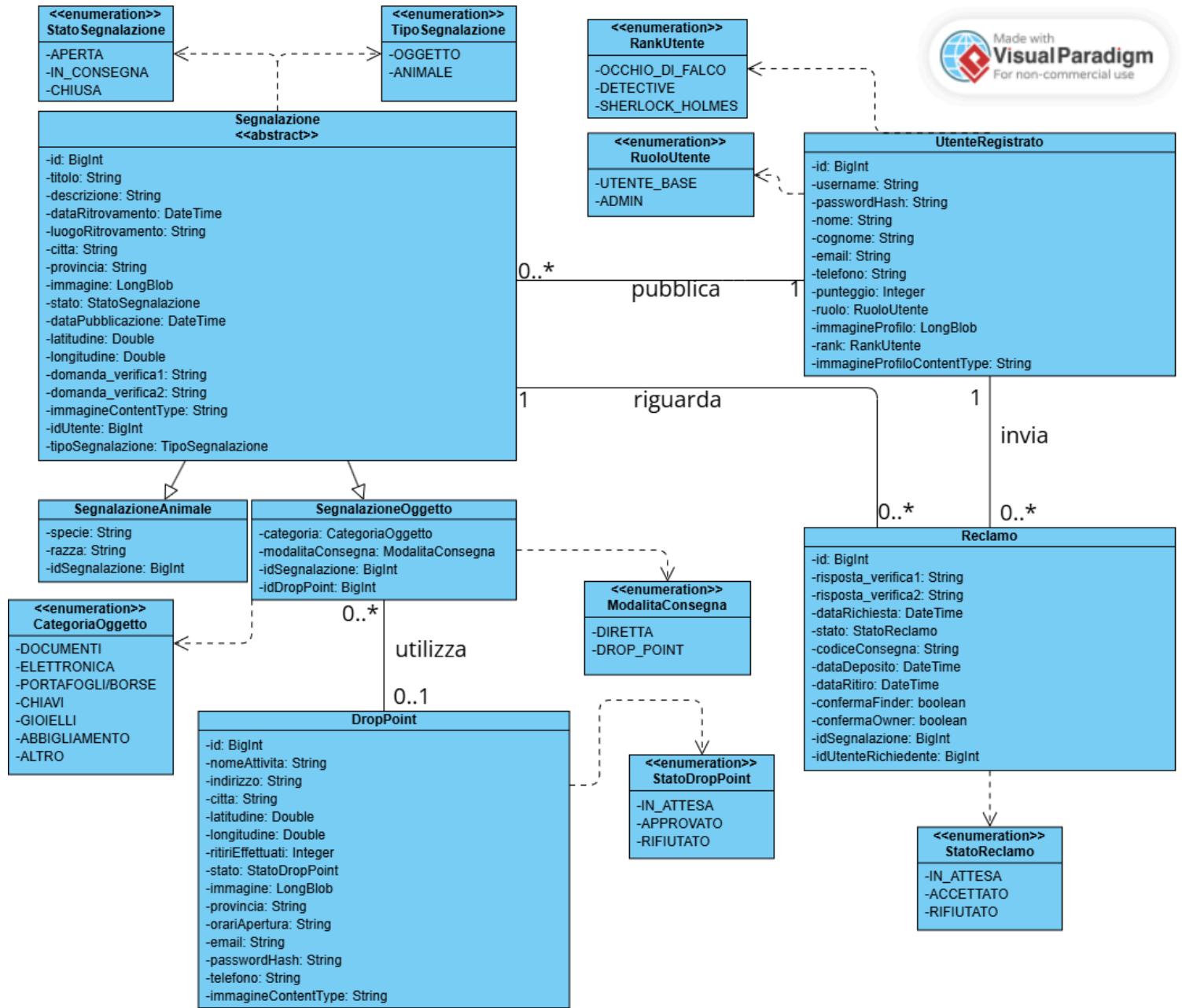
La scelta di un database in cloud, condiviso tra tutti i membri del gruppo, riduce i problemi legati alle installazioni locali, semplifica lo sviluppo collaborativo e rende più semplice l'evoluzione controllata dello schema dati di Foundly.

### CD\_SDD: Entity Class Diagram ristrutturato

Il Diagramma di Classe qui presentato è l'evoluzione del Modello Concettuale definito nel RAD. Questa ristrutturazione lo rende conforme agli standard di design per l'implementazione software.

