SAfeGuard

System Design Document

Riferimento: C08_SDD_1.0

Versione: 1.0

Data: 14 dicembre 2025

Destinatario: Prof.ssa Filomena Ferrucci, Prof.re Fabio Palomba

Presentato da: C08 - SAfeGuard

Documento generato il 14 dicembre 2025

Team Members

Nome	**Ruolo**	**Acronimo**	**Contatto**
Giuseppe Napolitano	Project Manager	GN	g.napolitano80@studenti.unisa.it
Pasquale Sorrentino	Project Manager	PS	p.sorrentino47@studenti.unisa.it
Alessandro Amendola	Team Member	AA	a.amendola51@studenti.unisa.it
Alessandro Masone	Team Member	AM	a.masone1@studenti.unisa.it
Antonello Castelluccio	Team Member	AC	a.castelluccio15@studenti.unisa.it
Francesco Carbone	Team Member	FC	f.carbone50@studenti.unisa.it
Francesco Zambrino	Team Member	FZ	f.zambrino@studenti.unisa.it
Gianpaolo Aquilone	Team Member	GA	g.aquilone1@studenti.unisa.it
Giorgio Zazzerini	Team Member	GZ	g.zazzerini@studenti.unisa.it
Giovanni Lamberti	Team Member	GL	g.lamberti55@studenti.unisa.it
Matteo Manganiello	Team Member	MM	m.manganiello15@studenti.unisa.it
Thomas Mercadino	Team Member	TM	t.mercandino@studenti.unisa.it
Victor Di Gennaro	Team Member	VDG	v.digennaro5@studenti.unisa.it

Revision History

Data	Ver.	Descrizione	Autori
17/11/25	0.1	Prima Stesura	AA, FC, GL, VDG
18/11/25	0.2	Design Goals e Trade-Offs	AA, FC, GL, VDG
19/11/25	0.2	Sottosistemi, Scelta architettura, Mapping hardware-software e Gestione dati persistenti	AM, AC, GA, TM, MM, FZ, GZ
20/11/25	0.3	Revisione	AA
13/12/25	1.0	Revisione finale	GL, AM

Indice

Capitolo 1

Introduzione

1.1 Obiettivo del Sistema

L'obiettivo primario del sistema SAfeGuard è sviluppare un applicativo mobile per ridurre il numero di feriti, decessi o dispersi in caso di calamità naturale. Il sistema mira ad aumentare la rapidità d'intervento e a ottimizzare il coordinamento dei soccorsi attraverso una piattaforma tecnologica unitaria e dinamica.

Il progetto si propone di creare un canale di comunicazione a doppio senso tra la popolazione e i soccorritori (come il VVF e la PC), sfruttando la geolocalizzazione, le notificazioni in tempo reale e moduli sperimentali di intelligenza artificiale.

Gli obiettivi principali includono:

- **Ridurre vittime e dispersi:** Consentire una rapida localizzazione degli utenti e una segnalazione immediata del loro stato ("sto bene / ho bisogno di aiuto").
- **Gestire i flussi di evacuazione:** Fornire indicazioni dinamiche sui percorsi più sicuri verso i punti di raccolta, riducendo il panico e il sovraffollamento.
- **Supportare i soccorsi:** Garantire ai centri operativi informazioni in tempo reale sulle zone maggiormente colpite e sulla posizione stimata dei dispersi, migliorando la prioritizzazione degli interventi.

1.2 Design Goals

Rank	Identificativo	Descrizione	Categoria	RNF Origine
1	DG_1 Sicurezza & Privacy	Il sistema deve garantire un elevato livello di sicurezza, proteggendo dati sensibili (posizione, dati sanitari opzionali), assicurando crittografia end-to-end nelle comunicazioni (SOS/notifiche) e piena conformità GDPR. L'accesso dei soccorritori deve essere autenticato e tracciato.	Affidabilità / Privacy	RNF-4.1, RNF-4.2, RNF-4.3, RNF-4.4

