

Sviluppo

Progetto	TrentoSicura	
Gruppo	T05	
Documento	D3	rev: 1.4

Indice

Scopo del documento	3
User Stories	4
Registrazione Utente	4
Registrazione Ente	5
Login	6
Crea segnalazione	7
Modifica segnalazione	9
Elimina segnalazione	10
Gestione Profilo Utente	10
Cambio password	12
Visualizzazione mappa e lista segnalazioni	14
Elimina profilo	16
Pannello di controllo	17
User Flow	18
Web APIs	19
Implementazione	20

Organizzazione repository	20
Backend	21
Frontend	21
Database	23
Testing	25
Front-End	30
Deployment	35

Scopo del documento

Il presente documento ha lo scopo di descrivere in modo dettagliato l'architettura, il funzionamento e lo sviluppo della web app **Trento Sicura**. In particolare presenta tutti gli artefatti necessari per realizzare i servizi di creazione e gestione delle segnalazioni.

Partendo da una panoramica delle User Stories, il documento prosegue con la presentazione degli User Flow, l'implementazione delle APIs per la gestione dei servizi web e l'organizzazione del codice. Infine, sono fornite informazioni relative ai test, alla distribuzione dell'applicazione, all'interfaccia utente front-end e agli aspetti di sicurezza.

User Stories

Nella sezione seguente vengono riportate le User Stories, una descrizione informale e sintetica delle funzionalità dell'applicazione dal punto di vista dell'utente finale , individuate partendo dai requisiti individuati nel documento **D1 di Trento Sicura**.

Esse comprendono diversi scenari d'uso, tra cui la registrazione e l'autenticazione, la creazione e gestione delle segnalazioni, la visualizzazione degli eventi su mappa e l'interazione con gli stessi.

User Story 1 - Registrazione Utente

La registrazione può avvenire per usufruire esclusivamente del servizio di notifiche personalizzate oppure per ottenere, oltre a questo servizio, la possibilità di creare e inviare segnalazioni all'interno del sistema.

Criteri di accettazione:

- Gli utenti sprovvisti di un profilo devono poter creare un account in qualsiasi momento attraverso l'apposita **pagina di registrazione**.
- Durante la creazione dell'account, il sistema deve verificare **l'autenticità e la sicurezza** delle informazioni fornite dall'utente.
- A registrazione completata, il sistema deve reindirizzare l'utente alla pagina di **login**, invitandolo a inserire le credenziali appena create.
- In caso di errore durante la registrazione, il sistema deve mostrare un **messaggio di avviso** all'utente, specificando la natura del problema e le istruzioni per risolverlo.

Tasks:

1. Implementazione dell'interfaccia di registrazione

Creare una pagina intuitiva che permetta agli utenti di registrarsi tramite **registrazione classica** (con email e password).

2. Verifica utenti già registrati

Implementare un sistema che **controlli** se l'utente è già presente nel database e, in tal caso, lo inviti a effettuare il login invece di una nuova registrazione.

3. Gestione della sicurezza della password

Assicurarsi che il campo della password verifichi il **livello di sicurezza** della stessa (ad esempio, mediante un indicatore di robustezza) e

che venga salvata in modo sicuro nel database utilizzando tecniche di hashing e salting.

User Story 2 - Registrazione Ente

Come **rappresentante di un ente o organizzazione**, voglio potermi registrare sulla web app con un profilo dedicato, così da poter gestire le segnalazioni relative al mio ambito territoriale o tematico.

Criteri di accettazione:

- Nella schermata di registrazione deve essere presente un'**opzione specifica** per potersi registrare come ente.
- Il modulo di registrazione deve richiedere **informazioni aggiuntive** per verificare l'autenticità dell'ente come sito internet, numero di telefono, referente.
- La fase di registrazione deve essere prevista di una **verifica** dell'identità dell'ente.

Tasks:

1. Form di registrazione ente

Creare un'interfaccia dedicata alla registrazione degli enti, distinta da quella riservata agli utenti privati. Il form deve essere accessibile tramite una selezione iniziale che consenta di scegliere tra "Utente" o "Ente" e presentare i campi appropriati in base alla scelta.

2. Campi aggiuntivi per dati ente

Integrare nel form campi specifici obbligatori per la registrazione di un ente come Denominazione dell'Ente, sito web, Codice Fiscale.

3. Gestione backend

Implementare la logica server per ricevere, validare e memorizzare i dati degli enti.

4. Invio di email con conferma registrazione

Il sistema deve inviare una mail di conferma di ricezione della registrazione.

User Story 3 - Login

Come utente già registrato, voglio poter accedere al mio account inserendo le mie credenziali tramite la pagina di login, così da poter usufruire dei servizi personalizzati della web app.

Criteri di accettazione:

- La pagina di login deve essere accessibile dalla schermata di registrazione e dalla homepage
- In caso di credenziali errate il sistema deve fornire un chiaro messaggio di errore
- In caso di accesso riuscito l'utente deve essere reindirizzato alla propria dashboard

Tasks:

1. Creazione pagina login

Progettare e implementare una pagina di login chiara, semplice e responsive. Deve includere i campi per email/username e password, un pulsante per l'invio del form, un link per la registrazione (se l'utente non ha ancora un account). Il design deve essere coerente con lo stile dell'intera web app.

2. Gestione autenticazione

Creare la logica per il controllo delle credenziali fornite. Il backend deve ricevere i dati, verificarli e restituire un token che indichi se l'autenticazione è andata a buon fine. Deve supportare sia gli utenti che gli enti, distinguendo i due ruoli.

3. Messaggi di errore

Gestire messaggi di errore specifici e chiari guidando l'utente nella risoluzione del problema, senza rivelare informazioni sensibili.

4. Reindirizzamento

Se il login ha successo l'utente viene reindirizzato alla dashboard personale, diversa a seconda del ruolo (utente o ente).

5. Persistenza sessione

Una volta effettuato il login, la sessione utente deve rimanere attiva per un certo periodo, oppure finché l'utente non effettua il logout.

User Story 4 - Creazione segnalazione

Come utente registrato e autenticato ho la possibilità di **creare una segnalazione** all'interno della web app.

La segnalazione può essere generata attraverso l'apposita schermata, accessibile sia dalla homepage sia dalla sezione del profilo personale, dopo aver effettuato il login.

