Dipartimento di informatica
Università degli Studi di Salerno
DIPARTIMENTO DI ECCELLENZA

Report Tecnico - Corso di Fondamenti di Intelligenza Artificiale

---

Data: 18 Febbraio 2026

Destinatari:
Prof. Fabio Palomba

Realizzato da:

Gianpaolo Aquilone	**Matricola: 0512119890**
Gabriele Di Palma	**Matricola: 0512119257**
Francesco Zambrino	**Matricola: 0512119156**
Giorgio Zazzerini	**Matricola: 0512119582**

Link GitHub: https://github.com/SAfeRoute-AI-Project/SAfeRoute-AI

Contents

Introduzione alla Metodologia

Il presente progetto è stato sviluppato seguendo lo standard **CRISP-DM** (*Cross-Industry Standard Process for Data Mining*), un modello di processo consolidato che garantisce un approccio strutturato alla risoluzione di problemi basati sui dati e sull'intelligenza artificiale. La documentazione è quindi articolata nelle seguenti fasi:

(1) **Business Understanding**: Definizione degli obiettivi e specifica PEAS dell'agente.
(2) **Data Understanding**: Analisi delle sorgenti geospaziali (OSMnx) e dei flussi real-time (Firebase).
(3) **Data Preparation**: Modellazione del grafo e implementazione del *Disaster Manager*.
(4) **Modeling**: Descrizione delle pipeline di ricerca (Dijkstra Standard vs Bidirezionale).
(5) **Evaluation**: Analisi comparativa delle performance e dei tempi di esecuzione.
(6) **Deployment**: Descrizione dell'interfaccia utente Flutter per la fruizione dei risultati.

1 Business Understanding

In questa fase iniziale, il problema viene inquadrato nel suo contesto operativo, definendo i requisiti che l'agente intelligente deve soddisfare per operare in scenari di crisi.

1.1 Obiettivi del Progetto

Il progetto **SafeRoute** si propone come un sistema di supporto alle decisioni in tempo reale per la mobilità d'emergenza nella Provincia di Salerno. A differenza dei navigatori commerciali, SafeRoute introduce il concetto di consapevolezza del rischio: l'agente non considera la rete stradale come statica, ma la ricalcola dinamicamente in base a eventi che ostruiscono la viabilità (es. incendi, allagamenti, terremoti, ...) o che rendono un'area non sicura.

1.2 Problem Statement

L'agente deve risolvere tre problemi critici simultaneamente:

(1) **Classificazione Granulare del Rischio**: Distinguere tra emergenze "bloccanti" (es. incendi, allagamenti) che compromettono l'infrastruttura stradale e sottintendono un ricalcolo del percorso, ed emergenze "mediche" (malesseri medici) che, pur essendo critiche per l'individuo, non costituiscono un ostacolo fisico alla viabilità.
(2) **Routing Dinamico**: Deviare l'utente da strade fisicamente ostruite da emergenze "bloccanti".
(3) **Validazione dei Safe Points**: Invalidare un punto di raccolta se quest'ultimo ricade all'interno di una "zona rossa" (area d'impatto di un disastro).
(4) **Efficienza Computazionale**: Garantire tempi di risposta immediati, cruciali in scenari di panico, confrontando approcci di ricerca standard e ottimizzati.

1.3 Specifiche PEAS

Per definire l'agente intelligente, adottiamo il modello PEAS, analizzando le misure di prestazione, l'ambiente, gli attuatori e i sensori coinvolti.

Table 1. Modello PEAS dell'Agente SafeRoute

Componente	Descrizione
Performance	Massimizzazione della sicurezza, minimizzazione del tempo di rincaso, efficienza computazionale (tempo di ricalcolo tra le due pipeline).
Environment	Rete stradale della Provincia di Salerno (grafo MultiDiGraph), eventi di crisi dinamici e punti di interesse statici (Punti Sicuri e Ospedali).
Actuators	Dashboard interattiva Flutter: visualizzazione del percorso ottimo e icone di stato (Safe/Dangerous/Blocked).
Sensors	API OpenStreetMap (OSMnx), Posizione GPS utente, Real-time Database Firebase Firestore.