Rank	Identificativo	Descrizione	Categoria	RNF Origine
2	DG_2 Disponibilità del Sistema	Il backend deve garantire altissima disponibilità (uptime 99%) e ridondanza, assicurando che l'invio degli SOS e delle notifiche sia sempre possibile anche in condizioni critiche.	Affidabilità	RNF-1.1, RNF-1.2, RNF-1.3
3	DG_3 Performance in Emergenza	Il sistema deve rispondere molto rapidamente: invio SOS < 2s, aggiornamento mappa < 1s, notifiche massive entro finestre temporali ridotte. La UI deve mantenere fluidità anche in situazioni di carico elevato.	Performance	RNF-2.1, RNF-2.2, RNF-2.3
4	DG_4 Usabilità in Condizioni di Stress	L'interfaccia deve essere estremamente semplice, chiara e immediata, con pulsanti grandi e zero ambiguità. La modalità soccorritore deve essere utilizzabile all'aperto, sotto luce solare e con guanti.	End-user	RNF-3.1, RNF-3.2
5	DG_5 Accessibilità	Il sistema deve essere accessibile anche a utenti con disabilità (visiva, motoria, cognitiva) e supportare almeno Italiano e Inglese. Deve usare design ad alto contrasto e compatibile con screen reader.	End-user	RNF-3.1, RNF-3.3
6	DG_6 Modularità dell'Architettura	L'architettura deve essere modulare (idealmente a microservizi), separando gestione SOS, notifiche, geolocalizzazione, utenti, log e mappa, così da facilitare manutenzione e aggiornamenti futuri.	Manutenzione	RNF-1.3, RNF-5.2 (implicito)
7	DG_7 Compatibilità Multipiattaforma	L'app deve funzionare su un ampio numero di dispositivi e versioni (Android 10+, iOS 15+).	End-user	RNF-5.1, RNF-5.2
8	DG_8 Gestione Offline & Fallback	Il sistema deve funzionare anche senza connessione: SOS via SMS, caching locale dei dati critici e ritrasmissione automatica quando ritorna la connessione.	Affidabilità / Performance	RNF-1.1, RNF-1.2
9	DG_9 Tracciabilità e Audit degli Eventi	Il sistema deve registrare automaticamente e manualmente gli eventi critici (SOS, aggiornamenti stato, azioni operatore), permettendo esportazione e audit completi.	Affidabilità	RF-3.1, RF-3.2 + RNF-4.3
10	DG_10 Scalabilità	Il sistema deve essere in grado di gestire migliaia di utenti simultanei, picchi di notifiche massive e grandi volumi di log senza degradare prestazioni.	Performance	RNF-1.3, RNF-2.2, RNF-2.3

1.3 Design trade-offs

Trade-Off	**Razionale**
Performance vs. Consumo Batteria	Per garantire aggiornamenti rapidi di SOS e localizzazione (<2s), il sistema deve aumentare la frequenza dei controlli GPS durante un'emergenza. Ciò comporta un maggiore consumo energetico. È stata preferita la performance, essenziale per la sicurezza dell'utente.
Privacy by Design vs. Precisione della Localizzazione	La protezione dei dati richiede minimizzazione della posizione in tempo di pace, ma i soccorritori necessitano di posizione precisa durante un SOS. Si è adottato un modello ibrido: posizione approssimativa normalmente, precisa solo in emergenza.
Modello 100% Online vs. Supporto Offline (SMS Fallback)	Un sistema unicamente online sarebbe più semplice da sviluppare, ma non affidabile durante alluvioni, blackout o congestione della rete. Si è scelto di integrare un fallback SMS automatico, accettando maggiore complessità per garantire disponibilità totale.
Modularità (microservizi) vs. Overhead di Manutenzione	I microservizi offrono scalabilità e separazione chiara delle funzionalità (SOS, notifiche, mappe, log), ma richiedono infrastrutture più complesse. È stata privilegiata la modularità poiché il sistema è critico e deve scalare rapidamente durante emergenze.
Usabilità estrema vs. Numero di funzionalità	Aggiungere molte funzioni aumenterebbe la complessità cognitiva in situazioni di panico. È stato scelto un design minimale nelle interazioni critiche (SOS, Sto Bene, Mappa), relegando funzioni avanzate a sezioni secondarie.
Aggiornamento Real-Time della Mappa vs. Carico di Rete	Aggiornare la mappa del soccorritore ogni secondo garantisce operatività immediata, ma aumenta il carico di rete e risorse. Si è privilegiata la reattività, fondamentale per coordinare i soccorsi.
Logging Completo vs. Spazio e Privacy	Registrare molti eventi supporta analisi post-intervento e audit, ma aumenta spazio occupato e rischio di raccogliere dati superflui. Si è scelta una registrazione completa ma anonimizzabile, bilanciando audit e tutela privacy.
Compatibilità Multipiattaforma VS. Ottimizzazione Hardware	Supportare un'ampia varietà di dispositivi riduce l'uso di tecnologie hardware specifiche. La scelta finale favorisce la compatibilità per i cittadini.

