Laurea Magistrale in informatica-Università di Salerno
Corso di *Gestione dei Progetti Software*- Prof.ssa F.Ferrucci

System Design Document

RoadGuardian

Riferimento	C07_SDD
Versione	1.0.0
Data	19/11/2025
Destinatario	Docenti di Ingegneria del Software
Presentato da	C07-Angela Setola, Simone Domenico Avitabile, Mattia D'Auria, Raffaele Cimino, Giovanna Massa, Ciro Navarra, Lorenzo Olivola, Davide Pio Lazzarini, Sabato Iaquino

Revision History

Data	Versione	Descrizione	Autori
17/11/2025	0.0.1	Introduzione	Giovanna Massa, Lorenzo Olivola, Raffaele Cimino, Simone Domenico Avitabile, Angela Setola
17/11/2025	0.1.0	Current architecture	Raffaele Cimino, Sabato Iaquino, Mattia D'Auria
18/11/2025	0.2.0	Overview e System Decomposition	Raffaele Cimino, Lorenzo Olivola, Simone Domenico Avitabile
19/11/2025	0.2.1	Aggiunta Boundary Condition	Raffaele Cimino, Sabato Iaquino, Mattia D'Auria
19/11/25	0.3.0	Global Software Control	Simone Domenico Avitabile, Giovanna Massa, Angela Setola
19/11/2025	0.4.0	Mapping Hardware/Software e Persistent Data Management	Lorenzo Olivola, Ciro Navarra, Davide Pio Lazzarini
22/11/2025	0.4.1	Modifica del diagramma sezione 3.2.2.2, modifica dell'introduzione, aggiunti sottoelenchi in: 3.2 System Decomposition	Simone Domenico Avitabile
23/11/2025	0.4.2	Modifica descrizione DG_07	Davide Pio Lazzarini

24/11/2025	0.4.3	Modifica del trade-off Performance vs Dependability e raffinamento degli Use Case riguardanti le Boundary Conditions	Raffaele Cimino
25/11/2025	0.4.4	Eliminazione Boundary Conditions	Raffaele Cimino, Lorenzo Olivola, Giovanna Massa
29/11/2025	0.4.5	Correzione DG_05 e DG_09	Davide Pio Lazzarini, Carlo Mancusi
01/12/2025	0.4.6	Sostituzione del Deployment Diagram	Lorenzo Olivola
10/12/2025	0.4.7	Modifica e rinominazione del DG_06	Giovanna Massa
14/12/2025	1.0.0	Revisione finale	Tutti i TM

Indice

1. Introduzione

- Scopo del sistema
- Design Goals
- Trade-offs
- Definizioni, acronimi e abbreviazioni
- Riferimenti
- Overview

1.1. Scopo del Sistema

L'obiettivo del progetto è la realizzazione di un'app mobile pensata per aumentare la sicurezza stradale. L'applicazione consentirà agli utenti di inviare rapidamente una segnalazione, includendo dettagli fondamentali come la posizione GPS, la tipologia d'incidente e, opzionalmente, una descrizione dell'accaduto. L'applicazione integra funzionalità aggiuntive, tra cui notifiche push per avvisare gli automobilisti che sono vicini alla zona dell'incidente che includono linee guida pratiche per il corretto comportamento da avere durante una specifica situazione d'emergenza.

1.2. Design Goals

I Design Goals scelti sono divisi nelle seguenti categorie:

- Performance
- Dependability
- End user
- Maintenance

ID	Nome	Descrizione	Categoria	RNF di origine	Priorità
DG_01	Formato dati registrazione	L'applicazione controlla la correttezza dei dati inseriti per evitare input non validi e quindi resistere ad attacchi che minano la robustezza	Dependability	RNF_01	4

ID	Nome	Descrizione	Categoria	RNF di origine	Priorità
		dello stesso.			
DG_02	Consenso alla posizione in tempo reale	L'applicazione deve chiedere il consenso esplicito all'utente per accedere alla posizione in tempo reale del dispositivo (specificando il suo consenso al primo accesso all'app), così da permettere la corretta visualizzazione degli incidenti nelle vicinanze e il corretto uso del sistema.	Dependability	RNF_04	5
DG_03	Password hashing	L'applicazione protegge le password degli utenti mediante funzione di hashing, in modo da garantire che non possano essere	Dependability	RNF_05	2

