

D3 - Documento di sviluppo dell'applicazione

Indice

Scopo del documento	2
1. User Stories	3
1. Facilità d'uso	3
2. Login e registrazione	4
3. Visualizzazione storico emergenze	5
4. Filtraggio e ricerca emergenze	5
5. Dashboard con emergenze in corso	6
6. Visualizzazione mappa	7
7. Comunicazioni operatore	8
8. Invio segnalazioni utenti	9
9. Gestione segnalazioni per l'operatore	10
10. Gestione segnalazioni per l'utente	10
11. Gestione profilo personale	11
12. Gestione utenti per l'operatore	12
2. User Flow	13
3. Web APIs	14
4. Implementazione	27
4.1 Organizzazione del codice	27
4.2 Branching strategy e organizzazione	28
4.3 Dependencies	28
4.4 Database	29
4.5 Testing	31
5. Frontend	36
6. Deployment	42

Informazioni sul documento

Deliverable n.	3
Versione	2.0.4
Descrizione	Documento di specifiche per lo sviluppo dell'applicazione descrivendo funzionalità di visualizzazione della mappa, gestione delle emergenze, User Stories, User Flows, API, organizzazione del codice, modello dati, testing, front-end e

	aspetti di deployment.
--	------------------------

Scopo del documento

Il presente documento riporta tutte le specifiche riguardanti lo sviluppo di una demo dell'applicazione Ocio. Le funzionalità implementate comprendono la visualizzazione della mappa e delle emergenze in corso, la ricerca delle emergenze e la gestione delle stesse da parte dell'operatore comunale. Il documento parte dalla descrizione delle User Stories, prosegue con gli User Flows e la presentazione delle API; vengono poi presentati l'organizzazione del codice e delle relative cartelle, il modello dati, e ulteriori informazioni su testing. Infine, una sezione dedicata al front-end e una agli aspetti di deployment.

1. User Stories

Legenda:

Importanza: Valori da 1 a 5	1 - Essenziale 2 - Necessario 3 - Importante 4 - Utile 5 - Facoltativo
Story points (carico di lavoro e complessità): Valori da XS a XL	XS S M L XL

1. Facilità d'uso

Importanza	3 - Importante
Story points	M

Descrizione:

Come utente, voglio un'interfaccia intuitiva e accessibile, in modo da poter utilizzare i servizi rapidamente e senza difficoltà

Criteri di accettazione:

- L'interfaccia deve essere chiara e intuitiva
- Le azioni principali dell'utente devono essere accessibili in 3 click o meno, come:
 - effettuare una segnalazione e visualizzare la mappa per il cittadino
 - pubblicare e rimuovere una comunicazione per l'operatore comunale
- I pulsanti, le icone e le etichette sono comprensibili e coerenti

Task:

1. Testare che le funzionalità siano accessibili in modo semplice e intuitivo
2. Progettare un'interfaccia utente chiara e intuitiva
 1. Strutturare la homepage con accesso rapido alle funzioni principali
 2. Utilizzare icone riconoscibili e testi descrittivi per le azioni principali
3. Implementare feedback visivo e messaggi di conferma
 1. aggiungere indicatori visivi per azioni come l'invio di segnalazioni o di comunicazioni

4. Migliorare l'accessibilità
 1. assicurarsi che i colori e i contrasti siano adatti a utenti con difficoltà visive

2. Login e registrazione

Importanza	1 - Essenziale
Story points	L

Descrizione:

Come cittadino o operatore comunale, voglio poter effettuare la registrazione e il login in modo semplice e sicuro, così da poter accedere rapidamente alle funzionalità dell'app

Criteri di accettazione:

- L'utente può scegliere tra registrazione/login con credenziali locali o autenticazione tramite Google
- Il sistema deve riconoscere l'utente già registrato e reindirizzarlo alla pagina corretta
- Se l'utente non è registrato o le credenziali non sono corrette, il sistema deve fornire dei messaggi d'errore adeguati
- Dopo un login riuscito, l'utente viene reindirizzato alla dashboard corrispondente al suo ruolo

Task:

1. Integrare entrambe le opzioni di login (credenziali locali e Auth di Google) nella stessa pagina
2. Configurare il flusso decisionale per il login
 1. Implementare la logica che distingue quale metodo di autenticazione (locale o Google) è stato scelto e avviare il processo corretto.
3. Gestire il routing dopo il login per entrambi i ruoli (utente e operatore)
 1. Assicurarsi che dopo l'autenticazione l'utente venga reindirizzato alla pagina di destinazione corretta in base al suo ruolo (utente o operatore).
4. Testare entrambe le modalità di accesso
 1. Testare l'autenticazione con credenziali locali per utenti e operatori.
 2. Testare l'autenticazione Google per utenti e operatori.
 3. Verificare che entrambe le opzioni funzionino correttamente e conducano alla dashboard giusta.
5. Implementare messaggi d'errore distinti per ciascun metodo
 1. Creare messaggi di errore specifici per problemi relativi al login con credenziali locali o tramite Google.