Le modifiche principali sono state:

1. **Generalizzazione:** Il vincolo {xor} del modello RAD è stato eliminato creando la classe astratta **Segnalazione**. SegnalazioneAnimale e SegnalazioneOggetto ereditano da essa, risolvendo l'ambiguità tramite **Polimorfismo**.
2. **Tipizzazione:** I tipi generici (Date, String) sono stati sostituiti con tipi specifici per il codice (LocalDateTime, List<String>).
3. **Standardizzazione:** Sono state introdotte le **Enumerazioni** (<<enumeration>>) per definire tutti gli stati e i ruoli, garantendo chiarezza e coerenza nel codice.
4. **Identificazione:** Ogni entità possiede ora un **ID** univoco per la persistenza nel database.





## Dizionario dei dati

Nome Entità		UTENTE_REGISTERATO	
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	BIGINT	PRIMARY KEY	AUTO_INCREMENT
username	VARCHAR(50)		UNIQUE, NOT NULL
nome	VARCHAR(50)		NOT NULL
cognome	VARCHAR(50)		NOT NULL
email	VARCHAR(100)		UNIQUE, NOT NULL
telefono	VARCHAR(20)		
punteggio	INT		DEFAULT 0
ruolo	ENUM		NOT NULL Valori: 'UTENTE_BASE', 'ADMIN'. DEFAULT 'UTENTE_BASE'
immagine_profilo	LONGBLOB		
badge	ENUM		NOT NULL, Valori: 'OCCHIO_DI_FALCO', 'DETECTIVE', 'SHERLOCK HOLMES' DEFAULT "OCCHIO_DI_FALCO"
password_hash	VARCHAR(255)		NOT NULL



immagine_profilo _content_type	VARCHAR(50)		
-----------------------------------	-------------	--	--

Nome Entità		SEGNALAZIONE (Entità Padre)	
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	BIGINT	PRIMARY KEY	AUTO-INCREMENT
id_utente	BIGINT	FOREIGN KEY(id: UTENTE_REGISTRA TO)	NOT NULL
titolo	VARCHAR(100)		NOT NULL
descrizione	TEXT		NOT NULL
data ritrovamento	DATETIME		NOT NULL
luogo ritrovamento	VARCHAR(100)		NOT NULL
citta	VARCHAR(50)		NOT NULL
provincia	CHAR(50)		NOT NULL
immagine	LONGBLOB		
stato	ENUM		Valori: 'APERTA', 'IN_CONSEGNA', 'CHIUSA'. NOT NULL, DEFAULT "APERTA"
data pubblicazione	DATETIME		DEFAULT CURRENT_TIMESTAMP



latitudine	DOUBLE		
longitudine	DOUBLE		
domanda_verifica1	VARCHAR(255)		NOT NULL
domanda_verifica2	VARCHAR(255)		NOT NULL
immagine_content_type	VARCHAR(50)		
tipo_segnalazione	ENUM		NOT NULL Valori: 'OGGETTO', 'ANIMALE'.

Nome Entità <b>SEGNALAZIONE_ANIMALE (Entità Figlia)</b>			
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	BIGINT	PRIMARY KEY, FOREIGN KEY (ref. SEGNALAZIONE)	NOT NULL
specie	VARCHAR(50)		NOT NULL
razza	VARCHAR(50)		

Nome Entità <b>SEGNALAZIONE_OGGETTO (Entità Figlia)</b>			
Nome campo	Tipo	Vincolo di chiave	Altri vincoli



id	BIGINT	PRIMARY KEY, FOREIGN KEY (ref. SEGNALAZIONE)	NOT NULL
id_drop_point	BIGINT	FOREIGN KEY(id: DROP_POINT)	
modalita_consegna	ENUM		Valori: 'DIRETTA', 'DROP_POINT'. NOT NULL
categoria	ENUM		Valori: 'DOCUMENTI', 'ELETTRONICA', 'PORTAFOGLI', 'CHIAVI', 'GIOIELLI', 'ALTRO'. NOT NULL

Nome Entità		RECLAMO	
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	BIGINT	PRIMARY KEY	AUTO-INCREMENT
id_utente_richiedente	BIGINT	FOREIGN KEY(ref. UTENTE_REGISTRA TO)	NOT NULL
id_segnalazione	BIGINT	FOREIGN KEY(ref. SEGNALAZIONE)	NOT NULL
risposta1	VARCHAR(255)		NOT NULL
risposta2	VARCHAR(255)		NOT NULL
stato	ENUM		Valori: 'IN_ATTESA', 'ACCETTATO', 'RIFIUTATO'.