Criteri di accettazione:

- L'utente deve fornire almeno le **informazioni essenziali** (titolo e descrizione) affinché la segnalazione possa essere salvata.
- È possibile, sebbene facoltativo ma consigliato, aggiungere la **posizione geografica** e **un'immagine** mediante le apposite sezioni della schermata, che consentono il reindirizzamento a pagine dedicate o tramite un selettore per la posizione attuale.
- L'utente può salvare o eliminare la segnalazione in qualsiasi momento durante la compilazione.
- In caso di inserimento di dati non validi o di file non supportati, il sistema deve fornire un **messaggio di errore** esplicativo, contenente indicazioni per la risoluzione del problema.

Tasks:

1. Interfaccia grafica

La schermata di creazione della segnalazione deve distinguere chiaramente tra **campi obbligatori** e **campi facoltativi**.

Ogni campo di inserimento testo deve includere un placeholder esplicativo, per guidare l'utente nell'inserimento corretto delle informazioni.

La scelta di colori e design coerenti deve facilitare la comprensione delle funzionalità dei pulsanti e delle diverse sezioni.

2. Controllo campi obbligatori

I campi Titolo e Descrizione sono **obbligatori** e costituiscono le informazioni minime per identificare una segnalazione.

Se l'utente tenta di salvare la segnalazione senza aver compilato questi campi, il sistema deve generare un **alert** di avviso.

3. Gestione posizione

L'utente può aggiungere la posizione selezionando il pulsante "Posizione", che reindirizza a una **schermata con una mappa** interattiva.

La selezione della posizione può avvenire sia cliccando un punto sulla mappa o tramite il pulsante per utilizzare la posizione attuale.

4. Gestione immagini

Per aggiungere una foto, l'utente deve cliccare sul pulsante “Foto”, che aprirà una **finestra di selezione** file.

L'utente potrà quindi scegliere un'immagine già presente sul proprio dispositivo.

5. Messaggi di errore

Qualora si verifichi un errore, il sistema deve fornire un **messaggio di avviso** chiaro e dettagliato, spiegando il problema e suggerendo come risolverlo.

6. Salvataggio nel database

Se tutti i dati sono stati inseriti correttamente, l'utente può salvare la segnalazione cliccando sul pulsante “Salva”.

La segnalazione verrà immediatamente registrata nel database e sarà visibile sulla homepage, sia nella lista delle segnalazioni sia sulla mappa interattiva.

User Story 5 - Modifica Segnalazione

Come utente registrato e autenticato, voglio poter modificare una segnalazione che ho creato, così da aggiornare o correggere le informazioni fornite.

Criteri di accettazione:

- L'utente può modificare solo le segnalazioni da lui create
- La schermata di modifica deve mostrare i dati precedenti già compilati
- Dopo le modifiche il sistema deve aggiornare la segnalazione in tempo reale
- In caso di errore il sistema deve fornire un messaggio esplicativo

Tasks:

1. Schermata modifica

Progettare e sviluppare una schermata dedicata alla modifica delle segnalazioni, accessibile solo per l'utente che ha originariamente creato la segnalazione, tramite un pulsante “Modifica” presente nella lista delle proprie segnalazioni.

2. Precompilazione automatica

Tutti i campi del modulo (Titolo, Descrizione, Posizione, Immagine, ecc.) devono essere **precompilati con i dati attuali della segnalazione**.

3. Verifica dei dati obbligatori e file validi

Implementare una logica di validazione simile a quella della creazione:

- Verifica che **Titolo** e **Descrizione** non siano vuoti.
- Controllo del formato e della dimensione dell'immagine (se modificata o aggiunta).
- Controllo dell'integrità della posizione (se modificata).

4. Aggiornamento nel database

Al salvataggio delle modifiche il sistema deve aggiornare le informazioni nel database.

Le modifiche devono essere **visibili in tempo reale** sulla lista delle segnalazioni dell'utente e sulla mappa (se coinvolge la posizione).

User Story 6 - Eliminazione Segnalazione

Come utente registrato, voglio poter eliminare una segnalazione da me creata, così da rimuoverla dalla piattaforma se non più valida.

Criteri di accettazione:

- Solo il creatore della segnalazione o un admin può eliminarla
- Deve essere prevista una finestra di conferma prima dell'eliminazione
- La rimozione deve essere definitiva e immediata

Tasks:

1. UI con pulsante elimina

Integrare nella schermata del profilo personale, accanto a ogni segnalazione, un **pulsante visibile e accessibile** per l'eliminazione. Il design del pulsante deve chiaramente indicare la sua funzione per evitare ambiguità.

2. Finestra di conferma

Prima che la segnalazione venga eliminata, il sistema deve mostrare una finestra di dialogo per la **conferma dell'azione da parte dell'utente**.

La finestra contiene un messaggio chiaro ("Sei sicuro di voler eliminare questa segnalazione? L'azione è irreversibile.").

3. Eliminazione dal database

Una volta confermata l'eliminazione il sistema deve eseguire l'**eliminazione definitiva** della segnalazione dal database.

In caso di errore (es. problemi di connessione o permessi), deve essere mostrato un **messaggio esplicativo** con opzione per riprovare.

User Story 7 - Gestione segnalazioni

Come utente registrato e autenticato che ha precedentemente creato una o più segnalazioni posso vederle tramite una **lista nel profilo personale**, dal quale posso anche modificarle o eliminarle.

Per la visualizzazione delle segnalazioni posso applicare i filtri.

Criteri di accettazione:

- L'utente deve essere **registrato, autenticato** ed aver creato almeno una segnalazione in passato.
- L'utente deve poter accedere a una **lista delle proprie segnalazioni** all'interno della sezione del profilo personale.
- Deve essere possibile **modificare o eliminare** una segnalazione direttamente dalla lista.
- Nel caso in cui l'utente non abbia ancora creato segnalazioni, deve essere mostrato un **messaggio informativo** che lo inviti a crearne una.
- Le modifiche apportate a una segnalazione devono essere salvate correttamente e aggiornate in tempo reale.
- Se l'utente elimina una segnalazione, deve apparire un **messaggio di conferma** prima dell'eliminazione definitiva.
- Le segnalazioni devono essere caricate in modo efficiente e, in caso di molte segnalazioni, deve essere previsto un **sistema di paginazione o caricamento progressivo** per migliorare le prestazioni dell'interfaccia.

Tasks:

1. Interfaccia Grafica

La schermata del profilo dispone di una sezione dedicata per visualizzare l'**elenco delle segnalazioni** e distinguere ogni segnalazione con dettagli essenziali.
Deve essere presente un sistema di filtro e ordinamento con pulsanti intuitivi.
Deve essere previsto un messaggio informativo nel caso in cui l'utente non abbia segnalazioni registrate.