ID	Nome	Descrizione	Categoria	RNF di origine	Priorità
		recuperate in chiaro e ridurre il rischio di accessi non autorizzati.			
DG_04	Documentazione degli artefatti	L'applicazione deve garantire che il codice e i vari artefatti siano documentati in modo da permettere una semplice manutenzione e aggiornamento.	Maintenance	RNF_07	6
DG_05	Mobile First	L'accesso al servizio dovrà avvenire tramite l'implementazione di un'app mobile.	End user	RNF_09	1
DG_06	Notifiche tempestive di prossimità	L'applicazione deve inviare una notifica agli utenti nel raggio di 3km dall'invio della segnalazione	Performance	RNF_09	3

ID	Nome	Descrizione	Categoria	RNF di origine	Priorità
		di incidente.			
DG_07	Usabilità dell'interfaccia	L'applicazione consente anche agli utenti inesperti di inviare e consultare rapidamente segnalazioni e dettagli, senza necessità di formazione.	End user	RNF_02	7
DG_08	UI responsiveness	L'applicazione deve adattare automaticamente il layout e i contenuti alle diverse dimensioni e orientamenti dello schermo, garantendo una visualizzazione corretta e una navigazione fluida su ogni dispositivo.	End user	RNF_03	8
DG_09	Evoluzione	L'architettura del sistema deve essere progettata per supportare	Maintenance	RNF_08	9

ID	Nome	Descrizione	Categoria	RNF di origine	Priorità
		un'evoluzione agevole, assicurando l'implementazione delle funzionalità residue attraverso un piano di rilasci predefinito.			

1.2.1. Trade-offs

Trade-off	Descrizione
Performance vs Dependability	Per assicurare il tempo di risposta di 30 secondi per l'invio delle notifiche di incidente, il sistema deve dare priorità al calcolo della vicinanza degli utenti dall'incidente e all'invio delle notifiche. Questo può portare a compromessi nella verifica rigorosa del consenso alla posizione in tempo reale. Potrebbe essere necessario quindi basarsi su dati di posizione meno precisi (riducendo la latenza di verifica), con la possibilità che gli utenti non ricevano quindi notifiche degli incidenti con la massima precisione in termini di distanza dallo stesso.
End user vs Dependability	Per garantire la sicurezza dei dati e robustezza del sistema, ci sarà una diminuzione della fluidità dello stesso. Ad esempio, attraverso controlli di formato, uso di password complesse e hashing delle

Trade-off	Descrizione
	password.

1.2.2. Definizioni, Acronimi e Abbreviazioni

Acronimo	Definizione
SDD	System Design Document
RAD	Requirements Analysis Document
SOW	Statement of Work
GPS	Global Positioning System
RNF	Requisito non Funzionale
DG	Design Goals
DB	Database
FCM	Firebase Cloud Messaging

1.3. Riferimenti

Documentazioni ed altro:

- 2025_C07_SOW;
- 2025_C07_RAD;
- Funzioni di altre app come Waze o simili per la segnalazione di incidenti;
- Slide del corso di Ingegneria del Software.

1.4. Overview

Il documento è organizzato come segue:

- Il Capitolo 2 descrive lo stato attuale dell'architettura;
- Il Capitolo 3 descrive la suddivisione del sistema in sottosistemi, il mapping hardware/software e della gestione dei dati persistenti.

2. Current architecture

Delle applicazioni presenti in commercio, Waze offre la possibilità di segnalare incidenti ma non permette di ricevere una notifica push in prossimità di una segnalazione d'incidente, ciò che il nostro sistema fornirà, e non permette di effettuare segnalazioni dettagliate o in maniera veloce. Inoltre, in nessuno dei sistemi in commercio, analizzati anche nel **Current System** del RAD, è presente la funzionalità di fornire linee guida comportamentali adattate in base al tipo di incidente. Quindi, tra le applicazioni in commercio, non ne è presente alcuna che implementi l'insieme delle funzionalità offerte da **RoadGuardian**.