			NOT NULL, DEFAULT ("IN_ATTESA")
codice_consegna	VARCHAR(6)		UNIQUE
data_richiesta	DATETIME		DEFAULT CURRENT_TIMESTAMP
data_deposito	DATETIME		
data Ritiro	DATETIME		
conferma_finder	BOOLEAN		DEFAULT FALSE
conferma_owner	BOOLEAN		DEFAULT FALSE

Nome Entità		DROP_POINT	
Nome campo	Tipo	Vincolo di chiave	Altri vincoli
id	BIGINT	PRIMARY KEY	AUTO INCREMENT
nome_attivita	VARCHAR(100)		NOT NULL
indirizzo	VARCHAR(100)		NOT NULL
citta	VARCHAR(50)		NOT NULL
latitudine	DOUBLE		
longitudine	DOUBLE		



ritiri_effettuati	INT		DEFAULT 0
provincia	VARCHAR(50)		NOT NULL
immagine	LONGBLOB		
stato	ENUM		Valori: 'IN_ATTESA', 'APPROVATO', 'RIFIUTATO'. DEFAULT 'IN_ATTESA', NOT NULL
orari_apertura	VARCHAR(255)		NOT NULL
email	VARCHAR(100)		NOT NULL UNIQUE
telefono	VARCHAR(20)		
password_hash	VARCHAR(255)		NOT NULL
immagine_content_type	VARCHAR(50)		



## Controllo degli accessi e sicurezza

Di seguito viene mostrata la matrice degli accessi per poter tenere traccia di quali attori possono accedere ai quali dei servizi offerti dal sistema.

Attori Oggetti	Utente registrato	Drop-Point	Admin	Ospite
Gestione Utente – Registrazione	-	-	-	RegistrazioneUtente, RegistrazioneDropPoint
Gestione Utente – Autenticazione	Login, Logout, VisualizzaProfilo, ModificaProfilo, CancellaAccount, RecuperoPassword	Login, Logout, VisualizzaProfilo, ModificaProfilo, CancellaAccount, RecuperoPassword	Login, Logout, VisualizzaProfilo, ModificaProfilo, CancellaAccount, RecuperoPassword	-
Gestione Segnalazioni (GS)	CreazioneSegnalazione, Ricerca/Filtrasegnalazioni, VisualizzaDettagliSegnalazione, VisualizzaBachecaSegnalazioni, CancellazioneSegnalazione	Ricerca/Filtrasegnalazioni, VisualizzaDettagliSegnalazione, VisualizzaBachecaSegnalazioni	Ricerca/Filtrasegnalazioni, VisualizzaDettagliSegnalazione, VisualizzaBachecaSegnalazioni	Ricerca/Filtrasegnalazioni, VisualizzaDettagliSegnalazione, VisualizzaBachecaSegnalazioni
Gestione Reclami (GR)	RichiestaSecureClaim, GestioneSecureClaim (accetta/rifiuta), AccessoRecapitiPerClaim, RestituzioneAnimaliDiretta, ModificaMetodoConsegna	-	-	-



Gestione Drop-Point (GDP)	VisualizzazioneDropPoint	<u>VisualizzazioneDropPoint</u> , <u>ConfermaDepositoConCodice</u> , <u>ConfermaRitiroConCodice</u> , <u>ModificaProfiloDropPoint</u>	ValidazioneDropPoint, GestioneDropPoint (approva/sospende/elimina)	RegistrazioneDropPoint, VisualizzazioneDropPoint
Gestione Amministrativa (GA)	-	-	GestioneUtenti (visualizza/elimina), GestioneSegnalazioni (visualizza/rimuove), GestioneDropPoint (visualizza/rimuove)	-
Community / Scoreboard (COM)	VisualizzaScoreboardUtenti	VisualizzaScoreboardUtenti	VisualizzaScoreboardUtenti	-

## Controllo globale del software

Foundly è una web-application interattiva: ogni funzionalità viene avviata da un'azione dell'utente sull'interfaccia grafica (browser). Ogni richiesta (es. click, invio di form) genera una chiamata HTTP che viene gestita dal controller, il quale inoltra la richiesta al sottosistema applicativo competente (Gestione Utente, Segnalazioni, Reclami, Drop-Point, Community). I servizi applicativi interagiscono con il livello di persistenza (DB) e restituiscono il risultato al controller, che aggiorna la vista da mostrare all'utente. Il controllo del flusso è quindi di tipo event-driven.