1.4 Definizioni, Acronimi e Abbreviazioni

Di seguito, una lista di definizioni, acronimi e abbreviazioni:

Acronimo	Descrizione
RF	Requisiti Funzionali
RNF	Requisiti Non Funzionali
VVF	Corpo Nazionale dei Vigili del Fuoco
SOS	Segnalazione di Richiesta di Soccorso
PC	Protezione Civile
GPS	Global Positioning System
DG	Design Goal
GIS	Geographic Information System
URI	Uniform Resource Identifier
UC	Use Case
RAD	Requirements Analysis Document
UI	User Interface
SDD	System Design Document
GDPR	General Data Protection Regulation
DBMS	Database Management System
PDF	Portable Document Format
CSV	Comma-Separated Values
NoSql	Not Only Sql
BaaS	Backend-as-a-service
SMS	Short Message Service

1.5 Riferimenti

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

- Statement Of Work
- Object Design Model
- Requirements Analysis Document
- Matrice di tracciabilità

1.6 Organizzazione del documento

Il seguente documento di System Design consta di due sezioni:

- **1. Introduzione:** ha lo scopo di fornire una panoramica completa dell'obiettivo che il sistema si propone di raggiungere, i principi di progettazione (design goals) e le scelte strategiche che hanno influito sul processo di sviluppo, evidenziando i principali design trade-off.
- **2. Sistema Proposto:** Descrive come il sistema sarà definito, comprende la decomposizione in sottosistemi, una descrizione dell'architettura del sistema, e l'hardware-software mapping. Seguono l'individuazione dei microservizi e la gestione dei dati persistenti, dove vengono descritte le modalità di archiviazione e gestione dei dati nel lungo periodo.

Capitolo 2

Sistema Proposto

2.1 Decomposizione in sottosistemi

I sottosistemi individuati per il progetto SAfeGuard sono i seguenti:

1 - Gestione Account (Utente)

Responsabilità: Si occupa della registrazione, autenticazione e gestione del ciclo di vita degli account.
Funzionalità Chiave:

- Gestisce la distinzione tra Utente Standard e Soccorritore.
- Include la gestione e la modifica delle informazioni del profilo, inclusi i dati volontari di accessibilità e sanitari.
- Gestisce l'eliminazione sicura dell'account.

2 - Emergenza (App Utente)

Responsabilità: Gestisce le interazioni e le azioni del "lato cittadino" durante un'emergenza.
Funzionalità Chiave:

- È responsabile dell'invio della "Richiesta di Aiuto" (SOS) e della segnalazione "Sto Bene", inclusa la logica di doppia conferma.
- Riceve le notifiche di emergenza "massive" inviate dal sottosistema Notifiche.
- Consuma i servizi del sottosistema Mappe e GIS per mostrare all'utente la mappa con i percorsi di fuga e le aree sicure.

3 - Monitoraggio (Dashboard Soccorritore)

Responsabilità: Rappresenta il pannello operativo per i soccorritori.
Funzionalità Chiave:

- Consuma i servizi del sottosistema Mappe e GIS per visualizzare la Mappa Operativa (GIS) in tempo reale.

- Mostra la posizione delle richieste di aiuto, lo stato degli utenti e i layer di sicurezza.
- Consuma i servizi del sottosistema Log e Reporting per permettere l'analisi e l'esportazione dei dati.

4 - Mappe e GIS

Responsabilità: Sottosistema centrale che gestisce, aggrega e fornisce tutti i dati e i servizi geospaziali.
Funzionalità Chiave:

- Fornisce le API e i layer di base per la mappa (strade, satellite).
- Gestisce l'archivio dei Punti Sicuri e delle aree di raccolta.
- Calcola e fornisce i percorsi di fuga dinamici per gli utenti.
- Aggrega i dati (SOS, "Sto Bene") sulla Mappa Operativa per i soccorritori.

5 - Geolocalizzazione e Privacy

Responsabilità: Sottosistema critico, prevalentemente lato client (app), che gestisce l'accesso alla posizione e la privacy.
Funzionalità Chiave:

- Implementa la logica di Privacy by Design, attivando la geolocalizzazione precisa solo in caso di SOS o allerta ufficiale.
- Gestisce i consensi e i permessi del dispositivo.
- È responsabile della logica di attivazione del fallback: rileva la connessione assente e innesca l'invio tramite SMS.

6 - Notifiche

Responsabilità: Gestisce l'infrastruttura di comunicazione (push e SMS) tra il sistema e gli utenti.
Funzionalità Chiave:

- È responsabile dell'invio delle notifiche "massive" a migliaia di utenti in un'area definita.
- Gestisce la coda e l'invio delle conferme di ricezione/invio.
- Funge da gateway per l'invio effettivo dell'SMS geolocalizzato (innescato dal sottosistema Geolocalizzazione).

7 - Log e Reporting

Responsabilità: Si occupa dell'archiviazione, della gestione e dell'esportazione dei log di intervento.
Funzionalità Chiave:

- Registra automaticamente tutti gli eventi (SOS ricevuti, notifiche inviate, variazioni di stato) in modo asincrono.

- Permette ai soccorritori (tramite il Monitoraggio) di esportare i dati per analisi post-intervento.