3. Proposed architecture

3.1 Overview

Per la nostra applicazione la scelta adottata è un'architettura Hybrid Fat Client, una variante evoluta dell'architettura Client-Server che combina i vantaggi del processing locale con la reattività degli aggiornamenti in tempo reale. Architettura particolarmente adatta per un'applicazione mobile di segnalazioni che richiede operazioni rapide, gestione locale della logica applicativa e notifiche push istantanee per eventi critici. L'approccio ibrido concentra la maggior parte della logica di presentazione e della logica di business non critica sul client mobile, mantenendo il server responsabile della persistenza, della business logic critica e della distribuzione di aggiornamenti real-time. Questo design minimizza il numero di chiamate API necessarie e migliora significativamente le performance percepite dall'utente.

L'architettura è organizzata nel seguente modo:

- **Client:** Gestisce i layer di presentazione e parte della logica di business che è strettamente legata alle interazioni dell'utente con l'app prima dell'invio dei dati al server. La tecnologia adottata per questo tier è quella di Flutter.
- **Server:** Gestisce la logica di business e si occupa della persistenza e del recupero dei dati dal DB. La tecnologia adottata per questo tier è Python.
- **Database:** Gestisce i dati persistenti ed è in comunicazione solo con il server. La tecnologia di persistenza adottata è MongoDB.

