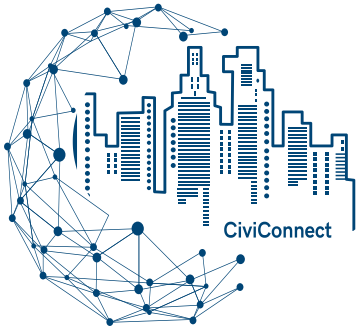
Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software* - Prof.ssa F. Ferrucci e Prof. F. Palomba



System Design Document

CiviConnect

Riferimento	C14_SDD_ver.0.1
Versione	1.0
Data	12/10/2024
Destinatario	ANCI-DTD, Prof. Filomena Ferrucci, Prof. Fabio Palomba
Presentato da	ConnecTeam

Approvato da	Bacco Luigi, Scala Benedetto

Revision History

Data	Versione	Descrizione	Autori
11/11/2024	0.1	Primo Template	MC
12/11/2024	0.2	Prima stesura dei Design Goals	Tutto il team
13/11/2024	0.3	Stesura Trade-Off	Tutto il team
13/11/2024	0.4	Definizione Rank Design Goals	Tutto il team
13/11/2024	0.6	Stesura Obiettivo del sistema	MB
13/11/2024	0.7	Stesura paragrafo 3.2	FF, GC
13/11/2024	0.8	Stesura Diagramma Architetturale	MC, GSP
15/11/2024	0.9	Stesura paragrafo 3.1	GG
15/11/2024	0.10	Modifica paragrafo 3.2	GC, FF
16//11/2024	0.11	Modifica Diagramma Architetturale	MC, GSP
17/11/2024	0.12	Stesura Paragrafi 1.4, 1.5	MC
19/11/2024	0.13	Review Documento	Tutto il team
15/12/2024	1.0	Revisione Generale del documento	Tutto il Team

Team Members

Nome	Acronimo	Contatto
Luigi Bacco	LB	l.bacco2@studenti.unisa.it
Benedetto Scala	BS	b.scala1@studenti.unisa.it
Generoso Sorridi	GSO	g.sorridi@studenti.unisa.it
Marco Brescia	MB	m.brescia16@studenti.unisa.it
Francesco Faiella	FF	f.faiella16@studenti.unisa.it
Gianluigi Citro	GC	g.citro37@studenti.unisa.it
Giuseppe Speranza	GSP	g.speranza15@studenti.unisa.it
Manuel Cieri	MC	m.cieri1@studenti.unisa.it
Domenico Auriemma	DA	d.auriemma4@studenti.unisa.it
Giuseppe Gambardella	GG	g.gambardella27@studenti.unisa.it

Sommario

1. Introduzione

Questo documento è dedicato al system design del sistema *CiviConnect,* alla definizione dei design goals, dell'architettura di sistema e trade-off implementativi.

1.1 Obiettivo del Sistema

CiviConnect ha come obiettivo principale quello di fornire una piattaforma che permetta ai cittadini di segnalare pericoli, dissesti e malfunzionamenti riscontrate sul territorio urbano. Il sistema permette ai cittadini di comunicare in modo efficace le problematiche all'amministrazione locale e di partecipare attivamente al processo di cura e miglioramento della città, contribuendo a rendere l'ambiente urbano più sicuro e vivibile per tutti.

Il sistema consente la registrazione ai cittadini che possono accedere alla piattaforma per inviare segnalazioni e monitorarne lo stato di avanzamento. Inoltre, il sistema permette alle amministrazioni comunali di gestire le problematiche segnalate.

1.2 Design Goals

In questa sezione sono specificati tutti i design goals cui il sistema punta a raggiungere.

Ogni Design goal ha:

- Un id
- Un rank
- Un nome
- Una descrizione
- Una categoria (Dependability, Performance, Maintenance, Cost, End User)
- I/Il RNF a cui è associato

Ad ogni design goal è assegnato un Rank, ovvero un numero che designa l'importanza del design goal a cui esso fa riferimento. Un numero più basso indica una priorità più alta.