2. Modifica e Eliminazione delle Segnalazioni

Implementazione di un pulsante "**Modifica**" accanto a ogni segnalazione che reindirizzi alla schermata di modifica e consente all'utente di aggiornare i dettagli della segnalazione e salvare le modifiche.
Aggiungere un pulsante "**Elimina**" con una finestra di conferma prima della cancellazione definitiva.
Assicurarsi che le modifiche siano riflesse in tempo reale nell'interfaccia utente.

3. Ottimizzazione della Visualizzazione

Implementare un sistema di **paginazione o caricamento progressivo** per migliorare le prestazioni della lista di segnalazioni.
Ottimizzare il caricamento dei dati per garantire una navigazione fluida e reattiva.

4. Messaggi di Errore e Conferma

Mostrare un messaggio di errore in caso di problemi nel recupero o aggiornamento delle segnalazioni.

Assicurarsi che i messaggi di conferma ed errore siano chiari e guidino l'utente nella **risoluzione** di eventuali problemi.

5. Salvataggio e Aggiornamento nel Database

Implementare l'interazione con il database per recuperare, modificare ed eliminare segnalazioni.

Assicurare che ogni modifica venga correttamente aggiornata e salvata nel database.

Garantire che le eliminazioni siano definitive e che i dati non siano più accessibili dopo la cancellazione.

User Story 8 - Gestione del Profilo Utente

Come utente registrato e autenticato, desidero poter **gestire il mio profilo** personale all'interno della web app, con la possibilità di visualizzare e aggiornare le mie informazioni personali, modificare i dati di accesso e configurare le preferenze relative alle notifiche e alla privacy.

Criteri di Accettazione:

- L'utente deve poter accedere a una sezione dedicata al profilo tramite il menu principale.
- Deve essere possibile **visualizzare i dati personali** registrati, come nome, cognome, email e impostazioni di notifica.
- L'utente deve avere la possibilità di **modificare** le proprie informazioni, inclusi nome, email e password.
- Il sistema deve richiedere conferma prima di salvare modifiche sensibili come la password.
- L'utente deve poter **gestire** le preferenze di notifica, decidendo se ricevere avvisi via email o notifiche push.
- Deve essere presente una funzionalità per **eliminare** definitivamente l'account, previa conferma dell'utente.
- Tutte le modifiche devono essere aggiornate in tempo reale e salvate in modo sicuro nel database.

Tasks:

1. Interfaccia grafica

La schermata per la gestione del profilo utente deve essere intuitiva, strutturata in sezioni chiare per la **modifica dei dati**, la gestione delle **notifiche** e le impostazioni di **sicurezza**.

2. Modifica dei dati

Devono essere presenti **campi editabili** per nome, email e password ai quali vengono implementati controlli per verificare la validità della mail e la robustezza della password.

Viene mostrato a schermo un messaggio di conferma prima del salvataggio delle modifiche sensibili.

3. Preferenze di notifica

Aggiungere un'area dedicata alla configurazione delle notifiche tramite un sistema di **toggle** per attivare o disattivare le notifiche via email o push.

4. Sicurezza e conferma delle modifiche

Implementare un sistema di autenticazione per la modifica di dati critici e prevedere un **sistema di verifica** (es. email di conferma) in caso di modifica della password.

5. Eliminazione account

Aggiungere un'opzione per la cancellazione definitiva dell'account implementato tramite una **doppia conferma** per evitare eliminazioni accidentali.

Assicurarsi che tutti i dati dell'utente vengano rimossi in modo sicuro dal database.

User Story 9 - Visualizzazione Mappa e Lista Segnalazioni

Come utente della web app, desidero poter visualizzare una mappa interattiva che mostri tutte le segnalazioni attive in tempo reale, così da ottenere rapidamente informazioni sugli eventi nella mia area. Inoltre, vorrei poter accedere a una lista dettagliata delle segnalazioni, con la possibilità di applicare filtri per trovare informazioni più facilmente.

Criteri di Accettazione:

La web app deve fornire una mappa interattiva che mostri tutte le segnalazioni attive con marker identificativi.

Deve essere possibile fare clic su un marker per visualizzare una finestra con dettagli della segnalazione.

L'utente deve poter accedere a una lista delle segnalazioni, ordinata cronologicamente e filtrabile.

Devono essere disponibili filtri avanzati per cercare segnalazioni per data e categoria.

Le segnalazioni devono essere aggiornate in tempo reale senza necessità di ricaricare la pagina.

L'interfaccia deve essere ottimizzata per dispositivi mobili e desktop, consentendo una navigazione fluida sia sulla mappa che sulla lista delle segnalazioni.

Tasks:

1. Implementazione della mappa interattiva

Integrare un servizio di mappe (OpenStreetMap) con visualizzazione di marker per ogni segnalazione attiva.

Aggiungere popup informativi con dettagli sulla segnalazione quando si clicca su un marker contenenti titolo e descrizione.

2. Creazione della lista Segnalazioni

Strutturare un elenco ordinato con titolo, data e descrizione di ogni segnalazione e implementare un meccanismo di caricamento dinamico per migliorare la velocità di navigazione.

3. Implementazione dei Filtri

Consentire il filtraggio per tipologia di segnalazione (tramite varie categorie preimpostate) e data.

Implementare un sistema di ricerca testuale per trovare segnalazioni specifiche.

4. Aggiornamento in tempo reale

Integrare WebSocket o polling per ricevere aggiornamenti automatici sulle nuove segnalazioni.

Assicurarsi che la mappa e la lista riflettano sempre le informazioni più aggiornate.

5. Ottimizzazione per dispositivi

Adattare la visualizzazione della mappa per schermi di diverse dimensioni.

Implementare un'interfaccia responsive che permetta di passare facilmente dalla mappa alla lista delle segnalazioni.

User Story 10 - Cambio Password

Come utente registrato, voglio poter cambiare la mia password in qualsiasi momento, così da garantire la sicurezza del mio account.

Criteri di accettazione:

- L'utente deve inserire la nuova password dopo aver fatto l'accesso
- La nuova password deve rispettare gli stessi criteri di sicurezza della creazione della password durante la registrazione

Tasks:

1. Interfaccia cambio password

Progettare e implementare una **schermata dedicata** alla modifica della password all'interno della sezione "Gestione Profilo".