## Condizioni limite

Nel presente paragrafo verranno presentate le boundary conditions inerenti all'avvio del sistema, spegnimento del sistema, fallimento del sistema ed errore di accesso ai dati persistenti.

### Avvio del sistema

<b>Identificativo</b>	UC_SYS_1 – Avvio del Server e Deploy	<b>Data</b>	19/11/2025
		<b>Versione</b>	1.0
		<b>Autori</b>	Nappi Natale
<b>Descrizione</b>	Lo UC permette l'avvio del Web Server (Apache Tomcat) e l'inizializzazione dell'applicazione Foundly, rendendola accessibile via browser.		
<b>Attore principale</b>	Amministratore di sistema		
<b>Attori secondari</b>	Database (Aiven)		
<b>Entry condition</b>	La macchina server ospitante è accesa e il sistema operativo è funzionante. I servizi di rete sono attivi.		
<b>Exit condition</b>	Il server Tomcat è attivo, la connessione al Database è stabilita e la Web App risponde alle richieste HTTP/HTTPS.		
<b>On success</b>			
<b>Exit condition</b>	Il sistema non si avvia o l'applicazione non è raggiungibile (errore 503 oppure errore 404).		
<b>Flusso di eventi principale</b>			
1	Amministratore	Esegue lo script di avvio del Web Container	
2	Sistema	Carica il contesto dell'applicazione (file .war)	
3	Sistema	Tenta di stabilire la connessione JDBC con il Database remoto.	
4	Sistema	Se la connessione ha successo, inizializza i Controller e si mette in ascolto sulla porta configurata 8080, rendendo i servizi disponibili.	
<b>I Flusso di Eventi Alternativo: I Dati Persistenti sono danneggiati</b>			
3.a1	Sistema	Notifica l'Amministratore di problemi ai dati persistenti e non effettua l'avvio.	
3.a2	Sistema	Scrive l'errore nei file di log e interrompe il deploy dell'applicazione.	
3.a3	Amministratore	Verifica la configurazione di rete/credenziali e riavvia il servizio (torna al passo 1).	



## Spegnimento del sistema

<b>Identificativo</b>		UCBC_2 – Spegnimento del Sistema	<b>Data</b> <b>Versione</b> <b>Autori</b>	19/11/2025 1.0 Salvatore Lepore
<b>Descrizione</b>		Lo UC permette lo spegnimento del sistema		
<b>Attore principale</b>		Amministratore		
<b>Attori secondari</b>		NA		
<b>Entry condition</b>		L'Amministratore accede al Server AND Il Sistema è stato precedentemente avviato AND Il Sistema non è stato ancora spento		
<b>Exit condition</b>		Il sistema viene spento correttamente		
<b>On success</b>				
<b>Exit condition</b>		Il sistema non viene spento		
<b>On failure</b>				
<b>Flusso di eventi principale</b>				
1	Amministratore	Invia un comando/segnale di spegnimento al sistema Foundly sul server applicativo		
2	Sistema	Controlla che non ci siano connessioni ancora aperte da o verso l'esterno e, se non ci sono, termina l'esecuzione del sistema.		
<b>I Flusso di eventi alternativo: Ci sono connessioni ancora aperte</b>				
2.a1	Sistema	Notifica all'Amministratore che ci sono ancora connessioni aperte verso l'esterno.		
2.a2	Sistema	Attende per un intervallo di tempo predefinito per consentire il completamento delle richieste in corso, senza accettare nuove richieste se non per rispondere a quelle già avviate.		
2.a3	Sistema	Controlla nuovamente che non ci siano connessioni ancora aperte da o verso l'esterno e, se non ci sono, termina l'esecuzione del sistema Foundly.		
2.a4	Sistema	Notifica l'Amministratore dell'avvenuto spegnimento del sistema.		
<b>II Flusso di eventi alternativo: Ci sono connessioni ancora aperte</b>				
2.a3.a1	Sistema	Recide forzatamente le connessioni verso l'esterno ancora attive.		
2.a3.a2	Sistema	Notifica all'Amministratore l'avvenuto spegnimento del sistema Foundly e il numero di connessioni terminate forzatamente.		