La sezione corrente specifica, inoltre, una serie di trade-off. Lo scopo di questi ultimi nasce dall'evidenza che un sistema perfetto è difatti impossibile. Questa conclusione porta alla necessità di bilanciare alcune scelte, in modo tale da avere miglioramenti in alcuni aspetti a discapito di altri.

Ogni Trade-Off ha:

- Un nome che indica gli aspetti di riferimento
- Una descrizione che indica le scelte prese dagli sviluppatori

1.2.1 Design Goals

ID	Rank	Nome	Descrizione	Categoria	RNF di Rif.
DG_1	11	Gestione Mappe	La visualizzazione delle HeatMap sarà garantita dall'uso delle API Open Street Map.	Maintenance	RNF_IN_1
DG_2	8	Gestione Concorrenza Utenti	La gestione della concorrenza degli utenti verrà garantita dal server Firebase.	Maintenance	RNF_PE_1
DG_3	1	Applicazione Mobile	Il sistema deve essere sviluppato seguendo lo standard architetturale Three-Tier per piattaforme Android 12 o successive	Maintenance	RNF_IM_1, RNF_PA_1
DG_4	2	Gestione Database	Il servizio di persistenza dei dati verrà garantito tramite l'utilizzo di un database Firestore.	Maintenance, Performance	RNF_IN_2, RNF_PE_3
DG_5	9	Tempo di Risposta	L'utente dovrà ricevere una risposta alle sue operazioni effettuate in un arco di 20 secondi.	Performance	RNF_PE_4
DG_6	5	Confidenzialità dei dati	Il sistema deve permettere una divisione tra le varie categorie di utenti, ovvero garantire che i dati e le risorse siano preservati dal possibile utilizzo o accesso	Dependability	RNF_AF_1, RNF_AF_3

			da parte di soggetti non autorizzati a compiere una determinata operazione.		
DG_7	6	Manutenibilità	Il sistema deve essere progettato per garantire semplicità nell'esecuzione di manutenzioni ordinarie, adottando una scomposizione modulare. La manutenibilità sarà misurata attraverso il Tempo Medio di Modifica di un Modulo (TMMM) con obiettivo entro le 4 ore.	Dependability	RNF_SU_1, RNF_SU_2, RNF_IM_3
DG_8	3	Notifica di fallimento	Il sistema deve segnalare all'utente l'errore, notificandoglielo attraverso un messaggio.	End User	RNF_AF_2
DG_9	10	Codice Libero	Trasparenza del codice sorgente come richiesto da software governativi.	End User	RNF_LE_3
DG_10	7	Trasparenza	Fornire una sezione informativa che permetta agli utenti di conoscere lo stato di avanzamento delle proprie segnalazioni.	End User	RNF_LE_2
DG_11	4	Facilità d'utilizzo	Il sistema deve risultare facilmente comprensibile ed utilizzabile anche da	End User	RNF_US_1, RNF_US_2, RNF_US_3

		un'utenza meno esperta, facendo uso delle "8 regole d'oro di Shneiderman" per il design delle interfacce grafiche.

1.2.2 Trade-Offs

Trade-Off	Descrizione
Manutenibilità / Sicurezza vs Performance	Si è scelto di preferire un sistema più manutenibile adottando un'architettura chiusa con controlli diversificati a più livelli anche a scapito di una perdita di performance.
Tempo di rilascio vs Funzionalità	Si punterà su un rilascio rapido dell'app, sacrificando alcune funzionalità per raggiungere il mercato nel minor tempo possibile.

1.3 Definizioni, Acronimi e Abbreviazioni

Si elencano tutte le definizioni, acronimi e abbreviazioni presenti all'interno del documento:

- **Categoria Segnalazione**: Rifiuti, Dissesto stradale, Manutenzione, Illuminazione
- **Stato Segnalazione**: In Verifica, Accettata, In lavorazione, Completata, Scartata
- **Priorità Segnalazione**: Bassa, Media, Alta, Non impostata
- **SDD**: System Design Document
- **GU**: Gestione Utente
- **GA**: Gestione Admin
- **GSC:** Gestione Segnalazione Cittadino
- **GSA**: Gestione Segnalazione Comune
- **AD**: Analisi Dati – Report
- **RF**: Requisito Funzionale
- **RNF**: Requisito Non Funzionale
- **AC**: Amministratore Comunale
- **DG**: Design Goal
- **CIA**: Confidenzialità, Integrità, Disponibilità

1.4 Riferimenti

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

Statement of Work

Requirements Analysis Document

Design Pattern Document

Matrice di tracciabilità

1.5 Overview del resto del documento

Il presente System Design Document descrive l'architettura del sistema CiviConnect, fornendo dettagli sulla progettazione, sui design goals, e sui trade-off implementativi necessari per raggiungere gli obiettivi del progetto.