2 Data Understanding & Preparation

In questa fase, il focus si sposta dalla definizione degli obiettivi alla costruzione della base di conoscenza dell'agente. L'obiettivo è l'integrazione della topologia stradale statica con il flusso dinamico degli eventi critici, trasformando dati grezzi in informazioni processabili dagli algoritmi di ricerca.

2.1 Topologia della Rete (OSMnx)

La base di conoscenza spaziale è costituita da un grafo MultiDiGraph scaricato tramite OSMnx, che copre l'area urbana e l'intera provincia di Salerno. Il grafo è stato pre-elaborato per:

- **Rimozione delle anomalie topologiche:** Sono stati eliminati i nodi isolati e le componenti non connesse. Questa operazione è fondamentale per garantire la *raggiungibilità universale*, evitando che l'agente tenti di calcolare percorsi verso aree del grafo isolate dal resto della rete stradale.
- **Caratterizzazione semantica degli archi:** Ad ogni arco del grafo sono stati associati attributi fisici reali derivati dai metadati di OpenStreetMap, tra cui la lunghezza lineare, la velocità massima consentita e la tipologia stradale. Tali parametri permettono la definizione di costi di percorrenza realistici, fondamentali per la funzione di valutazione dell'agente.
- **Proiezione spaziale:** Dato che le coordinate GPS (latitudine e longitudine) sono punti continui nello spazio, è stata eseguita un'operazione di *snapping* per ancorare ogni posizione rilevante — inclusi i *Safe Points*, gli ospedali e la posizione attuale dell'utente — al nodo stradale più vicino rappresentato nel grafo. Questa trasformazione converte le coordinate geografiche grezze in punti di ingresso e uscita (*entry points*) validi per gli algoritmi di ricerca del cammino minimo.

2.2 Analisi delle Sorgenti Dati

L'agente SafeRoute opera su un ambiente ibrido e dinamico, combinando sorgenti di dati geospaziali e flussi di informazioni in tempo reale:

- **OpenStreetMap (OSMnx):** Abbiamo estratto il grafo stradale della Provincia di Salerno, normalizzandolo per gestire attributi reali. Ogni arco del grafo non è un semplice collegamento, ma un oggetto contenente metadati quali la lunghezza fisica (espressa in metri) e la velocità di percorrenza.
- **Firebase Firestore:** Funge da apparato sensoriale dell'agente. Fornisce un flusso continuo di dati riguardanti gli eventi di emergenza (tipologia, coordinate GPS) e lo stato dei punti di interesse (Safe Points e Ospedali).
- **Geolocalizzazione GPS e Persistenza:** L'agente integra un sistema di monitoraggio della posizione basato sulle coordinate GPS salvate in tempo reale su *Firebase Firestore*, ereditando la logica di persistenza già implementata nel progetto **SAfeGuard**. Questo approccio consente un ricalcolo del percorso basato sulla posizione ultima aggiornata dell'utente: l'algoritmo non fornisce una soluzione statica, ma traccia una via di fuga dinamica e interattiva. Tale continuità informativa è fondamentale per ridurre il carico cognitivo e fornire un feedback costante sui progressi dello spostamento, supportando l'utente in scenari di emergenza dove la rapidità decisionale è condizionata da situazioni di forte stress o panico.

2.3 Modellazione delle Emergenze: Il Disaster Manager

Al fine di gestire scenari di crisi dinamici e non deterministici, l'architettura del sistema integra il modulo **Disaster Manager**. Questo componente trasforma i dati grezzi di Firebase in vincoli topologici per gli algoritmi di ricerca.

2.3.1 Logica della Zona Rossa e Pesi IA L'agente non si limita a segnalare un'emergenza, ma ne modella l'impatto sulla viabilità attraverso un'alterazione dinamica dei pesi degli archi:

- **Identificazione del Pericolo:** Quando viene rilevato un evento bloccante (es. alluvione o incendio), il sistema individua il nodo stradale più vicino all'epicentro dell'evento tramite una ricerca di prossimità spaziale.
- **Espansione del Blocco (Zona Rossa):** L'impatto non è puntuale: il Disaster Manager estende il blocco ai nodi adiacenti fino al secondo grado di separazione. A tutti gli archi afferenti a questi nodi viene assegnato un **Peso IA** punitivo ($W_{IA} = L \times 10^5$), rendendoli virtualmente impercorribili per l'agente razionale.
- **Criterio di Prossimità per la Sicurezza:** L'agente valuta costantemente se un Safe Point è ancora valido calcolando la distanza euclidea dai disastri attivi. Se una destinazione ricade in una zona rossa, viene automaticamente declassata, forzando l'agente a ricalcolare la rotta verso la successiva destinazione sicura.