Questa interfaccia deve includere un campo per l'inserimento della **nuova password**.

2. Validazioni e corrispondenza

Implementare controlli lato client e server per garantire che la nuova password rispetti i **criteri di sicurezza** definiti alla registrazione e che le due nuove password **corrispondano perfettamente**.

3. Autenticazione

Prima di procedere con l'aggiornamento della password il sistema deve **verificare l'identità dell'utente**, controllando che la **password attuale inserita sia corretta** (match con quella nel database, usando hashing).

Solo in caso di autenticazione riuscita si procede all'**aggiornamento sicuro** della nuova password nel database.

User Story 11 - Eliminazione Profilo

Come utente registrato, voglio poter eliminare definitivamente il mio profilo, così da revocare la mia presenza e i miei dati dal sistema.

Criteri di accettazione:

- Il sistema deve richiedere una doppia conferma prima dell'eliminazione
- L'eliminazione deve comportare la rimozione di tutte le segnalazioni e dati personali
- Deve essere inviata una conferma email dell'eliminazione
- Dopo l'eliminazione, l'utente deve essere disconnesso e impossibilitato ad accedere

Tasks:

1. Pulsante “Elimina Profilo”

Aggiungere all'interno della sezione “Impostazioni Account” un pulsante visibile ma non invasivo, denominato **“Elimina Profilo”**.

2. Doppia conferma

Implementare un meccanismo di conferma tramite finestra popup che riassume le conseguenze dell'eliminazione del profilo.

3. Cancellazione dal database

Una volta confermata l'azione:

- Il sistema deve **eliminare tutte le informazioni personali** dell'utente (nome, email, preferenze, ecc.).
- Tutte le **segnalazioni associate** al profilo devono essere cancellate o, se necessario, rese anonime.
- Eventuali file o immagini collegati all'utente devono essere rimossi dallo storage.
- L'eliminazione deve essere **definitiva**, conforme alle normative sulla protezione dei dati (es. GDPR).

4. Disconnessione e blocco accesso

Subito dopo l'eliminazione l'utente deve essere automaticamente disconnesso dalla sessione e qualsiasi tentativo successivo di login con le credenziali eliminate deve fallire, mostrando un messaggio come: “Account inesistente o eliminato.”

User Story 12 - Pannello di controllo

Come admin registrato, voglio poter gestire le segnalazioni degli utenti per poter cambiare stato, modificarle o eliminarle

Criteri di accettazione:

- Il sistema deve verificare l'identità dell'admin prima di permettere l'accesso alla sezione di gestione delle segnalazioni.
- Deve essere presente una schermata dedicata dove siano elencate **tutte le segnalazioni e gli utenti** collegati.
- Deve essere possibile **filtrare le segnalazioni per categoria**.
- Ogni segnalazione deve includere i dati dell'utente o dell'organizzazione che l'ha creata.

Tasks:

1. Autenticazione admin

Implementare middleware di verifica dei privilegi admin lato backend per impedire l'accesso alla schermata di gestione a utenti non autorizzati.

2. Interfaccia Pannello di Controllo

- Progettare e sviluppare una schermata “Pannello di Controllo” accessibile solo da admin.
- Mostrare un elenco tabellare di tutte le segnalazioni con dettagli come titolo, descrizione, categoria, stato e dati utente.
- Includere opzioni di ricerca e filtro per categoria.

3. Funzionalità di Gestione Segnalazioni

- Implementare pulsanti o azioni per:
- Cambiare lo stato di una segnalazione.
- Modificare i dettagli di una segnalazione (titolo, descrizione, categoria).
- Eliminare una segnalazione.
- Gestire le operazioni in modo sicuro lato backend con rotte protette.

4. Visualizzazione Dati Utente

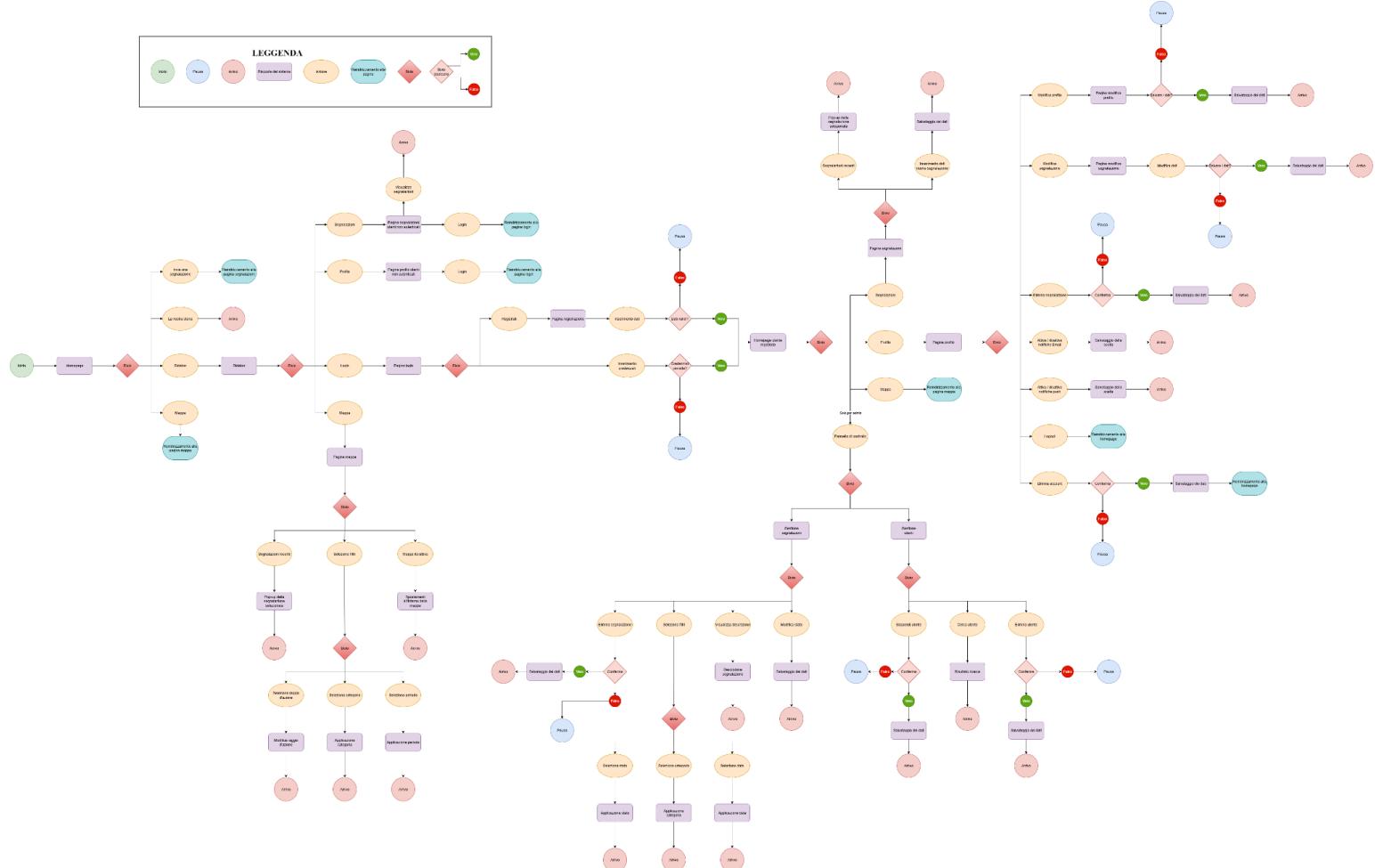
Mostrare per ogni segnalazione il nome utente o organizzazione, eventuali contatti utili (es. email o telefono se previsto). Assicurarsi che i dati siano corretti e aggiornati.