8 - Analisi (IA)

Responsabilità: Modulo sperimentale responsabile dell'analisi dei dati per fornire insight ai soccorritori.
Funzionalità Chiave:

- Analizza la densità e il tipo di segnalazioni per suddividere le zone colpite in macroaree (cluster).
- Fornisce i "Layer Evento - IA" che vengono poi visualizzati dal sottosistema Mappe e GIS.

9 - Persistenza

Responsabilità: Gestisce l'accesso e la memorizzazione di tutti i dati persistenti del sistema.
Funzionalità Chiave:

- Centralizza l'accesso al database (o ai database) per le entità principali.
- Garantisce l'integrità e la sicurezza dei dati.

2.1.1 Component Diagram

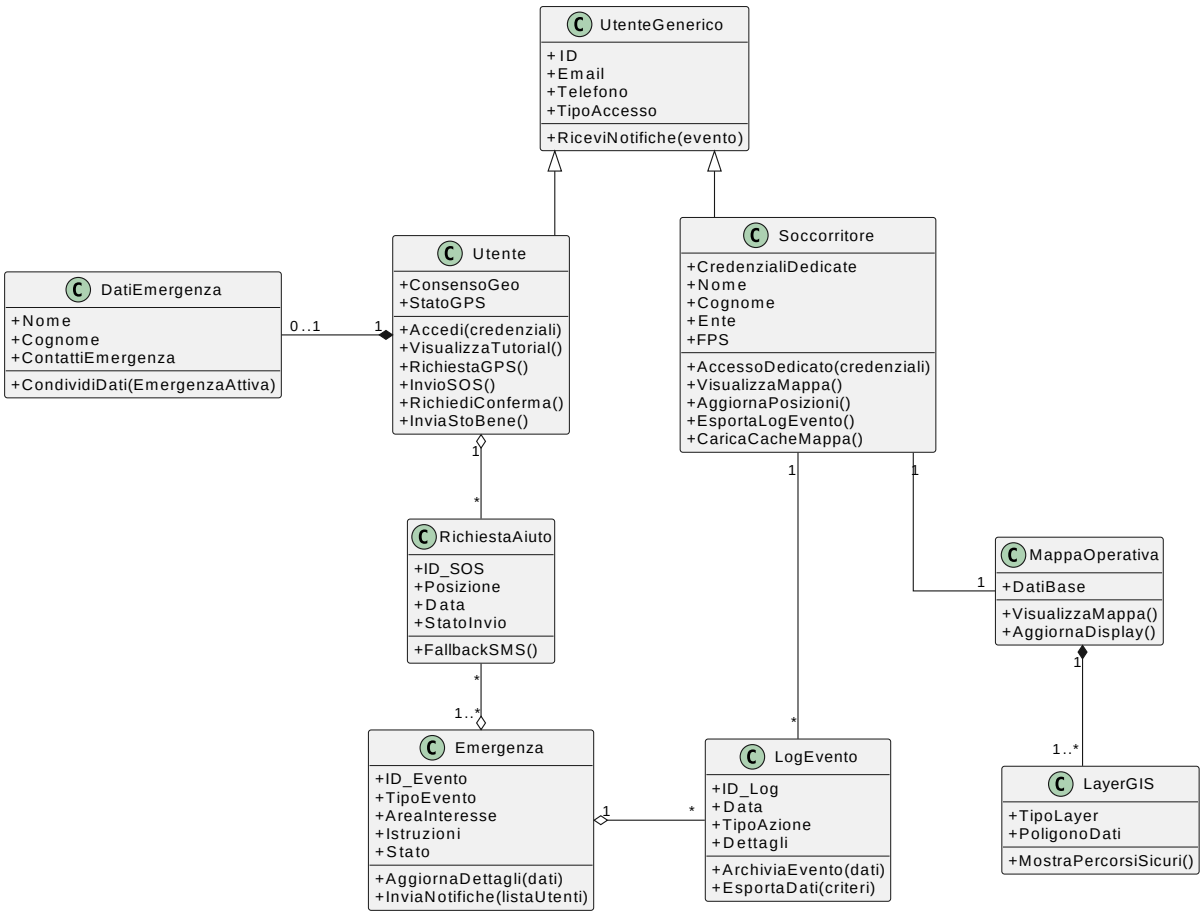

Figura 2.1: Component Diagram

2.2 Architettura del sistema

L'architettura scelta per il sistema SAfeGuard è un'architettura a microservizi, poiché risponde efficacemente ai requisiti critici di affidabilità (RNF-1.3), disponibilità e prestazioni definiti nel RAD. Questa scelta strategica garantisce che il fallimento di un singolo modulo non comprometta l'intero sistema, aspetto vitale per una piattaforma di gestione delle emergenze.