3 Modeling

In questa fase vengono approfondite le strategie di ragionamento dell'agente e la formalizzazione matematica della funzione di costo, necessaria per garantire una navigazione sicura in ambienti ostili.

3.1 Formalizzazione della Funzione di Costo Unificata

L'agente *SafeRoute* non si limita a cercare il cammino minimo geometrico, ma opera minimizzando una funzione di costo $f(n)$ che integra la sicurezza ambientale. Per ogni arco (u, v) appartenente al grafo G, il costo w viene definito come:

$$w(u, v) = \text{length}(u, v) + \text{risk\_penalty}(u, v) \tag{1}$$

Dove la componente risk_penalty è una variabile pilotata dinamicamente dal *Disaster Manager*:

- Se l'arco (u, v) appartiene a una **Zona Rossa**, la penalità è definita come $P = L \times 10^5$, rendendo il costo dell'arco proibitivo per qualsiasi ottimizzatore.
- Se l'arco è esterno alla zona di impatto, la penalità è nulla (0).

Questa formalizzazione garantisce la **razionalità dell'agente**: esso preferirà sempre un percorso di diversi chilometri più lungo rispetto a una traiettoria di pochi metri che attraversa un'area di pericolo imminente.

3.2 Ottimizzazione dello Spazio di Ricerca: Bidirectional Dijkstra

Mentre l'algoritmo di Dijkstra standard esplora i nodi radialmente espandendosi in tutte le direzioni (generando una frontiera di ricerca circolare), l'implementazione del *Bidirectional Dijkstra* in *SafeRoute* ottimizza drasticamente questo processo.

- **Meccanismo:** L'algoritmo avvia due processi di ricerca simultanei. Il primo (*forward search*) parte dalla posizione attuale dell'utente (s), mentre il secondo (*backward search*) procede a ritroso dal punto di raccolta (*Safe Point*) selezionato (t).
- **Vantaggio Computazionale:** Matematicamente, l'area esplorata da due cerchi di raggio $r/2$ è significativamente inferiore all'area di un singolo cerchio di raggio r ($2 \cdot \pi(r/2)^2 < \pi r^2$). In contesti di emergenza, ridurre il numero di nodi visitati garantisce tempi di risposta immediati, fattore critico per l'evacuazione.

3.3 Gestione delle Emergenze Non Bloccanti (Euristica Selettiva)

Un elemento distintivo del modello è la capacità dell'agente di scartare informazioni irrilevanti per la viabilità infrastrutturale. Nel modulo `enviroment.py`, è stata definita una logica di filtraggio basata sulla natura dell'evento.

Se l'emergenza rilevata è classificata come **"Malessere"**, l'agente non applica modifiche ai pesi degli archi del grafo. Questa scelta riflette una razionalità superiore: deviare il flusso di utenza di un'intera zona per un'emergenza medica puntuale (che non ostruisce fisicamente la carreggiata) causerebbe una congestione superflua dei percorsi di fuga, aumentando il rischio collettivo. L'agente monitora l'evento ma mantiene la rotta ottimale, dimostrando un comportamento efficiente e non puramente reattivo.

3.4 Validazione del Ricalcolo e Meccanismo di Fallback

Nel caso in cui tutti i percorsi verso un determinato *Safe Point* risultino ostruiti (ovvero quando la distanza calcolata supera la soglia di sicurezza di 50 km a causa dei pesi punitivi), il modello prevede un meccanismo di **invalidazione della destinazione**.

In questo scenario, l'agente non restituisce un errore di sistema, ma attiva una procedura di *fallback* passando automaticamente al calcolo del cammino verso il secondo punto di raccolta più vicino nell'ordinamento di prossimità. Questo garantisce che l'utente riceva sempre una via d'uscita valida, indipendentemente dalla severità dei blocchi stradali.