Nel Capitolo 2, si analizza il contesto attuale, evidenziando le problematiche affrontate dal sistema proposto e sottolineando l'assenza di soluzioni centralizzate comparabili.

Il Capitolo 3 introduce l'architettura del sistema proposto, descrivendo la suddivisione in strati Three-Tier (Presentation Layer, Service Layer, Data Layer) e fornendo una decomposizione in sottosistemi principali. Viene inoltre presentato il diagramma delle componenti e il diagramma architetturale per chiarire le relazioni tra i vari elementi.

Nel Capitolo 4, viene esposta la mappatura hardware/software del sistema, specificando come i diversi strati dell'architettura interagiscono con l'infrastruttura fisica e i servizi cloud, in particolare le API di Firebase, tra cui Firestore, Authenticator e Cloud Functions.

Infine, il documento fornisce un glossario dei termini utilizzati per agevolare la comprensione e una lista di riferimenti utili per approfondimenti.

2. Architettura Attuale

Al momento, non esiste alcun software che metta a disposizione tutte le funzionalità offerte da CiviConnect. Le amministrazioni locali, infatti, non dispongono di uno strumento centralizzato che semplifichi la gestione delle segnalazioni urbane e promuova la trasparenza. I cittadini si avvalgano spesso dei social network per segnalare problemi come buche stradali o rifiuti abbandonati, disperdendo tra i post le segnalazioni e rendendo difficile per le amministrazioni monitorarle e rispondere in modo tempestivo. Per questo motivo, non è possibile effettuare uno specifico confronto con architetture già esistenti.

3. Architettura del Sistema Proposto

3.1 Sintesi del Sistema

Civiconnect ha come obbiettivo quello di segnalare i pericoli, i dissesti e i malfunzionamenti all'interno dei comuni. Attraverso l'applicazione viene fornita ai cittadini la possibilità di segnalare, in modo centralizzato, al proprio comune le problematiche presenti in città.

Il sistema proposto è basato su architettura **Three-Tier**, mantenendo la suddivisione degli strati in Presentation Layer, Service Layer e Data Layer

Per quanto riguarda la parte di front-end e generazione delle interfacce del Presentation Layer, verrà utilizzato il framework di Flutter.

La logica della piattaforma, tradotta nel Service Layer, sarà sviluppata in Dart per la gestione delle richieste tra Presentation Layer e Service Layer, e verranno utilizzate le API di Firebase per accedere alla base dati.