## Fallimento del sistema

<b>Identificativo</b>	UCBC_3 – Fallimento del Sistema	<b>Data</b>	19/11/2025
		<b>Versione</b>	1.0
		<b>Autori</b>	Salvador Davide Passarelli
<b>Descrizione</b>	L'UC definisce il comportamento del Sistema in caso di fallimento.		
<b>Attore principale</b>	Amministratore		
<b>Attori secondari</b>	NA		
<b>Entry condition</b>	Il Sistema viene terminato inaspettatamente		
<b>Exit condition</b>	Il Sistema viene riavviato correttamente		
<b>On success</b>			
<b>Exit condition</b>	Il Sistema non viene riavviato		
<b>On failure</b>			
<b>Flusso di eventi principale</b>			
1	Amministratore	<b>Include UCBC_1</b>	

## Errore di Accesso ai Dati Persistenti

<b>Identificativo</b>	UCBC_4 – Errore di Accesso ai Dati Persistenti	<b>Data</b>	19/11/2025
		<b>Versione</b>	1.0
		<b>Autori</b>	Salvador Davide Passarelli
<b>Descrizione</b>	L'UC descrive il comportamento del sistema qualora fosse impossibile accedere ai dati persistenti o questi risultassero corrotti.		
<b>Attore principale</b>	Admin		
<b>Attori secondari</b>	NA		
<b>Entry condition</b>	Il Sistema non può accedere ai dati persistenti OR I dati persistenti risultano corrotti		
<b>Exit condition</b>	Il Sistema riprende il normale funzionamento		
<b>On success</b>			
<b>Exit condition</b>	Il Sistema non riprende il normale funzionamento		
<b>On failure</b>			
<b>Flusso di eventi principale</b>			
1	Sistema	Notifica l'amministratore dell'impossibilità di accedere ai dati persistenti	
2	Sistema	Cessa di processare eventuali richieste dall'esterno e risponde a tutte le richieste con un messaggio di errore.	
3	Amministratore	<b>Include UCBC_2</b>	
4	Amministratore	Ripristina l'accessibilità o la sanità dei dati persistenti.	
5	Amministratore	<b>Include UCBC_1</b>	



## 4 Servizi dei sottosistemi

In questa sezione vengono descritti i servizi offerti da ciascun sottosistema logico definito nell'architettura. Per ogni servizio viene indicata la descrizione funzionale e l'interfaccia logica di riferimento.

### Sottosistema Registrazione & Profilo

Servizio	Descrizione	Interfaccia
Registrazione utente	Gestisce l'iscrizione di un nuovo cittadino (Finder/Owner), validando e salvando i dati anagrafici e le credenziali.	RegistrazioneService
Registrazione Drop-Point	Gestisce la richiesta di iscrizione di un'attività commerciale, creando il profilo con stato iniziale "In Attesa".	RegistrazioneService
Visualizza profilo	Recupera i dati dell'utente autenticato, incluse le statistiche della Scoreboard e lo storico attività.	RegistrazioneService
Modifica Dati	Consente all'utente di aggiornare le proprie informazioni (es. telefono, indirizzo, foto) o cambiare la password.	RegistrazioneService
Cancellazione Account	Permette l'eliminazione definitiva dell'account e dei dati associati dal sistema.	RegistrazioneService



## Sottosistema Autenticazione & Autorizzazione

Servizio	Descrizione	Interfaccia
Login	Verifica le credenziali di accesso e avvia la sessione assegnando i permessi corretti (User, Admin, Drop-Point).	AutenticazioneService
Logout	Termina la sessione corrente dell'utente invalidando i token di accesso.	AutenticazioneService
Recupera password	Gestisce la procedura di reset della password tramite generazione e verifica di codice inviato via e-mail.	AutenticazioneService

## Sottosistema Segnalazioni (include Ricerca & Filtri)

Servizio	Descrizione	Interfaccia
Crea segnalazione	Permette al <i>Finder</i> di inserire una nuova segnalazione (oggetto o animale), caricare foto e definire le domande di verifica.	GestioneSegnalazioneService
Ricerca e filtri	Permette di recuperare la lista delle segnalazioni attive filtrando per categoria, luogo, data o parole chiave.	GestioneSegnalazioneService
Visualizza Dettaglio	Fornisce tutti i dati pubblici di una specifica segnalazione.	GestioneSegnalazioneService
Elimina Segnalazione	Permette al creatore (o all'Admin) di rimuovere una segnalazione dal sistema.	GestioneSegnalazioneService