La scelta è motivata dalla necessità di isolare i processi critici, come la ricezione delle richieste di soccorso (SOS) e l'invio delle notifiche massive, dalle funzionalità accessorie. Un'architettura a microservizi permette di scalare orizzontalmente le singole componenti in risposta a picchi di carico improvvisi (es. durante una calamità naturale), assicurando tempi di risposta minimi (RNF-2.1) e la gestione efficiente di grandi volumi di dati geolocalizzati.

L'architettura verrà implementata avvalendosi delle seguenti tecnologie:

Per il front-end abbiamo scelto Flutter (basato su Dart) per una ragione strategica fondamentale: la produttività cross-platform. Essendo un team compatto, non possiamo permetterci di scrivere due codici separati (Kotlin per Android e Swift per iOS). Flutter ci permette di mantenere un unico codice sorgente che compila nativamente su entrambi i sistemi operativi senza sacrificare le prestazioni, grazie al suo motore di rendering grafico proprietario.

Per il back-end abbiamo deciso di utilizzare Dart per creare un ecosistema "Full-stack Dart". Questa scelta elimina la barriera linguistica tra Front-end e Back-end: tutto il team parla la stessa lingua. Il vantaggio tecnico enorme è la condivisione del codice: possiamo scrivere le classi dei dati una volta sola e usarle sia nel server che nell'app mobile, evitando errori di conversione e disallineamenti. Se il team back-end ha bisogno di supporto, il team front-end può leggere e comprendere il suo codice immediatamente, rendendo il team flessibile e interscambiabile.

Per il database abbiamo scelto Firebase (di Google) perché offre l'integrazione più naturale e matura esistente con Flutter. Non è solo un database, ma una piattaforma "BaaS" che ci toglie l'onere di gestire e configurare server fisici complessi, permettendoci di concentrarci sulle funzionalità dell'app. In particolare, abbiamo scelto Firebase per le sue capacità Real-time: ogni modifica ai dati viene spinta istantaneamente a tutti i dispositivi connessi senza bisogno di ricaricare la pagina, una caratteristica cruciale per un'esperienza utente moderna e reattiva.

2.3 Hardware-software mapping

Il sistema sviluppato si basa su un applicativo mobile multipiattaforma (Android/iOS), progettata per garantire la compatibilità con la maggior parte dei dispositivi. Ciò permette una manutenibilità del codice migliorata e uno sviluppo ottimizzato a fronte di un unico codice sorgente. L'interfaccia grafica è realizzata sfruttando moderne tecnologie, nello specifico Flutter con il linguaggio Dart, per permettere un'esperienza utente fluida. Ogni operazione viene gestita da servizi di backend (microservizi) messi a disposizione dal sistema sfruttando la medesima tecnologia.

La persistenza dei dati viene affidata ad un database di tipo non relazionale, ovvero Firebase. Garantisce riservatezza delle informazioni utente, sia del cittadino che del soccorritore conforme con il GDPR (RNF-4.1). Inoltre, archivia e permette di accedere allo storico degli eventi, utile per le analisi post-intervento (UC_10). Inoltre, secondo il RNF-1.1 il software funziona anche in modalità offline, ciò è possibile grazie alla logica inserita per l'assenza di connettività, ovvero il servizio SMS del dispositivo.