User Flow

La presente sezione fornisce una descrizione dettagliata dello user flow dell'applicazione, illustrando in maniera chiara e strutturata le principali interazioni che l'utente può effettuare all'interno dell'interfaccia. L'obiettivo è quello di rappresentare il percorso logico e operativo che guida l'utente nell'utilizzo delle diverse funzionalità disponibili, evidenziando le azioni consentite, le scelte possibili e le relative conseguenze all'interno del sistema.

A supporto della comprensione del diagramma, viene inoltre fornita una legenda esplicativa che descrive i diversi elementi grafici impiegati nella rappresentazione dello user flow, specificando il significato di simboli, colori e connettori utilizzati per rappresentare stati, transizioni e interazioni utente-sistema.

Al seguente link è possibile visualizzare lo schema in dettaglio: [UserFlow](#)



Web APIs

Le API RESTful della presente applicazione sono state documentate seguendo lo **standard OpenAPI 3.0**, al fine di garantire chiarezza, interoperabilità e facilità di integrazione per gli sviluppatori terzi.

La documentazione è stata generata utilizzando **Swagger UI**, che consente di visualizzare, esplorare e testare le API in modo interattivo direttamente da browser.

È possibile consultare la documentazione completa delle API al seguente indirizzo:

[**Vai alla documentazione interattiva delle API**](#)

La specifica completa delle API è disponibile nel file YAML api-docs.yaml, accessibile direttamente al link seguente:

[**Scarica o visualizza il file OpenAPI**](#)

Implementazione

L'applicazione è stata sviluppata adottando un moderno stack tecnologico full-stack, costituito da:

- **Node.js** per il backend, scelto per la sua efficienza, scalabilità e l'ampio ecosistema di librerie;
- **Vue.js** per il frontend, selezionato per la sua semplicità, reattività e facilità di integrazione con componenti dinamici;
- **MongoDB** come database NoSQL, ideale per la gestione flessibile e scalabile di dati non strutturati.

Queste tecnologie sono state selezionate con l'obiettivo di garantire **chiarezza progettuale, rapidità nello sviluppo** e **manutenibilità del codice** nel lungo periodo. La combinazione di questi strumenti consente di realizzare un'applicazione performante, modulare e facilmente estendibile.

Organizzazione repository

Il codice sorgente del progetto è disponibile su GitHub al seguente indirizzo: [TrentoSicura](#). Tutti i contributi sono tracciati e documentati nella cronologia dei commit, consultabile attraverso la sezione *history* di ciascun repository.

Al fine di migliorare la gestione dello sviluppo, favorire la collaborazione tra i membri del team e mantenere una struttura modulare, è stata creata un'apposita **organizzazione GitHub**, all'interno della quale il progetto è stato suddiviso in **tre repository distinti**:

- **Frontend**: contiene tutto il codice relativo all'interfaccia utente, sviluppata in Vue.js;
- **Backend**: include la logica server-side, implementata in Node.js e Express, nonché le API REST;
- **Deliverables**: raccoglie la documentazione tecnica, le specifiche del progetto, i file OpenAPI e gli altri materiali consegnabili.

Questa struttura consente di separare logicamente le componenti dell'applicazione, semplificando lo sviluppo parallelo, il versionamento e il deploy continuo.

BackEnd

```
Backend/
  controllers/
    └── OrgController.js
    └── ReportController.js
    └── UserController.js

  middleware/
    └── authMiddleware.js

  models/
    └── Org.js
    └── Report.js
    └── User.js

  routes/
    └── OrgRoutes.js
    └── ReportRoutes.js
    └── UserRoutes.js

  utils/
    └── db.js

  .env
  .gitignore
  api-docs.yaml
  package.json
  README.md
  server.js
```

- **controllers/**: contiene la logica per ciascuna entità (Users, Reports, Orgs)
- **routes/**: definisce le rotte Express che collegano gli endpoint HTTP ai controller
- **models/**: modelli Mongoose per la persistenza su MongoDB
- **middleware/**: middleware per autenticazione e autorizzazione
- **utils/**: funzionalità per la connessione al database ([db.js](#))
- **api-docs.yaml**: definizione delle API in formato OpenAPI 3.0
- **server.js**: entry point dell'applicazione

FrontEnd

```
Frontend/
  public/
    └── index.html

  src/
    ├── assets/
    │   └── logo.svg
    ├── components/
    │   └── Header.vue
    │   └── Sidebar.vue
    ├── router/
    │   └── index.js
    ├── services/
    │   └── authHeader.js
    │   └── authService.js
    │   └── userService.js
    ├── store/
    │   └── index.js
    ├── views/
    │   └── Dashboard.vue
    │   └── Login.vue
    │   └── Register.vue
    └── App.vue
    └── main.js

  .gitignore
  package.json
  README.md
  vite.config.js
```

- **public/**: contiene l'HTML statico principale, usato da Vite per il mounting dell'app Vue.
- **src/components/**: componenti riutilizzabili dell'interfaccia utente.
- **src/views/**: pagine principali (route-level) della SPA.
- **src/router/**: gestione delle rotte tramite Vue Router.
- **src/services/**: moduli per interagire con le API (autenticazione, gestione utenti, ecc.).
- **src/store/**: stato globale dell'app, gestito tramite Vuex.
- **App.vue e main.js**: punto di avvio e componente radice dell'applicazione.