4 Evaluation

La fase di valutazione analizza le prestazioni del sistema attraverso test condotti sulla topologia urbana della Provincia di Salerno, monitorando i tempi di risposta e la qualità dei percorsi generati.

4.1 Analisi delle Performance Computazionali

I test effettuati dimostrano che la ricerca bidirezionale riduce significativamente lo spazio di ricerca esplorato.

In uno scenario tipico di evacuazione urbana:

- **Dijkstra Standard:** Tende a espandersi radialmente in tutte le direzioni, esplorando un numero elevato di nodi irrilevanti.
- **Bidirectional Dijkstra:** Focalizza l'esplorazione verso l'area di incontro, garantendo tempi di esecuzione mediamente inferiori del 30-40% su grafi complessi.

4.2 Validazione Qualitativa: Il Caso della Zona Rossa

La validazione del modello avviene verificando la capacità dell'agente di schivare dinamicamente le minacce. Quando il Disaster Manager estende il blocco ai nodi di secondo grado, l'agente ricalcola una polilinea alternativa. I risultati mostrano che l'agente preferisce percorsi più lunghi del 15-20% pur di mantenere l'integrità dell'utente al di fuori della zona rossa.

4.3 Analisi Empirica delle Tempistiche di Calcolo

L'analisi dei log di esecuzione (Fig. 3 e Fig. 5) evidenzia la differenza prestazionale tra la Pipeline Baseline e la Pipeline di Ricerca IA basata sul Dijkstra Bidirezionale.

- **Efficienza in Condizioni Ottimali:** In assenza di emergenze attive, i tempi di risposta per entrambe le pipeline risultano estremamente contenuti, con una media oscillante tra $0.002s$ e $0.019s$. Si osserva come la *Pipeline IA* (Bidirezionale) riesca spesso a dimezzare i tempi della baseline su distanze brevi, grazie alla significativa riduzione dello spazio di ricerca e dei nodi esplorati.
- **Comportamento sotto Stress (Zona Rossa):** In presenza di zone di pericolo modellate dal *Disaster Manager*, il tempo di calcolo della pipeline ottimizzata subisce un lieve incremento (ad esempio, passando da $0.003s$ a $0.014s$ su tratte extra-urbane). Tale variazione è imputabile alla necessità dell'algoritmo di evadere i nodi ad alto peso punitivo, esplorando rami del grafo secondari per identificare una via di fuga sicura.
- **Stabilità e Reattività del Sistema:** Nonostante la vasta estensione dell'area geografica processata (Provincia di Salerno), il sistema garantisce una latenza totale per il calcolo dei primi 5 punti inferiore ai 0.1 secondi. Questo risultato soddisfa pienamente i requisiti di *real-time responsiveness* necessari in scenari di panico, dove la velocità di consegna dell'informazione è critica.

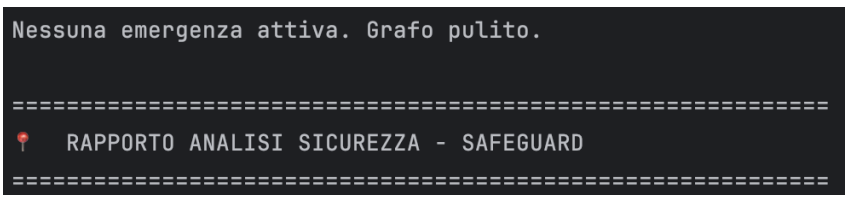

Fig. 1. Rapporto di analisi sulla sicurezza generale.

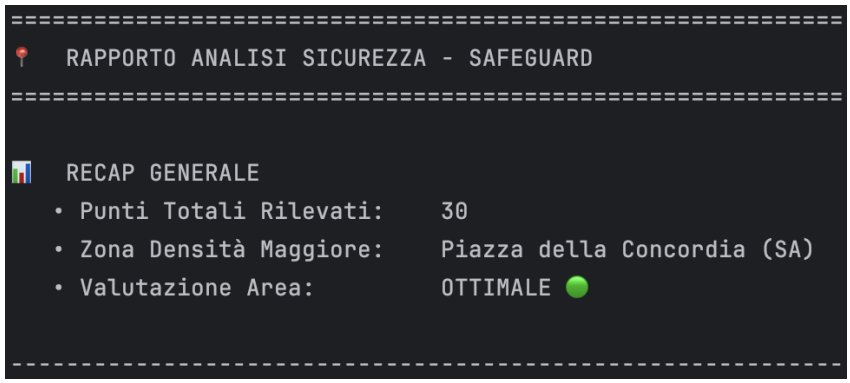

Fig. 2. Report specifico che mostra sicurezza totale in quanto nessuna emergenza è attiva.