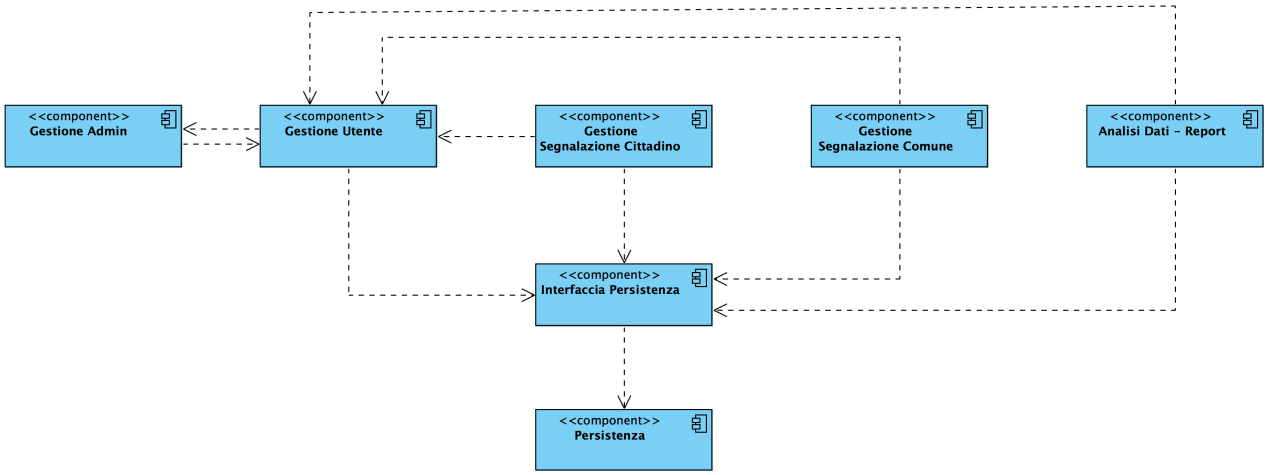
La persistenza dei dati, tradotta nel Data Layer, verrà realizzata un'interfaccia di persistenza e attraverso il servizio di database Cloud Firestore fornito da Firebase, al quale sarà possibile collegarsi attraverso le API fornite da Firebase.

3.2. Decomposizione in sottosistemi

Nella presente sezione sono stati individuati i seguenti sottosistemi:

- **Gestione Utente:** Si occupa della gestione dell'autenticazione, del login e del logout degli utenti, ossia Cittadino, Amministratore Comunale e Admin, e include funzionalità di registrazione, visualizzazione dell'area utente del cittadino e modifica dei dati dell'account per il Cittadino e l'Admin.
- **Gestione Admin:** Gestisce la generazione delle credenziali che verranno assegnate al AC.
- **Gestione Segnalazione Cittadino:** Per il cittadino si occupa di gestire le funzionalità riguardanti la creazione delle segnalazioni con le relative informazioni, permette di visualizzare le segnalazioni già presenti sulla piattaforma.
- **Gestione Segnalazione Comune:** Per l'AC, si occupa di gestire le funzioni riguardo la visualizzazione, modifica delle segnalazioni
- **Analisi Dati - Report:** Permette al AC di visualizzare dati riguardanti tutte le segnalazioni sulla piattaforma.
- **Interfaccia Persistenza:** Si interpone tra i sottosistemi e il sottosistema Persistenza.
- **Persistenza:** Gestisce la persistenza dei dati con un database.