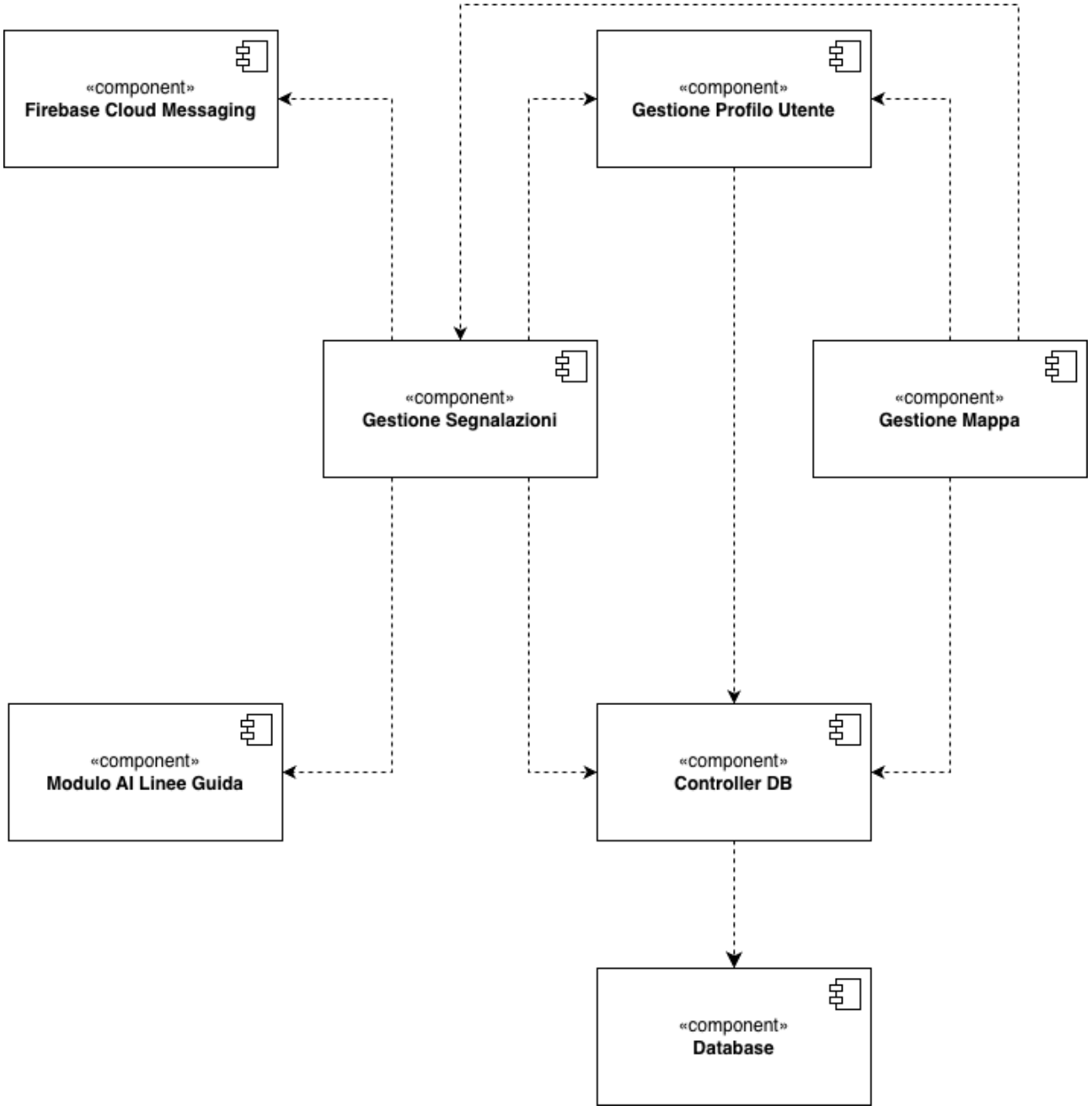
3.2 System Decomposition

I sottosistemi individuati per il nostro sistema sono:

- **Gestione Profilo Utente:**
  Si occupa delle seguenti funzionalità:
  - Registrazione del profilo utente.

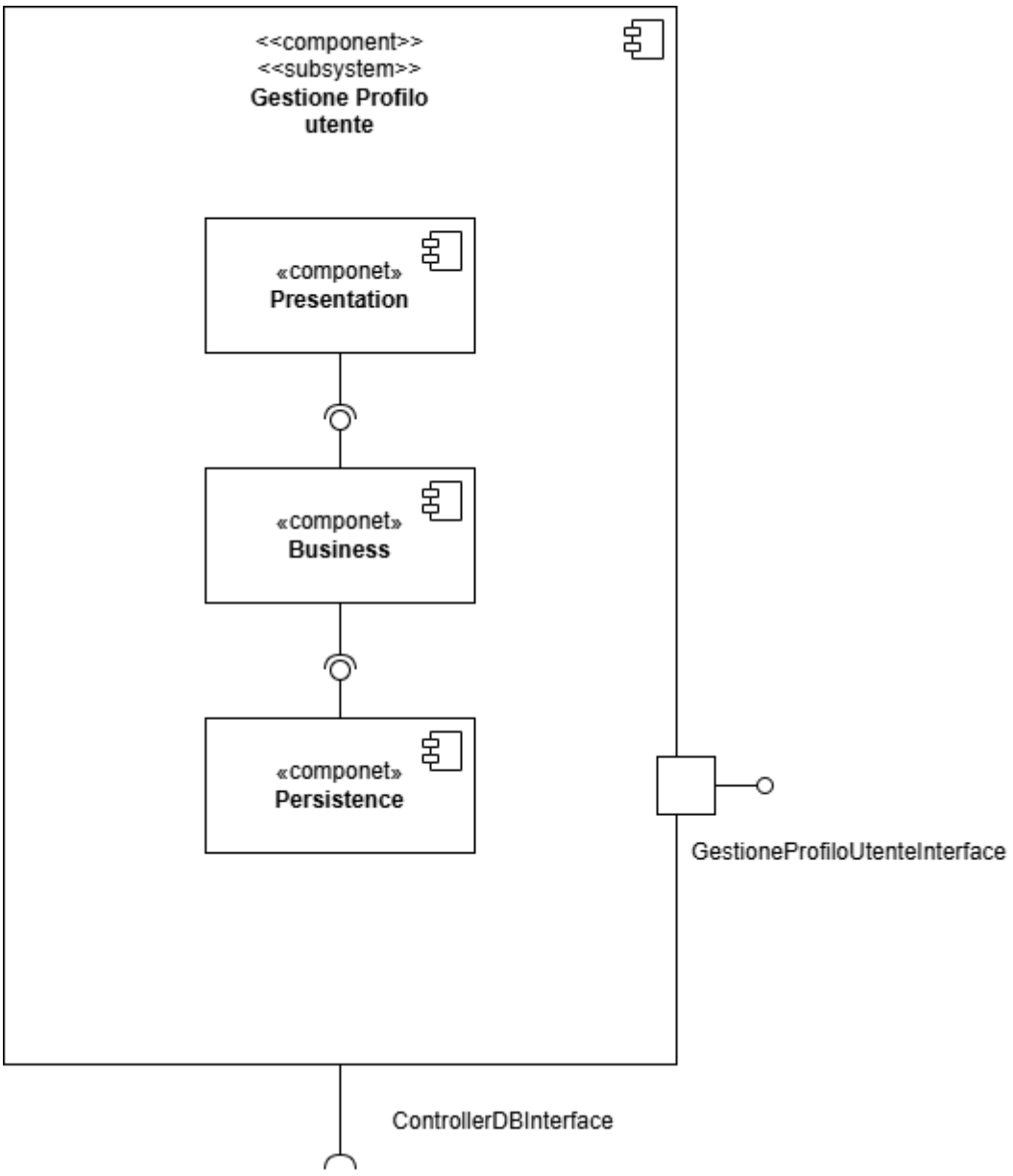
  - Modifica del profilo utente.
  - Cancellazione profilo utente.
  - Login.
  - Logout.
- **Gestione Segnalazioni:**
  Si occupa delle seguenti funzionalità:
  - Segnalazione manuale.
  - Segnalazione veloce.
  - Visualizzazione dei dettagli di un incidente.
  - Visualizzazione di linee guida.
  - Invio di Notifiche ai contatti preferiti.
  - Aggiornamento dello stato della segnalazione.
- **Gestione Mappa:**
  Si occupa delle seguenti funzionalità:
  - Visualizzazione della mappa.
  - Invio di Notifica d'incidente nei paraggi.
  - Visualizzazione delle segnalazioni attive.
  - Filtraggio per tipo d'incidente.
  - Classificazione degli incidenti per numero di segnalazioni.
- **Modulo AI Linee Guida (**implementato nel prossimo incremento previsto per il progetto di FIA**):**
  Si occupa della generazione delle Linee guida AI associate agli incidenti.
- **Firebase Cloud Messaging**:
  Si occupa della gestione delle notifiche push verso l'app client Android.
- **Controller DB:**
  Si interpone tra i vari sottosistemi e il sottosistema Database.
- **Database:**
  Si occupa della persistenza dei dati.