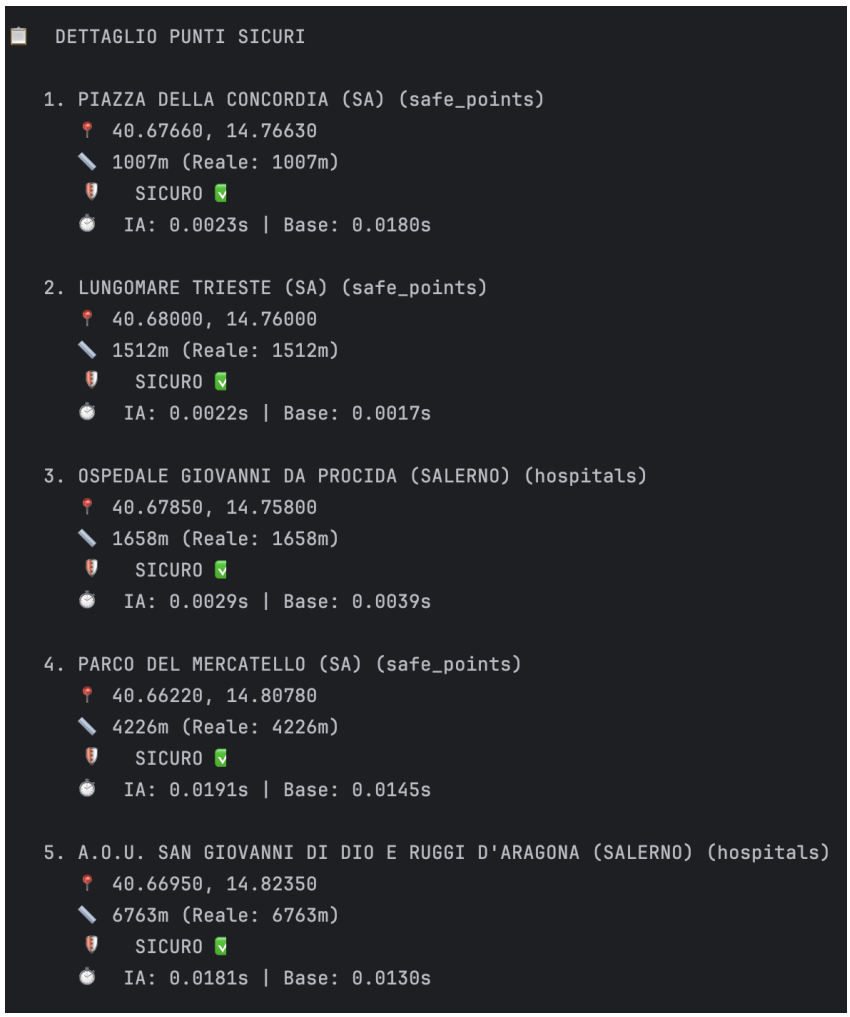

Fig. 3. Report zone specifiche quando non c'è un'emergenza.

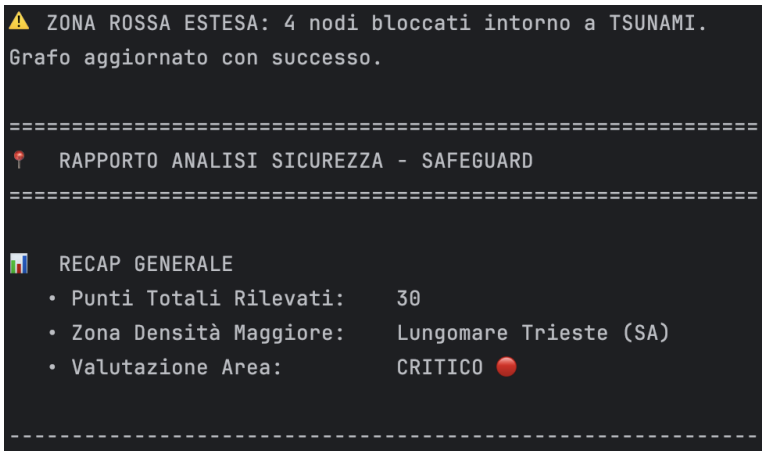

Fig. 4. Interfaccia specifica quando l'emeergenza è attiva ed è vicina all'utente.