2.3.1 Deployment Diagram

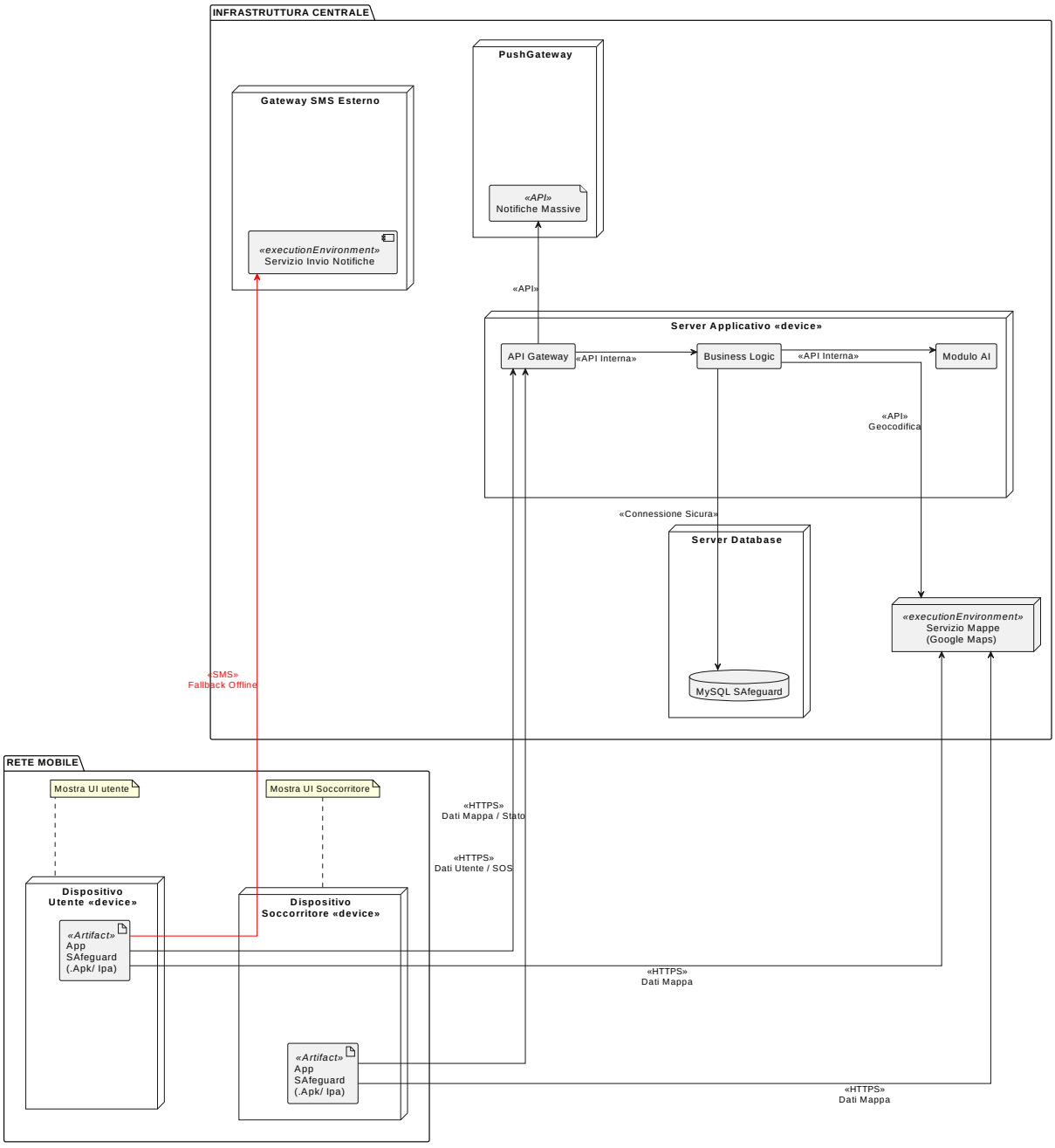

Figura 2.2: Deployment Diagram

2.4 Individuazione microservizi

In base ai sottosistemi sono stati trovati i seguenti microservizi:

Sottosistema Account

Servizio	Parametro	Descrizione
Registrazione Google	GoogleID, nome, URL (del profilo), e-mail.	Questa funzionalità permette la registrazione al sistema, come Cittadino, tramite il proprio account Google.
Registrazione Apple ID	e-mail, nome, cognome	Questa funzionalità permette la registrazione al sistema, come Cittadino, tramite il proprio account Apple.
Registrazione E-mail	E-mail, password.	Questa funzionalità permette ad un utente di creare un account di tipo Cittadino (se la verifica andrà a buon fine). Se l'e-mail inserita è di tipo istituzionale verrà creata un account di tipo Soccorritore (se la verifica andrà a buon fine).
Login Google	GoogleID, nome, URL (del profilo), e-mail.	Questa funzionalità permette l'accesso al sistema, come Cittadino, tramite il proprio account Google.
Login Apple ID	e-mail, nome, cognome	Questa funzionalità permette l'accesso al sistema, come Cittadino, tramite il proprio account Apple.
Login E-mail	E-mail, password.	Questa funzionalità permette ad un utente di accedere al proprio account inserendo e-mail e password.
Invio codice di verifica su e-mail	E-mail, Codice Generato	Questa funzionalità permette la verifica dell'e-mail inserita dall'utente. Se l'e-mail inserita a tempo di login è istituzionale e la verifica va a buon fine, viene creato un account di tipo Soccorritore.
Invio codice di verifica al numero di telefono	Numero Telefono, Codice Generato.	Questa funzionalità permette la verifica del numero di telefono inserito dall'utente. Se la verifica è andata a buon fine, viene creato l'account.
Eliminazione del profilo	E-mail, Numero Telefono, nome, cognome, allergie, medicinali, Emergenze Segnalate, disabilità.	Questa funzionalità permette di eliminare l'account registrato e tutte le informazioni annesse. Può essere eseguita sia da un cittadino che da un soccorritore.