Dependencies Backend:

- **bcryptjs (2.4.3):** Hashing sicuro delle passwords
- **cookie-parser (2.4.3):** Parsing dei cookie HTTP
- **cors (2.8.5):** Abilitazione CORS (Cross-Origin Resource Sharing)
- **dotenv (16.3.1):** Gestione delle variabili d'ambiente
- **express (4.18.2):** Web framework per Node.js
- **jsonwebtoken (9.0.2):** Autenticazione tramite JWT
- **mongoose (8.3.3):** ODM per MongoDB
- **swagger-ui-express (4.7.2):** Interfaccia Swagger per le API
- **yamljs (0.3.0):** Parsing file YAML

Dependencies Frontend:

- **axios (1.6.8):** Client HTTP per chiamate API
- **core-js (3.9.1):** Polyfill per compatibilità JS
- **pinia (2.1.7):** Nuovo store system
- **vue (3.4.21):** Framework frontend
- **vue-router (4.3.0):** Gestione delle rotte SPA

Database:

ReportSchema

```
const ReportSchema = new mongoose.Schema({
  title: { type: String, required: true },
  description: { type: String, required: true },
  user: { type: mongoose.Schema.Types.ObjectId, ref: 'User', required: true },
  location: { lat: { type: Number, required: false }, lng: { type: Number, required: false } },
  photo: { type: String, required: false },
  tags: { type: [String], required: false },
}, {
  timestamps: true,
});
```

- title / description: titolo e descrizione della segnalazione (obbligatori).
- user: riferimento all'utente che ha creato il report (ref: 'User').
- location: coordinate geografiche opzionali (lat, lng).
- photo: URL o path a una foto associata.
- tags: lista di parole chiave per classificare il report.

OrgSchema

```
const OrgSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  phone: { type: Number, required: true, unique: true },
  indirizzo: { type: String, required: true },
  segnalazioni: [{ type: mongoose.Schema.Types.ObjectId, ref: "Report", default: [] }],
}, {
  timestamps: true,
});
```

- username, email, password: credenziali di accesso.
- phone, indirizzo: dati di contatto.
- segnalazioni: array di ObjectId riferiti ai report ricevuti o gestiti.

UserSchema

```
const UserSchema = new mongoose.Schema({
  username: { type: String, required: true, unique: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  phone: { type: Number, required: false },
  posizione: { type: Boolean, required: false, default: false },
  segnalazioni: [{ type: mongoose.Schema.Types.ObjectId, ref: "Report", default: [] }],
}, {
  timestamps: true,
});
```

- username, email, password: credenziali (tutti univoci).
- phone: numero di telefono (facoltativo).
- posizione: booleano per indicare la disponibilità di posizione geografica.
- segnalazioni: array di ObjectId riferiti ai report creati dall'utente.

Testing

Registrazione

N°	Descrizione	Test Data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
1	Creazione account	tutti i dati richiesti sono corretti			creazione corretta account	= risultato atteso	
1.1	Creazione account con email già in uso	username già in uso	Dati dell'account già presenti nel sistema	1	messaggio di errore con username già presente	= risultato atteso	
1.2	Creazione account non specificando un campo	email vuota			messaggio di errore inserire tutti i campi	= risultato atteso	
1.3	Creazione account con password insicura	'ciao00' come password			messaggio di errore password insicura	= risultato atteso	

Login

N°	Descrizione	Test Data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
2	Login in modo corretto	username e email presenti nel sistema	Dati dell'account già presenti nel sistema	1	login corretto	= risultato atteso	
2.1	Login con credenziali non esistenti	email non presente nel sistema			messaggio di errore con email non presente	= risultato atteso	
2.2	Login con credenziali sbagliate	password errata			messaggio di errore password errata	= risultato atteso	

Modifica password

N°	Descrizione	Test Data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
3	Nuova password corretta	password rispetta i requisiti	Account esistente e loggato	1 2	aggiornamento corretto	= risultato atteso	
3.1	Nuova password non sicura	password non rispetta i requisiti			messaggio password insicura	= risultato atteso	
3.2	Nuova password uguale a quella vecchia	password uguale alla vecchia			messaggio password uguale	La password viene cambiata ugualmente	Per risalire a quella vecchia bisognerebbe usare una funzione crittografica invece che una di hash rendendo il database poco sicuro

Creazione segnalazione

N°	Descrizione	Test Data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
4	Creazione segnalazione corretta	Tutti i campi sono compilati correttamente	Account esistente e loggato	1 2	Segnalazione creata correttamente	= risultato atteso	
4.1	Segnalazione senza un campo obbligatorio	Titolo non inserito			Messaggio di errore	= risultato atteso	
4.2	Segnalazione con posizione errata	Posizione inserita manualmente			Messaggio di errore	= risultato atteso	la posizione va cliccata sulla mappa

Modifica segnalazione

N°	Descrizione	Test Data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
5	Tutti i campi sono corretti	Viene aggiornato il titolo	Account esistente e loggato come proprietario della segnalazione	1 2 4	aggiornamento corretto	= risultato atteso	
5.1	Un campo è stato lasciato vuoto	La descrizione è lasciata vuota			messaggio di errore	= risultato atteso	
5.2	Niente viene cambiato	Segnalazione invariata			segnalazione salvata come prima	= risultato atteso	

JEST

È possibile eseguire i test automatizzati definiti nel progetto utilizzando il comando:

npm test

Questo comando avvia Jest, il framework di testing adottato nel progetto, ed esegue tutti i test presenti nei file [user.test.js](#), [org.test.js](#), [report.test.js](#) presenti nella cartella test del Backend.