## Sottosistema Secure Claim (Restituzione Diretta)

Servizio	Descrizione	Interfaccia
Invia reclamo	Permette all' <i>Owner</i> di rispondere alle domande di verifica e inviare una richiesta di restituzione (Secure Claim).	SecureClaimService
Valuta reclamo	Permette al <i>Finder</i> di visualizzare le risposte ricevute e decidere se accettare o rifiutare il reclamo.	SecureClaimService
Generazione Codice	Genera il codice univoco di consegna qualora il reclamo accettato preveda l'uso di un Drop-Point.	SecureClaimService
Conferma Restituzione Diretta	Gestisce il doppio handshake (conferma consegna e conferma ricezione) per gli scambi diretti tra utenti.	SecureClaimService



## Sottosistema Gestione Drop-Point

Servizio	Descrizione	Interfaccia
Ricerca Drop-Point	Recupera la lista dei Drop-Point approvati per mostrarli sulla mappa o in elenco durante la creazione della segnalazione o del reclamo.	GestioneDropPointService
Gestione Deposito	Permette all'operatore del Drop-Point di validare il <i>Codice di Consegnna</i> per prendere in custodia un oggetto (Check-in).	GestioneDropPointService
Gestione Ritiro	Permette all'operatore del Drop-Point di validare il <i>Codice di Consegnna</i> per restituire l'oggetto al legittimo proprietario (Check-out).	GestioneDropPointService

## Sottosistema Admin e Moderazione

Servizio	Descrizione	Interfaccia
Approva/Rifiuta Drop-Point	Permette all'Admin di valutare le richieste di registrazione dei Drop-Point pendenti.	AmministrazioneService
Gestione Utenti	Permette all'Admin di visualizzare la lista utenti e, se necessario, sospendere o eliminare account non conformi.	AmministrazioneService
Moderazione Segnalazioni	Permette all'Admin di rimuovere forzatamente segnalazioni che violano le policy della piattaforma.	AmministrazioneService



## 5 Design Pattern

Questa sezione descrive i principali Design Pattern adottati per l'implementazione del sistema **Foundly**. La scelta dei pattern è stata guidata dalla necessità di mantenere una chiara separazione delle responsabilità, facilitare la manutenibilità del codice e gestire la complessità delle operazioni di business (come l'invio di email e la gestione dei Drop-Point).

### 5.1. Pattern Strutturale: Facade

Pattern	Applicazione	Obiettivo
<b>Facade</b>	EmailService, SegnalazioneService, DropPointService, UtenteService, RecuperoPasswordService	Fornire un'interfaccia unica e semplificata per l'accesso a sottosistemi complessi o flussi articolati

#### 5.1.1. Contesto e Implementazione

Il sistema deve gestire operazioni complesse come l'invio di email e la gestione coordinata delle entità (es. accettazione di un reclamo che comporta aggiornamenti DB e notifiche). Abbiamo applicato il pattern **Facade** introducendo classi di servizio che nascondono questa complessità ai Controller (Servlet).

#### 5.1.2. Vantaggi

- **Disaccoppiamento:** Le Servlet non dipendono dalle classi interne dei sottosistemi.
- **Semplicità:** Il codice client (Servlet) è più leggibile e meno propenso a errori.



- **Architettura chiusa:** Per esempio la logica di invio email è encapsulata nel Facade, prevenendo usi impropri da parte del livello di presentazione

## 5.2. Pattern Comportamentale: Observer

Pattern	Applicazione	Obiettivo
<b>Observer</b>	Sistema di notifiche (EmailService)	Mantenere sincronizzati gli attori (utenti) al verificarsi di eventi nel sistema

### 5.2.1. Contesto e Implementazione

Quando lo stato di una Segnalazione o di un Reclamo cambia (agendo da Subject), è necessario notificare automaticamente gli utenti interessati (Observer).