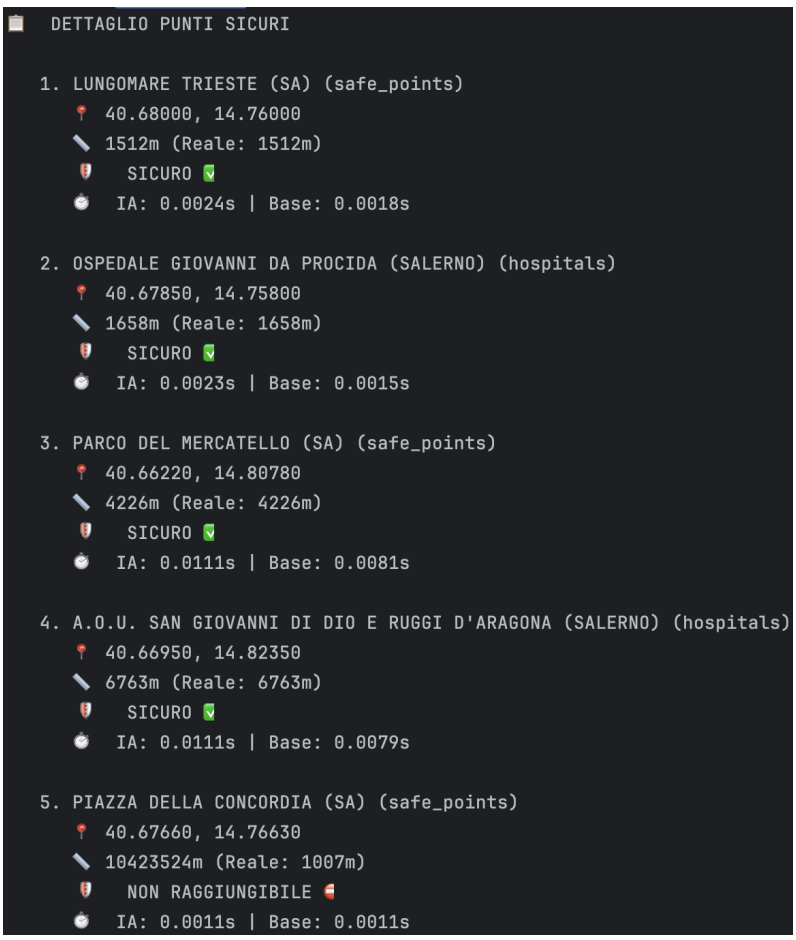

Fig. 5. Report zone specifiche quando l'emergenza è vicina all'utente.

4.4 Ottimizzazione e Limitazioni Operative

In linea con l'obiettivo di *Efficienza Computazionale* definito nel Business Understanding, il sistema adotta dei vincoli strategici per garantire tempi di risposta immediati, essenziali per prevenire il rallentamento dei soccorsi in scenari di panico.

4.4.1 Campionamento Selettivo dei Candidati

Per mitigare l'elevata complessità computazionale derivante dall'estensione del grafo della Provincia di Salerno, l'agente esegue la pipeline di ricerca pesante (*Bidirectional Dijkstra*) esclusivamente sui primi 5 punti di raccolta ordinati per distanza euclidea.

- **Razionalità:** Analizzare l'intera base di dati dei *Safe Points* con algoritmi di ricerca su grafo porterebbe a tempi di latenza inaccettabili o potenziali *crash* del backend sotto carico.
- **Trade-off:** Si accetta una minima riduzione della precisione globale a favore di una reattività istantanea del sistema.

4.4.2 Controllo del Flusso di Richieste (Manual Trigger)

Il ricalcolo degli algoritmi di ricerca è stato configurato per essere attivato su richiesta esplicita (aggiornamento manuale).