I test effettuati su Jest riguardano:

Registrazione Utente e Organizzazione

```
describe('POST /api/users', () => {
  it('should register a user successfully', async () => {
    const res = await request(app).post('/api/users').send({
      email: 'test@example.com',
      username: 'testuser',
      password: 'Password1'
    });
    expect(res.statusCode).toBe(201);
  });

  it('should fail with existing email', async () => {
    await User.create({ email: 'test@example.com', username: 'existing', password: 'Password1' });

    const res = await request(app).post('/api/users').send({
      email: 'test@example.com',
      username: 'newuser',
      password: 'Password1'
    });
    expect(res.statusCode).toBe(400);
    expect(res.body.message).toMatch(/already exists/i);
  });

  it('should fail with no email', async () => {
    const res = await request(app).post('/api/users').send({
      username: 'newuser',
      password: 'Password1'
    });
    expect(res.statusCode).toBe(500);
  });

  it('should fail with weak password', async () => {
    const res = await request(app).post('/api/users').send({
      email: 'new@example.com',
      username: 'weakpass',
      password: 'ciao00'
    });
    expect(res.statusCode).toBe(400);
  });
});
```

- Registrazione utente correttamente (201)

- Registrazione con email esistente (400)

- Registrazione senza inserimento email (500)

- Registrazione con password non sicura (400)

Login Utente e Organizzazione

```
describe('POST /api/users/session', () => {
  beforeEach(async () => {
    await request(app).post('/api/users').send({
      email: 'login@example.com',
      username: 'loginuser',
      password: 'Password1'
    });
  });

  it('should login successfully', async () => {
    const res = await request(app).post('/api/users/session').send({
      email: 'login@example.com',
      password: 'Password1'
    });
    expect(res.statusCode).toBe(200);
  });

  it('should fail with wrong password', async () => {
    const res = await request(app).post('/api/users/session').send({
      email: 'login@example.com',
      password: 'WrongPass'
    });
    expect(res.statusCode).toBe(400);
  });

  it('should fail with nonexistent email', async () => {
    const res = await request(app).post('/api/users/session').send({
      email: 'noexist@example.com',
      password: 'Password1'
    });
    expect(res.statusCode).toBe(400);
  });
});
```

- Login corretto (200)

- Login con password sbagliata (400)

- Login con email non esistente (400)

Creazione segnalazione

```
it('should create a report', async () => {
  const res = await request(app)
    .post('/api/reports')
    .set('Cookie', [`token=${token}`])
    .send({
      reportData: {
        title: 'Test Report',
        description: 'Description here',
      }
    });
  expect(res.statusCode).toBe(201);
  expect(res.body).toHaveProperty('message', 'Report created successfully');
  expect(res.body).toHaveProperty('report');
});

it('should fail when title is missing', async () => {
  const res = await request(app)
    .post('/api/reports')
    .set('Cookie', [`token=${token}`])
    .send({
      reportData: {
        description: 'Description here',
      }
    });
  expect(res.statusCode).toBe(500);
  expect(res.body).toHaveProperty('error');
});

it('should fail when coordinates are invalid', async () => {
  const res = await request(app)
    .post('/api/reports')
    .set('Cookie', [`token=${token}`])
    .send({
      reportData: {
        title: 'Test Report',
        description: 'Description here',
        location: {
          lat: 'invalid',
          lng: null
        }
      }
    });
  expect(res.statusCode).toBe(500);
  expect(res.body).toHaveProperty('error');
});
```

- Creazione Report corretto (201)

- Creazione report senza titolo (500)

- Creazione report con posizione invalida (500)

Front-End

L'interfaccia frontend dell'applicazione è organizzata in sezioni tematiche, navigabili attraverso una barra laterale (sidebar) che consente un accesso rapido e intuitivo alle diverse funzionalità.

Alcune pagine sono liberamente accessibili senza necessità di autenticazione. Tra queste rientrano:

- **La mappa delle segnalazioni**, che consente la consultazione generale delle segnalazioni geolocalizzate;
- **La schermata introduttiva**, che fornisce una panoramica sull'utilizzo e gli obiettivi della webapp.

Le restanti sezioni dell'applicazione sono accessibili esclusivamente previa autenticazione e la loro visibilità e disponibilità variano in base al ruolo dell'utente.

Nello specifico:

- **Gli utenti registrati** possono accedere alla propria area personale, dalla quale è possibile:
 - Visualizzare e modificare i dati del proprio profilo;
 - Gestire le proprie segnalazioni, ovvero creare nuove segnalazioni, modificarne i contenuti oppure eliminarle.
- **Le organizzazioni** (o enti autorizzati) hanno accesso a funzionalità avanzate per la gestione delle segnalazioni ricadenti nel proprio ambito di competenza. Tali funzionalità includono:
 - La modifica del livello di priorità associato a ciascuna segnalazione;
 - L'eliminazione delle segnalazioni non più rilevanti;
 - L'aggiornamento dello stato delle segnalazioni, con la possibilità di contrassegnarle come "completate" una volta risolte.

Questa differenziazione dei permessi garantisce un'esperienza d'uso coerente con il ruolo dell'utente e assicura che le operazioni critiche siano riservate a soggetti autorizzati.

Schermata Home

The screenshot shows the homepage of the TrentoSicura platform. At the top left is a navigation bar with links: Home, Mappa, Segnalazioni, Profilo, and Login. The main title "TrentoSicura" is centered at the top in a large, bold font. Below it is a subtitle: "La piattaforma digitale che rivoluziona la sicurezza urbana attraverso la partecipazione attiva dei cittadini". Two buttons are present: "ESPLORA LA MAPPA" and "INVIA UNA SEGNALAZIONE". The central section is titled "Il Problema che Risolviamo" and contains two boxes: "Sfide della Sicurezza Urbana" (Challenges of Urban Safety) and "La Nostra Soluzione" (Our Solution). The "Sfide" box lists issues like ineffective communication between citizens and authorities, long response times for safety reports, lack of transparency on territorial problems, and difficulty monitoring the state of problems. The "Soluzione" box lists benefits such as a unified digital reporting platform, precise geolocation and photographic documentation, transparent tracking of report status, and active community involvement for a safer city. Below this is a section titled "Come Funziona" (How It Works) with three cards: "Mappa Interattiva" (Interactive Map), "Segnalazioni Rapide" (Quick Reports), and "Comunità Attiva" (Active Community).

Mappa

This screenshot shows the map interface. On the left is a sidebar with the same navigation links as the home page. The main area features a map of a town or city with several locations marked by blue and red pins. A large blue circle indicates the search radius, set to 1600m. To the right of the map is a "Segnalazioni Recenti" (Recent Reports) section showing two recent reports: "Segnalazione di prova frontend" (circa 3 ore ago) and "Segnalazione di prova frontend" (circa 2 ore ago). Both reports are categorized under "Sicurezza". The map also includes labels for various streets and landmarks.