- **Funzionamento:** Quando il SegnalazioneService esegue un cambio di stato (es. "Reclamo Accettato"), innesca automaticamente l'invio di una email tramite EmailService per avvisare il Finder e l'Owner dell'aggiornamento.
- Questo evita che gli utenti debbano controllare manualmente lo stato della pratica.

### 5.2.2. Vantaggi

- **Reattività:** Il sistema reagisce agli eventi di dominio notificando le parti interessate in tempo reale.
- **Disaccoppiamento:** La logica che cambia lo stato nel database è separata dai dettagli di come viene consegnata la notifica all'utente.



## 5.3. Pattern Creazionale: Abstract Factory

Pattern	Applicazione	Obiettivo
<b>Factory Method</b>	Creazione delle istanze di Segnalazione	Isolare la logica di creazione degli oggetti e gestire la gerarchia delle classi

### 5.3.1. Contesto e Implementazione

Il sistema Foundly gestisce diverse tipologie di segnalazioni (es. **SegnalazioneOggetto** e **SegnalazioneAnimale**) che condividono la struttura base astratta Segnalazione.

La logica di creazione è centralizzata nel livello di servizio: in base all'input dell'utente (il "tipo" selezionato nel form), il sistema decide a runtime quale sottoclassificazione specifica istanziare (es. new **SegnalazioneOggetto**(...) o new **SegnalazioneAnimale**(...)) prima di passarla al DAO.

### 5.3.2. Vantaggi

- **Estendibilità:** L'aggiunta di nuove tipologie di segnalazione in futuro non richiederà modifiche strutturali al codice che utilizza le segnalazioni in modo generico.
- **Coerenza:** La logica di istanziazione è gestita in un unico punto, garantendo la creazione di oggetti consistenti.



## 6 Glossario

Nella presente sezione sono raccolte le sigle o i termini del documento che necessitano di una definizione.

Sigla/Termino	Definizione
Foundly	Nome dell'applicativo web per la gestione collaborativa di oggetti e animali smarriti.
Schema ER	Schema Entità-Relazione. Rappresentazione concettuale della struttura dei dati, che definisce le entità e le relazioni reciproche.
SDD	Software Design Document. Documento che descrive l'architettura, la struttura dei dati, i componenti e l'interfaccia di un sistema software.
Condizione Limite	Valore o stato estremo (minimo, massimo, caso vuoto, ecc.) che il sistema deve saper gestire; viene utilizzato per valutare la robustezza del design e per la definizione dei casi di test.
Utente registrato	Persona fisica che interagisce con il sistema, autenticandosi e pubblicando segnalazioni o reclami.
Admin	Utente con permessi elevati, responsabile della gestione degli utenti, dell'approvazione dei Drop-Point e della moderazione delle segnalazioni.
Segnalazione	Registrazione di un oggetto o di un animale ritrovato (o associato a uno smarrimento), pubblicata sulla piattaforma con descrizione, dati di ritrovamento, immagini e metodo di restituzione.
Drop-Point	Punto fisico (attività commerciale) registrato e approvato come luogo sicuro per la custodia temporanea di oggetti ritrovati e per il loro ritiro da parte del legittimo proprietario.
Secure Claim	Procedura tramite la quale l'Owner rivendica la proprietà di una segnalazione rispondendo alle domande di verifica, generando (in caso di approvazione) un codice di consegna.



Domanda Verifica	Domanda definita dal Finder all'atto della creazione di una segnalazione, utilizzata per verificare la reale proprietà dell'oggetto o dell'animale durante la gestione del reclamo.
Codice Consegnna	Codice univoco generato dal sistema all'accettazione di un reclamo, utilizzato per tracciare le operazioni di deposito e ritiro presso un Drop-Point o per confermare la consegna diretta.
Modalità Consegnna	Modalità con cui avviene la restituzione dell'oggetto/animale, ad esempio consegna diretta tra utenti oppure tramite Drop-Point.
Stato Segnalazione	Indica lo stato attuale di una segnalazione. Nel sistema vengono utilizzati almeno gli stati: aperta (segnalazione attiva) e chiusa (segnalazione conclusa).
Scoreboard	Classifica pubblica che mostra il punteggio accumulato dagli utenti che hanno completato restituzioni con successo.
Finder/Owner	<i>Finder:</i> Utente che pubblica una segnalazione dopo aver trovato un oggetto/animale. <i>Owner:</i> Utente che invia un reclamo per rivendicare un oggetto/animale smarrito.