3.2.1 Component Diagram

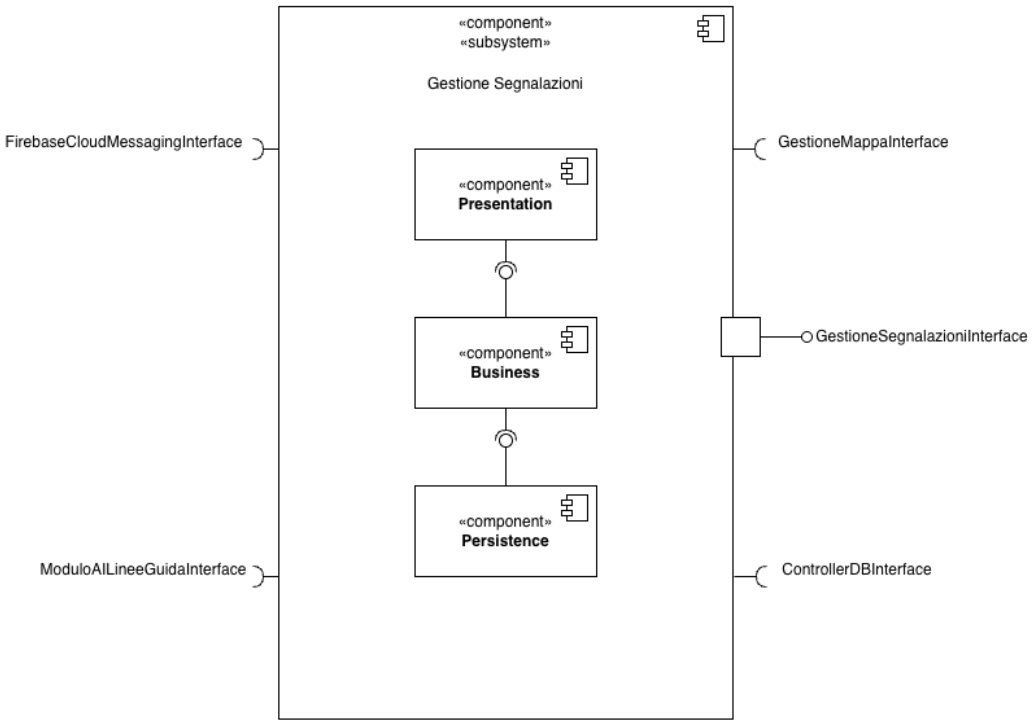


3.2.2 Diagramma Architetturale

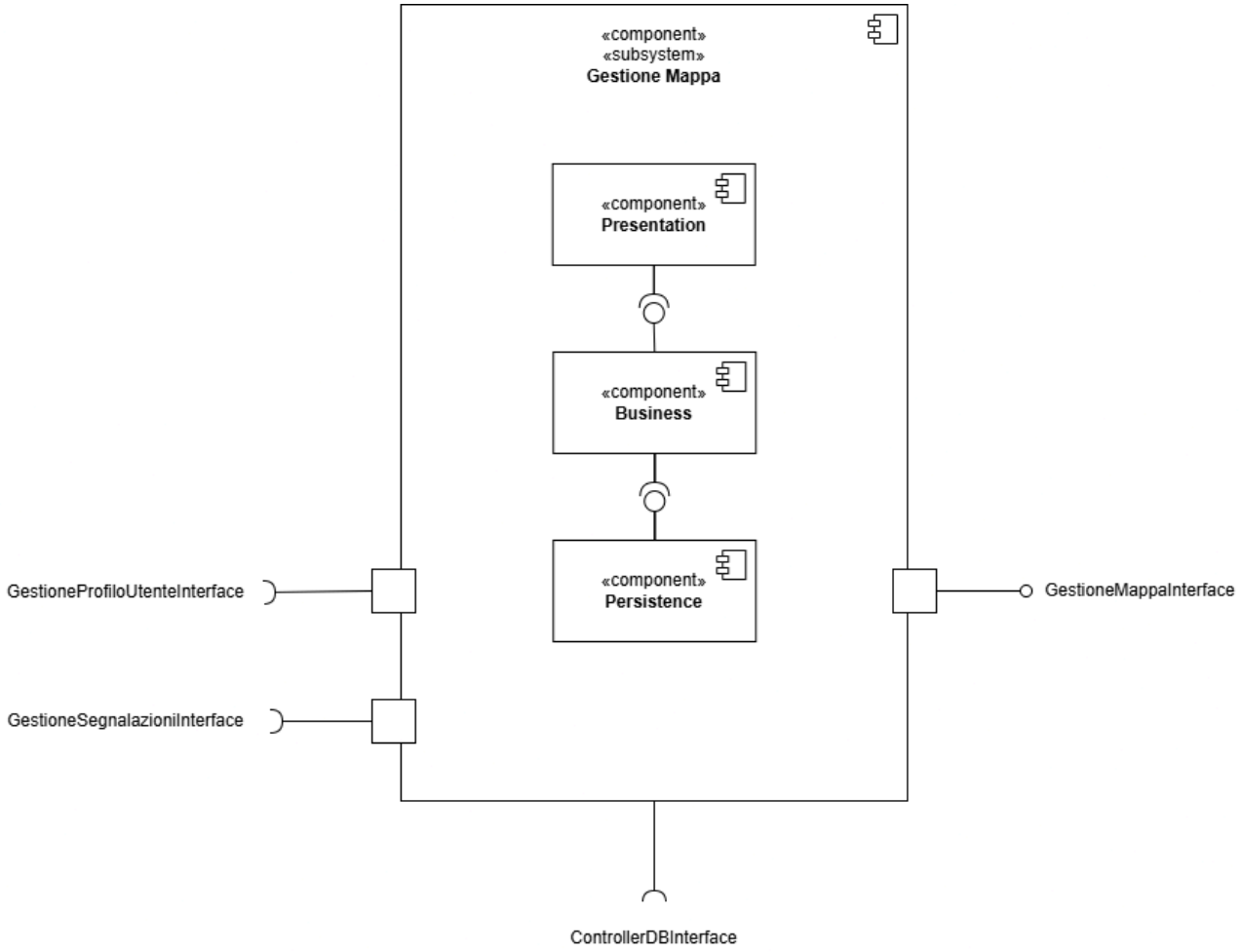
3.2.2.1 Sottosistema Gestione Profilo Utente