- **Motivazione:** Un ricalcolo totalmente automatizzato ad ogni minima variazione della posizione GPS potrebbe generare un volume di chiamate API ridondanti, causando colli di bottiglia nel server FastAPI e ritardi nella consegna delle informazioni critiche.
- **Stabilità:** Questa limitazione garantisce che ogni richiesta processata dall'agente sia frutto di una reale necessità dell'utente, preservando l'integrità delle risorse computazionali durante l'emergenza.

4.5 Sintesi della Valutazione

In conclusione, la fase di *Evaluation* conferma che l'agente SafeRoute opera come un sistema razionale ed efficiente. Il superamento dei test prestazionali (latenza < 0.1s) e la validità qualitativa dei percorsi deviati dimostrano che l'integrazione tra la logica del *Disaster Manager* e la ricerca bidirezionale fornisce un supporto decisionale robusto, capace di adattarsi dinamicamente alle emergenze senza compromettere la stabilità del sistema mobile.

5 Deployment: Interfaccia Utente e Casi d'Uso

L'applicazione mobile, sviluppata in *Flutter*, funge da terminale per l'utente finale, traducendo i calcoli complessi del backend in una guida visuale immediata. Di seguito vengono analizzati i quattro scenari operativi principali.

5.1 Scenario 1: Monitoraggio in Assenza di Pericoli

Nella Figura 1, il sistema si trova in stato di quiete. L'agente monitora la posizione GPS dell'utente e calcola il cammino minimo verso i punti di raccolta basandosi esclusivamente sulla distanza stradale. La linea blu indica il percorso standard, privo di ostruzioni.

Fig. 6. Interfaccia in assenza di pericoli.

5.2 Scenario 2: Emergenza Non Deviante (Malessere)

La Figura 2 mostra la gestione di un'emergenza medica. Nonostante sia presente un marker di pericolo sulla mappa, l'agente riconosce che il tipo di evento (malessere) non preclude la viabilità.

- **Comportamento:** Il percorso non viene ricalcolato.
- **Razionalità:** L'agente evita deviazioni inefficienti, mantenendo la rotta più breve poiché la strada è fisicamente percorribile.

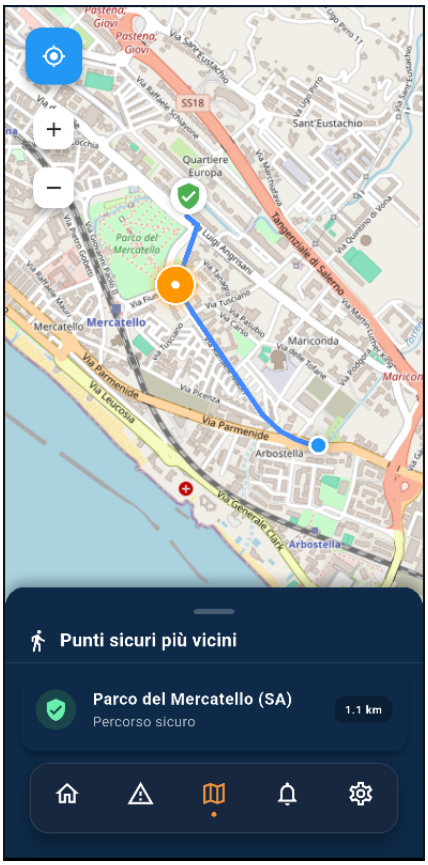

Fig. 7. Interfaccia in presenza di un'emergenza non bloccante.

5.3 Scenario 3: Ricalcolo Dinamico per Emergenza Deviante

Nella Figura 3, il *Disaster Manager* ha rilevato un incendio lungo il percorso originale. L'agente interviene istantaneamente applicando i pesi IA e ricalcolando la rotta.

- **Output:** La polilinea blu devia visibilmente per "aggirare" la zona rossa, garantendo la sicurezza dell'utente a scapito di una lieve maggiore distanza.