Sottosistema Mappa e Monitoraggio

Servizio	Parametro	Descrizione
Visualizzazione mappa	PosX, PosY, PosZ.	Questa funzionalità permette al cittadino di visualizzare la mappa e annesse emergenze nella zona o più zone colpite.
Condivisione dati di posizione	Posizione.	Questa funzionalità permette la condivisione della posizione delle emergenze ai soccorritori e i soccorritori ai cittadini.
Visualizzazione stato emergenze	Stato ("StoBene" o "NonStoBene"), posizione	Questa funzionalità permette ai soccorritori di vedere le posizioni delle emergenze segnalate. Se lo stato è "StoBene" allora la posizione della segnalazione comparirà verde. Se lo stato è "NonStoBene" allora la posizione della segnalazione comparirà rossa.
Visualizzazione punti di raccolta	Posizione PuntiRaccolta	Questa funzionalità permette all'utente di visualizzare i punti di raccolta più vicini in base all'emergenza segnalata.

Sottosistema Notifiche

Servizio	Parametro	Descrizione
Notifiche di emergenza	Permesso Notifiche ("Si" o "No"), tipologia (es. Terremoto), posizione, descrizione (opzionale), ora, data.	Questa funzionalità (se Permesso Notifiche è "Si") permette dal cittadino, che ha dato consenso alle notifiche, di ricevere notifiche in base alle emergenze.
Notifiche delle segnalazioni	Permesso Notifiche ("Si" o "No"), nome, cognome, tipologia (es. Terremoto).	Questa funzionalità (se Permesso Notifiche è "No") permette al soccorritore di ricevere notifiche delle emergenze che vengono segnalate dai cittadini.

Sottosistema Emergenza

Servizio	Parametro	Descrizione
Creazione emergenza	Posizione, nome, cognome, data, ora, disabilità, allergie, medicinali.	Permette ad un cittadino o ad un soccorritore di segnalare un'emergenza, rispettivamente ad un soccorritore e ai cittadini, in modo veloce.
Creazione emergenza specifica	Posizione, nome, cognome, data, ora, disabilità, allergie, medicinali, tipologia (es. Terremoto), descrizione, aiuto ("Si" o "No").	Permette ad un cittadino o ad un soccorritore di segnalare un'emergenza, rispettivamente ad un soccorritore e ai cittadini, un'emergenza con una descrizione e in maniera più specifica.
Visualizzazione emergenze	Posizione, nome, cognome, data, ora.	Permette ad un soccorritore di visualizzare le emergenze segnalate dai cittadini.

Sottosistema Log e Reporting

Servizio	Parametro	Descrizione
Creazione LOG automatico	Posizione, nome, cognome, data, ora, disabilità, allergie, medicinali, tipologia (es. Terremoto), descrizione, stato ("StoBene" o "NonStoBene").	Permette ad un soccorritore di visualizzare le emergenze segnalate dai cittadini.
Esportazione LOG	Posizione, nome, cognome, data, ora, invalidità, allergie, medicinali, tipologia (es. Terremoto), descrizione, aiuto ("Si" o "No").	Permette ad un soccorritore di esportare il log.

2.5 Gestione dati persistenti

Il sistema richiede un'archiviazione significativa per supportare un numero elevato di utenti (utenti e soccorritori), eventi di emergenza, log dettagliati delle operazioni e dati geospaziali complessi. Questi dati, destinati a crescere esponenzialmente durante le fasi di crisi, sono caratterizzati da accessi concorrenti massivi e necessitano di tempi di risposta immediati.

Alla luce di queste esigenze, si è scelto di adottare una soluzione NoSQL basata su Firebase (Cloud Firestore), che funge da backend-as-a-service (BaaS) per l'ecosistema applicativo sviluppato interamente in Flutter. Questa scelta strategica permette di superare i limiti di scalabilità

dei database relazionali tradizionali, delegando la complessità infrastrutturale al cloud di Google. L'interazione con il database avviene direttamente tramite l'SDK Firebase per Flutter, che abilita un'architettura reattiva basata su Stream: questo approccio garantisce una sincronizzazione istantanea dei dati tra i dispositivi e il cloud, offrendo tempi di risposta immediati e supportando nativamente la modalità offline per operare anche in assenza di connettività stabile.

L'applicazione sviluppata in Flutter gestisce autonomamente la logica applicativa e di presentazione direttamente sui dispositivi, mentre Firebase garantisce la sincronizzazione dei dati in tempo reale (Real-time listeners). Questo approccio diretto assicura che, in situazioni di carico elevato o instabilità di rete (RNF-1), i client (app utenti e dashboard soccorritori) rimangano sempre reattivi, ricevendo aggiornamenti istantanei senza necessità di effettuare polling continuo e sfruttando la cache locale per operare anche offline.

Il DBMS selezionato (Firestore) fornisce, in linea con i design goals, strumenti avanzati per lo sviluppo del sistema:

- **Sincronizzazione in Tempo Reale (Real-time Sync):** A differenza dei database tradizionali, Firebase invia automaticamente le modifiche ai client connessi (push). Questo è cruciale per le emergenze: la posizione di un soccorritore o lo stato di un SOS si aggiornano istantaneamente sulle mappe operative.