3.2.2.2 Sottosistema Gestione Segnalazioni



3.2.2.3 Sottosistema Gestione Mappa



3.3 Mapping Hardware/Software

L'applicazione Android integrerà logica di presentazione e parte della logica di business principalmente basata sulla correttezza del formato dei dati. Il database MongoDB verrà installato sulla macchina Windows/Mac che farà da server dove verranno installati anche il restante della logica di business implementata con python, il modello AI delle linee guida, e firebase cloud messaging.

3.3.1 Interface and Application Logic Layer: Client

- **Gestione profilo utente**:
  - Interfaccia di login.
  - Interfaccia di logout.
  - Interfaccia di registrazione.
  - Logout.
  - Validazione locale dei dati di registrazione e modifica.
  - Controllo formale dell' email e password.
- **Gestione segnalazioni**:

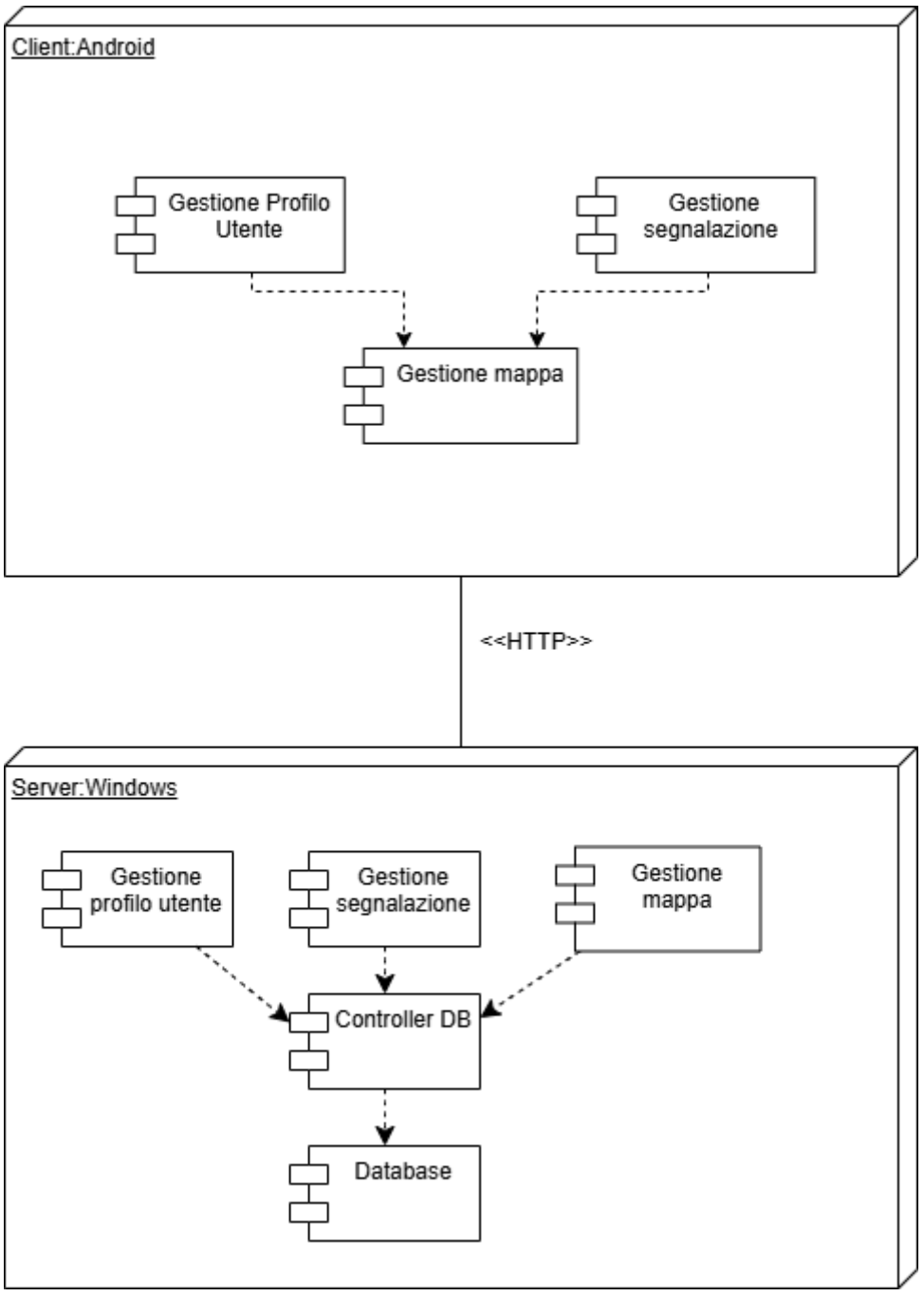
- Pre-validazione dei dati prima dell'invio.
- Raccolta dati GPS in tempo reale.
- Visualizzazione dettagli incidente.
- Visualizzazione linee guida per incidente.
- Ricezione notifiche push inviate dal server.
- Visualizzazione pop-up alert all'interno dell'app.