Segnalazioni

The screenshot shows a mobile application interface for reporting incidents. On the left is a vertical navigation bar with icons for Home, Mappa, Segnalazioni (highlighted in blue), Profilo, and Login. The main content area has a blue header with the title "Ultime Segnalazioni degli Utenti". Below the header are three report cards:

- Grave incidente in Via Rosmini 43** (Traffic, 15 luglio 2025, RESOLVED): A camion e una macchina hanno fatto un frontale, la polizia è in arrivo. (User)
- Grave crepa sul ponte ferroviario** (Safety, 15 luglio 2025, PENDING): Ponte pedonale in Via Caduti di Nassiria ha una crepa che sembra strutturale. (User)
- Immondizia sparsa per tutto il parco** (Environment, 15 luglio 2025, IN_PROGRESS): Il parco di piazza Venezia ha immondizia sparsa per tutto il parco a causa dei cestini rotti. (User)

Login

The screenshot shows a login screen for the "TrentoSicura" application. On the left is a vertical navigation bar with icons for Home, Mappa, Segnalazioni (highlighted in blue), Profilo, and Login. The main content area features a blue background with white text and graphics. At the top center is the "TrentoSicura" logo. Below it is the text "Gruppo T05:" followed by "HaoJie Z. - Giacomo P. - Luca B.". To the left of the form is the "UNIVERSITÀ DI TRENTO" logo. The login form itself has fields for "E-mail utente" and "Password", a "Accedi" button, a "Registra" button, and a "G" button.

Registrazione Utente

Home
Mappa
Segnalazioni
Profilo
Login

Crea il tuo account

Selezione il tipo di account

Privato
Cittadino

Ente Pubblico
Organizzazione

Nome utente
Es. Mario Rossi

Email
Es. mario@email.com

Password
.....

Conferma Password
.....

Consenti la geolocalizzazione per segnalazioni più precise

Registrati come Cittadino

Hai già un account? [Accedi](#)

Registrazione Ente

Home
Mappa
Segnalazioni
Profilo
Login

Crea il tuo account

Selezione il tipo di account

Privato
Cittadino

Ente Pubblico
Organizzazione

Nome utente
Es. Comune di Trento

Email
Es. info@comune.trento.it

Password
.....

Conferma Password
.....

Telefono *
Es. 0461123456

Indirizzo *
Es. Via Roma 1, 38122 Trento TN

Descrizione *
Descrivi di cosa si occupa la tua organizzazione...

Registrati come Ente Pubblico

Hai già un account? [Accedi](#)

Pagina Profilo

The screenshot shows the 'Profilo Utente' (User Profile) page. On the left, a sidebar menu includes Home, Mappa, Segnalazioni, Profilo (selected), and Pannello di Controllo. The main area displays the user's profile: Trentino Trasporti (admin@trasporti.com), registered on 15/07/2025. Below this is a section for 'Attività Recenti' (Recent Activity) showing a single entry: 'Segnalazione di prova frontend' (Test signalization) from 16/07/2025, with 'Modifica' and 'Elimina' buttons. The 'Impostazioni' (Settings) section includes toggles for 'Notifiche Email' and 'Notifiche Push', a 'Logout' button, and a red 'ELIMINA ACCOUNT' button for account deletion.

Pannello di Controllo

The screenshot shows the 'Pannello di Controllo' (Control Panel) for managing the organization. The sidebar menu is identical to the profile page. The main area features a dashboard with four cards: 'Utenti Registrati' (5 users), 'Segnalazioni Totali' (4 total reports), 'In Attesa' (1 pending report), and 'In Corso' (2 ongoing reports). Below this is a 'Gestione Segnalazioni' (Report Management) table. The table has columns: TITOLO, UTENTE, CATEGORIA, STATO, DESCRIZIONE, and AZIONI. It lists four reports: 'Grave incidente in Via Rosmini 43' (User1, Traffico, Risolto, View, Elimina), 'Grave crepa sul ponte ferroviario' (User1, Sicurezza, In Attesa, View, Elimina), 'Immondizia sparsa per tutto il parco' (User2, Ambiente, In Corso, View, Elimina), and 'impalcatura pericolante' (User2, Sicurezza, In Corso, View, Elimina).

Deployment

L'applicazione è strutturata secondo un'architettura client-server ed è composta da due componenti principali: **frontend** e **backend**, entrambi ospitati sulla piattaforma **Render**, che ne garantisce la disponibilità, la scalabilità e la gestione semplificata del deploy.

Per quanto riguarda GitHub, la piattaforma utilizzata per salvare e rendere accessibili i file di implementazione, è stata adottata una **strategia master only**. In questo modo tutto lo sviluppo viene gestito direttamente sul branch principale, garantendo che il codice sia sempre aggiornato e facilmente consultabile da tutti i collaboratori.

Backend

Il backend dell'applicazione si occupa dell'interazione con il database e dell'esposizione degli endpoint REST necessari per le funzionalità offerte dal sistema.

È raggiungibile al seguente indirizzo:

<https://backend-3801.onrender.com>

La documentazione completa e aggiornata degli endpoint disponibili (comprensiva di metodi, parametri, risposte e codici di stato) è consultabile tramite un'interfaccia interattiva all'indirizzo:

Frontend

Il frontend costituisce l'interfaccia utente dell'applicazione, accessibile tramite browser web. È sviluppato per garantire una navigazione fluida e un'interazione intuitiva con i dati esposti dal backend.

È disponibile al seguente indirizzo:

<https://frontend-cb2s.onrender.com>

Accesso e Tipologie di Account

Per testare e valutare le funzionalità dell'applicazione, sono disponibili due tipologie di account preconfigurati, ciascuno con permessi e funzionalità differenti:

- **Account Utente**

Questo tipo di account consente l'accesso alle funzionalità base, tra

cui la creazione, modifica e cancellazione delle proprie segnalazioni, nonché la gestione del profilo personale.

Credenziali di prova:

- **Email:** utente1@gmail.com
- **Password:** Ciao00

Credenziali di prova:

- **Email:** utente2@gmail.com
- **Password:** Ciao00

● **Account Organizzazione**

Riservato a enti e organizzazioni autorizzate, questo account permette la gestione avanzata delle segnalazioni, tra cui il cambio di priorità, l'eliminazione e la marcatura come completate.

Credenziali di prova:

- **Email:** adminTrasporti@gmail.com
- **Password:** Ciao00

Entrambe le tipologie di account possono essere utilizzate per esplorare l'applicazione in ambiente di test e verificarne il comportamento in scenari realistici.