3.2.1 Diagramma Delle Componenti



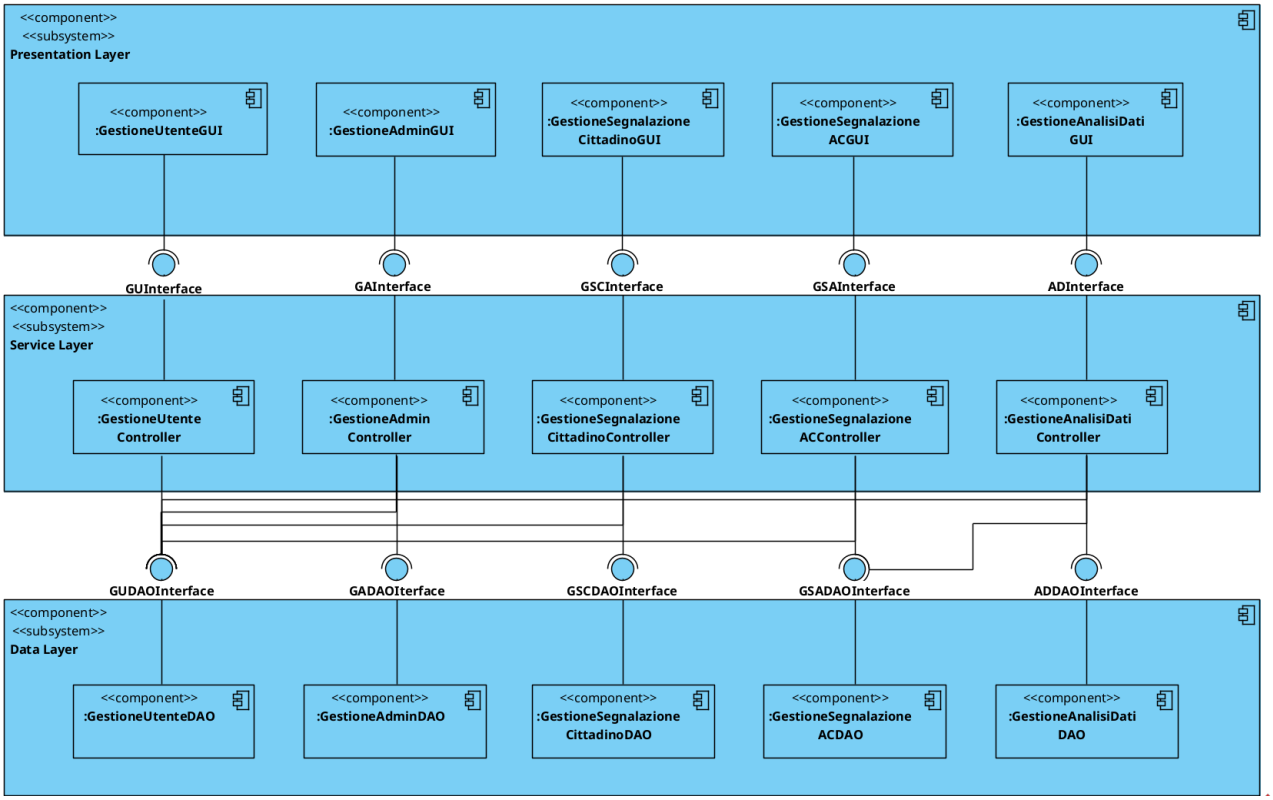
3.2.2 Architettura e Diagramma Architetturale

Architettura Three-Tier

Le seguenti caratteristiche motivano la scelta del Three-Tier come architettura, essendo conforme con i design goal e trade-off individuati e adattandosi bene con la scelta di un applicativo mobile:

- Separazione delle Responsabilità: Ogni livello ha responsabilità ben definite (UI, logica di business, accesso ai dati).
- Manutenibilità: Modifiche a uno strato non influenzano gli altri strati, rendendo più facile aggiungere o aggiornare le funzionalità.
- Scalabilità: Separando l'accesso ai dati dalla logica di business e dall'interfaccia, è possibile scalare e aggiornare ogni livello indipendentemente.

Tuttavia, l'architettura Three-Tier potrebbe aggiungere uno strato di complessità che senza un'accurata implementazione potrebbe risultare in un'opera di over-engineering per un'app mobile e diminuire le performance.


<<component>>
<<subsystem>>
Presentation Layer
<<component>>
:GestioneUtenteGUI
<<component>>
:GestioneAdminGUI
<<component>>
:GestioneSegnalazione
CittadinoGUI
<<component>>
:GestioneSegnalazione
ACGUI
<<component>>
:GestioneAnalisiDati
GUI
GUInterface
GAInterface
GSCInterface
GSAInterface
ADInterface
<<component>>
<<subsystem>>
Service Layer
<<component>>
:GestioneUtente
Controller
<<component>>
:GestioneAdmin
Controller
<<component>>
:GestioneSegnalazione
CittadinoController
<<component>>
:GestioneSegnalazione
ACController
<<component>>
:GestioneAnalisiDati
Controller
GUDAOInterface
GADAOIterface
GSCDAOInterface
GSADAOInterface
ADDAOInterface
<<component>>
<<subsystem>>
Data Layer
<<component>>
:GestioneUtenteDAO
<<component>>
:GestioneAdminDAO
<<component>>
:GestioneSegnalazione
CittadinoDAO
<<component>>
:GestioneSegnalazione
ACDAO
<<component>>
:GestioneAnalisiDati
DAO

3.3. Mapping Hardware/Software

Il sistema utilizzerà un'architettura hardware di tipo Three-Tier.
Il client è rappresentato da un dispositivo mobile Android su cui è installata l'applicazione, che necessita di una connessione Internet per connettersi al server e interagire con il sistema.