- **Affidabilità e Scalabilità Serverless:** Essendo una soluzione completamente gestita (serverless) su infrastruttura Google, il sistema scala automaticamente orizzontalmente per gestire picchi improvvisi di traffico (es. durante una catastrofe), garantendo un uptime elevato (99%) e replica globale dei dati.

- **Sicurezza e Privatezza:** L'accesso ai dati sensibili è protetto dalle Firebase Security Rules e dall'integrazione con Firebase Authentication. I dati sono cifrati a riposo e in transito, e l'accesso granulare ai documenti assicura la piena conformità con il GDPR (RNF-4).

Gestione dei Dati Geospaziali e File di Log

I file pesanti, come i report esportati dei log (PDF/CSV) o i layer cartografici statici, saranno gestiti tramite Firebase Cloud Storage. Questa soluzione di object storage disaccoppia i file binari dal database strutturato: nel database (Firestore) verrà memorizzato solo il riferimento (URI sicuro) alla risorsa.

Per quanto riguarda i dati vettoriali necessari per il calcolo in tempo reale (coordinate utenti), Firestore utilizza il tipo di dato nativo GeoPoint (latitudine/longitudine). Sebbene l'applicazione Flutter possa eseguire calcoli geometrici complessi direttamente lato client (sfruttando librerie Dart per calcolare l'intersezione con poligoni di pericolo), la memorizzazione nativa su Firebase permette query spaziali rapide e l'aggiornamento fluido delle posizioni sulla mappa.

Denormalizzazione e Ottimizzazione (Design NoSQL)

Adottando un approccio NoSQL, la ridondanza dei dati non è solo tollerata, ma è una best practice di design (denormalizzazione) per ottimizzare la lettura.

- **Stato e Posizione:** Lo Stato Corrente di un utente (es. "In Pericolo") e la sua Ultima Posizione vengono mantenuti come attributi aggiornati direttamente nel documento Utente o Segnalazione, evitando di dover ricostruire lo stato leggendo l'intero storico dei log ad ogni richiesta.

- **Aggregati Real-time:** Dati come il Numero Totale di Richieste Attive per una zona non vengono calcolati con query di conteggio (che su grandi dataset sono lente e costose), ma vengono mantenuti aggiornati tramite Cloud Functions o logica backend che incrementa/decrementa contatori distribuiti o documenti di riepilogo.

I benefici di questa struttura superano i costi di spazio aggiuntivo. La lettura diretta dello stato attuale consente un aggiornamento della Dashboard Soccorritore (RNF-2.3) in millisecondi, migliorando drasticamente la reattività operativa. Questo approccio assicura che il sistema rimanga performante anche con milioni di documenti attivi.

2.5.1 Class Diagram Ristrutturato

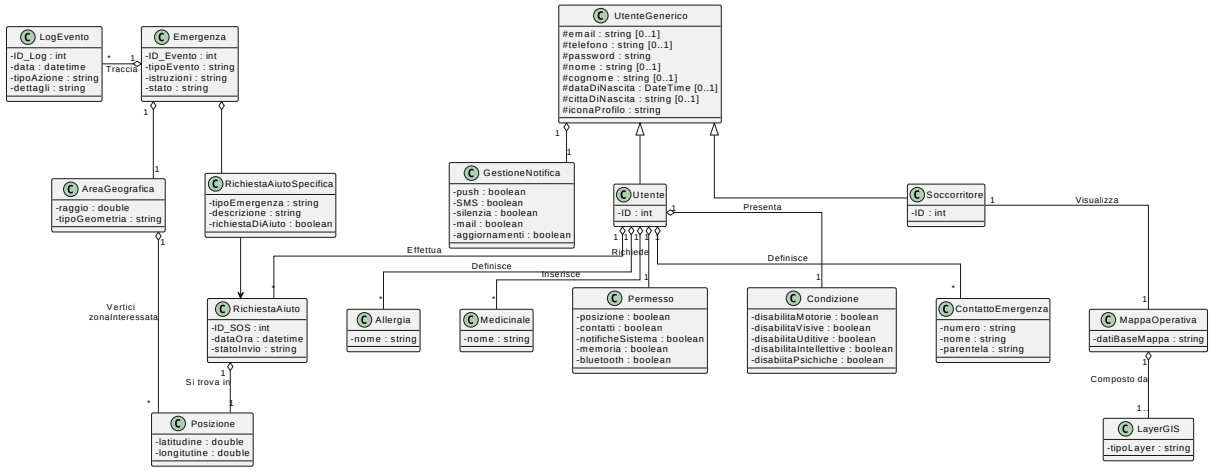

Figura 2.3: Class Diagram Ristrutturato