- **Gestione mappa**:
  - Renderizzazione mappa.
  - Visualizzazione icone incidenti aggiornate.
  - Filtraggio lato client basato su tipologie d'incidente.

3.3.2 Interface and Application Logic Layer: Server

- **Gestione profilo utente**:
  - Login.
  - Registrazione.
  - Hashing sicuro delle password.
  - Sincronizzazione dati sul DB.
- **Gestione segnalazioni**:
  - Registrazione della posizione GPS della segnalazione.
  - Aggiornamento stato segnalazione.
- **Gestione mappa**:
  - Verifica quali utenti si trovano nel raggio definito.
  - Preparazione notifica push.
  - Invio tramite FCM.
- **Modulo AI linee guida**:
  - Generazione di linee guida basate sull'AI.

3.3.3 Deployment Diagram



3.4 Persistent Data Management

Per la gestione e la memorizzazione dei dati abbiamo scelto **MongoDB**, un database NoSQL che memorizza i dati in formato JSON. I motivi di questa scelta sono:

1. **Modello dati flessibile e intuitivo:** Non richiede di definire uno schema fisso: ogni documento in una collection può avere campi diversi.

2. **Ricchezza di query e funzionalità:** MongoDB offre funzionalità di interrogazione molto avanzate, tra cui:
    a. Aggregation pipeline
    b. Query ad-hoc
    c. Funzionalità avanzate quali indici secondari, TTL, Query geo-spaziali e transazioni multidocumento
3. **Sincronizzazione in tempo reale:** Supporta la sincronizzazione dei dati in tempo reale molto utile per l'aggiornamento istantaneo delle segnalazioni.
4. **Architettura e denormalizzazione ottimale:** MongoDB eccede nella denormalizzazione dei dati, una pratica chiave nel design NoSQL che migliora drasticamente le prestazioni in lettura.

3.5 Global Software Control

RoadGuardian è un sistema interattivo, nel quale il flusso delle operazioni viene attivato sia da comandi dati dall'utente tramite l'interfaccia grafica, sia da eventi generati automaticamente dal sistema. Quando viene impartito un comando o si verifica un evento, il sistema seleziona il componente di controllo corrispondente, che si occupa di gestire l'elaborazione e di instradare il flusso verso il sottosistema appropriato. In seguito all'attivazione di una funzionalità, il flusso prosegue verso i moduli applicativi che eseguono la logica prevista, eventualmente coinvolgendo il server per l'elaborazione dei dati e l'accesso alle informazioni persistenti. Una volta completata l'elaborazione, il risultato viene restituito al client. Per questi motivi, il sistema adotta un meccanismo di controllo del flusso di tipo **event-driven**, in quanto l'attivazione delle funzionalità deriva da eventi generati dall'utente o dal sistema, e il controllo viene gestito dai componenti incaricati di coordinare la sequenza delle operazioni.