Il Presentation Layer gestisce l'interfaccia utente, occupandosi della visualizzazione dei dati e della ricezione degli input forniti dall'utente.

Il Service Layer è responsabile della logica applicativa. Gestisce le operazioni principali del sistema, come la gestione degli utenti e delle segnalazioni. Coordina le richieste provenienti dal Presentation Layer e interagisce con il Data Layer utilizzando le API di Firebase.

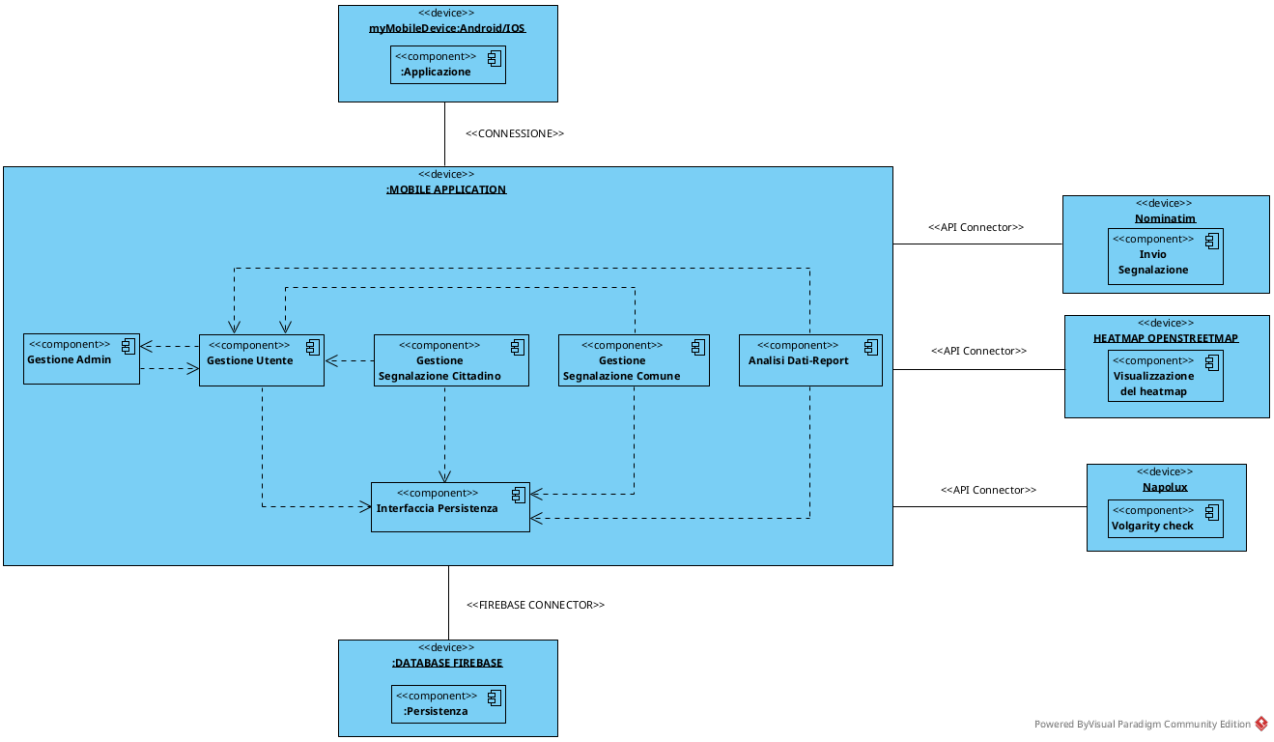
Il Data Layer, tramite l'interfaccia di persistenza di Firebase, gestisce e conserva i dati persistenti relativi al dominio dell'applicazione.

Nel sistema verranno utilizzate le API OpenStreetMap, un progetto open source che raccoglie dati geografici, come strade, edifici e punti di interesse, attraverso una comunità globale di volontari. I dati sono liberamente accessibili e possono essere utilizzati per una vasta gamma di applicazioni. Il progetto offre diverse API per gestire, analizzare e visualizzare dati geografici, inclusa la creazione di mappe stradali personalizzate.