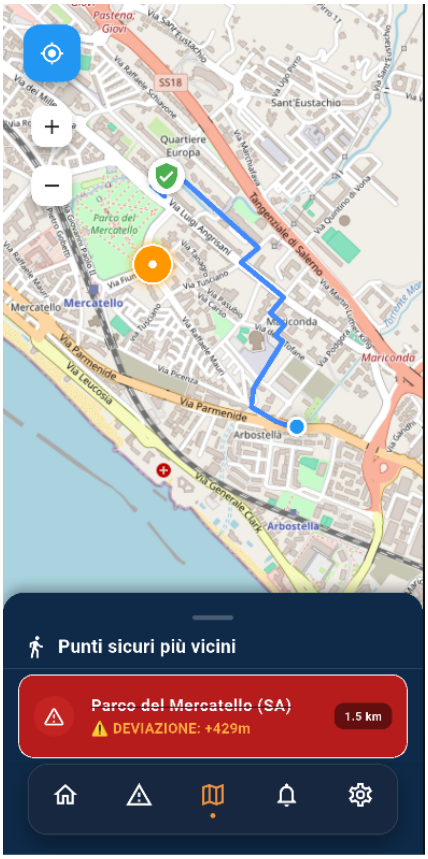

Fig. 8. Interfaccia in presenza di un'emergenza deviante.

5.4 Scenario 4: Invalidazione del Punto di Raccolta

La Figura 4 illustra il caso limite in cui l'emergenza è talmente prossima alla destinazione da renderla non sicura (e quindi bloccata).

- **Logica:** Se il Safe Point ricade nel raggio d'azione del disastro (nodi di 2° grado), l'agente lo marca come *Blocked*.
- **UI Feedback:** L'interfaccia segnala l'impossibilità di raggiungere quel punto specifico e l'agente suggerisce automaticamente la navigazione verso il successivo punto sicuro disponibile nella lista ordinata.

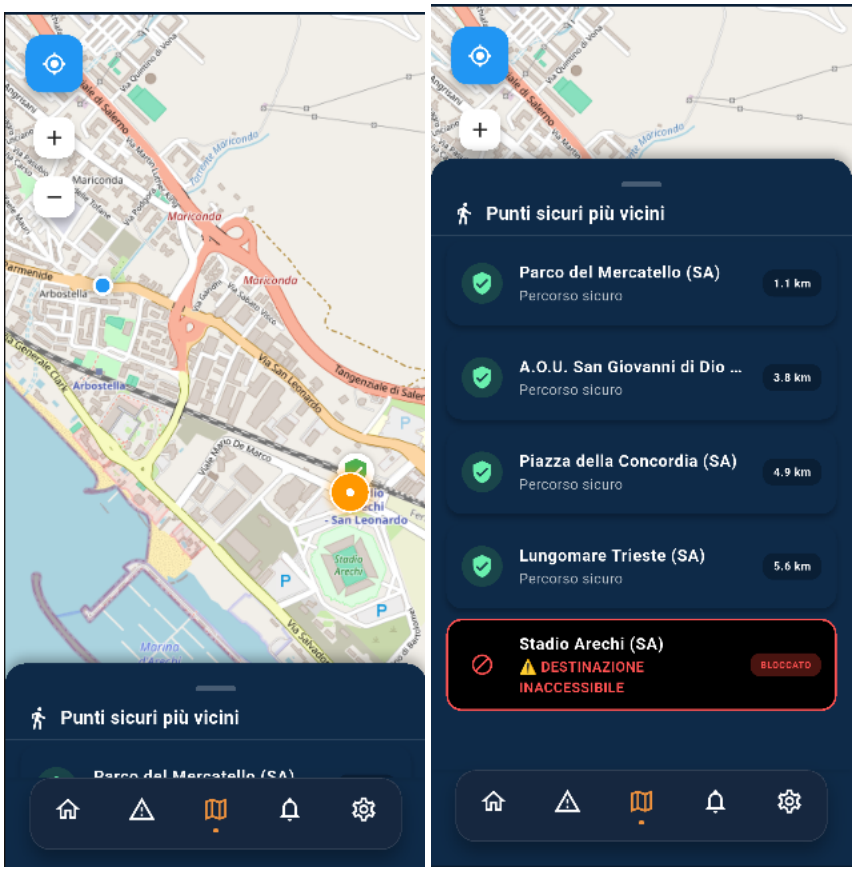

Fig. 9. Gestione di un'emergenza Bloccante.