L'integrazione di Flutter con Firebase presenta diversi vantaggi e svantaggi:

- **Vantaggi**:
  - **Sviluppo rapido**: Flutter è un framework che consente lo sviluppo di applicazioni mobili cross-platform, permettendo la creazione di app per Android a partire da un unico codice sorgente. Questo approccio riduce i tempi di sviluppo e i costi di manutenzione legati alla gestione delle applicazioni su diverse piattaforme.
  - **Integrazione nativa con Firebase**: Firebase offre una vasta gamma di servizi (autenticazione, database in tempo reale, cloud storage, analitiche, ecc.) facilmente integrabili con Flutter grazie ai pacchetti ufficiali forniti da Google. Ciò riduce il lavoro di configurazione e accelera lo sviluppo.
  - **Backend as a Service (BaaS)**: Firebase fornisce una soluzione backend completa e scalabile, eliminando la necessità di creare e gestire un'infrastruttura server dedicata. Questo rende il sistema più semplice da gestire e più economico nelle fasi iniziali del progetto.
  - **Aggiornamenti in tempo reale**: Firestore e altri servizi Firebase offrono funzionalità in tempo reale che aggiornano automaticamente l'interfaccia utente quando i dati cambiano, migliorando così l'esperienza utente.

- **Svantaggi**:
  - **Dipendenza dall'ecosistema Google**: L'utilizzo di Firebase comporta una forte dipendenza dall'ecosistema Google, che potrebbe risultare problematica qualora si desideri migrare verso un'altra infrastruttura o servizio. Questo effetto di "vendor lock-in" può limitare la flessibilità a lungo termine.


<<device>>
myMobileDevice:Android/IOS
<<component>>
:Applicazione
<<CONNESSIONE>>
<<device>>
:MOBILE APPLICATION
<<component>>
Gestione Admin
<<component>>
Gestione Utente
<<component>>
Gestione
Segnalazione Cittadino
<<component>>
Gestione
Segnalazione Comune
<<component>>
Analisi Dati-Report
<<component>>
Interfaccia Persistenza
<<API Connector>>
<<device>>
Nominatim
<<component>>
Invio
Segnalazione
<<API Connector>>
<<device>>
HEATMAP OPENSTREETMAP
<<component>>
Visualizzazione
del heatmap
<<API Connector>>
<<device>>
Napolux
<<component>>
Volgarity check
<<FIREBASE CONNECTOR>>
<<device>>
:DATABASE FIREBASE
<<component>>
:Persistenza
Powered ByVisual Paradigm Community Edition

4. Glossario

Terminologia	Definizione
TMMM	Il Tempo Medio di Modifica di un Modulo (TMMM) include localizzazione, modifica e verifica del modulo. Un TMMM di 4 ore indica, per una modifica ordinaria, dei tempi di localizzazione del modulo in meno di 30 minuti, modifica e implementazione in meno di 2 ore, test locali e verifiche in meno di 1,5 ore.
8 Regole di Shneirderman	• Coerenza nell'interfaccia, per ogni tipo di azione • Usabilità universale per tutti • Riscontri delle azioni eseguite • Progettare spazi di input e di output per ottenere un risultato preciso • Prevenire gli errori degli utenti • Rendere le azioni quanto più reversibili possibili • Fornire quanto più controllo delle azioni all'utente • Ridurre il carico di memoria a breve termine
Firebase	Firebase è una piattaforma serverless per lo sviluppo di applicazioni mobili e web. Open source ma supportata da Google, Firebase fornisce una suite di strumenti per scrivere, analizzare e mantenere applicazioni cross-platform.
Firestore	Database cloud di tipo NoSQL flessibile e scalabile offerto da Google Cloud, usato per lo storage e la sincronizzazione di dati per lo sviluppo client- e server-side.
Framework	Architettura logica di supporto sulla quale un software può essere progettato e realizzato fornendo un'infrastruttura generale lasciando al programmatore il contenuto vero e proprio dell'applicazione.

Three-tier	L'architettura Three-tier suddivide il sistema in Presentation Layer che si occupa dell'interfaccia grafica, il Service Layer che si occupare della logica di business e Data Layer che gestisce la persistenza dei dati.