3. Visualizzazione storico emergenze

Importanza	2 - Necessario
Story points	M

Descrizione:

Come utente voglio poter visualizzare l'elenco di tutte le emergenze, sia attuali che passate (in corso e terminate) con i rispettivi dettagli principali e la possibilità di visualizzare anche i dettagli completi

Criteri di accettazione:

- Gli utenti possono visualizzare un elenco delle emergenze
 - Le emergenze sono disposte in ordine cronologico, dal più recente al meno recente come data di pubblicazione
- I dettagli di ogni emergenza sono consultabili cliccando sul pulsante "Espandi" corrispettivo

Task:

1. Creare l'interfaccia per lo storico delle emergenze
 1. Progettare una pagina dedicata con la lista delle emergenze
2. Collegare lo storico al database
 1. Assicurarsi che i dati vengono caricati in modo corretto
3. Permettere la visualizzazione dei dettagli completi di una emergenza
 1. Implementare una finestra dedicata per mostrare le informazioni dettagliate e complete
4. Testare le funzionalità dello storico
 1. Verificare che i dati vengano caricati in modo corretto
 2. Testare la visualizzazione su diversi dispositivi

4. Filtraggio e ricerca emergenze

Importanza	3 - Importante
Story points	S

Descrizione:

Come utente, voglio poter cercare le emergenze tramite parole chiave, filtrarle in base alla categoria o allo stato attuale, e ordinarle in ordine alfabetico o cronologico (sia ascendente che discendente) in modo da trovare rapidamente le informazioni più rilevanti

Criteri di accettazione:

- L'utente può cercare un'emergenza utilizzando la barra di ricerca
- L'utente può filtrare le emergenze per:
 - Categoria
 - Stato (in corso, terminata)
- L'utente può ordinare i risultati secondo diversi criteri
 - Ordine cronologico (dal più recente al meno recente e viceversa)
 - Ordine alfabetico (A-Z, Z-A)
- Il sistema aggiorna i risultati in tempo reale in base ai criteri selezionati

Task:

1. Creare la barra di ricerca nella UI
 1. Implementare un campo di ricerca testuale
2. Implementare i filtri per categoria e stato
 1. Aggiungere un menù a checkbox per selezionare le categorie
 2. Aggiungere un menù a checkbox per selezionare lo stato
3. Implementare l'ordinamento delle emergenze
 1. Creare un menù per scegliere l'ordinamento
4. Collegare la ricerca e i filtri al backend
5. Mostrare i risultati aggiornati
 1. Visualizzare gli aggiornamenti in tempo reale senza ricaricare la pagina
 2. Assicurarsi che l'utente possa combinare più filtri
6. Testare la funzionalità di ricerca e filtraggio
 1. Testare la ricerca testuale con diverse parole chiave
 2. Verificare che i filtri per categoria e stato funzionino correttamente
 3. Controllare che l'ordinamento venga applicato correttamente in tutti i casi

5. Dashboard con emergenze in corso

Importanza	1 - Essenziale
Story points	L

Descrizione:

Come utente voglio visualizzare l'elenco delle emergenze in corso nella pagina principale, affiancato alla mappa con i rispettivi segnaposto delle emergenze, in modo da restare aggiornato sulle emergenze attuali.

Criteri di accettazione:

- Le emergenze in corso vengono visualizzate in modo ordinato in un menù laterale dedicato
- Ogni emergenza è contenuta a sua volta in un riquadro che contiene:
 - Il titolo dell'emergenza
 - La categoria dell'emergenza
 - La data di pubblicazione
 - Un pulsante che reindirizza alla pagina dei dettagli per quella segnalazione
- I dati della dashboard si aggiornano in tempo reale

Task:

1. Progettare l'interfaccia della dashboard
 1. Creare una struttura a due colonne: una contiene la lista delle emergenze in corso, l'altra la mappa con i relativi segnaposto
2. Implementare il recupero delle emergenze in corso
 1. Creare un endpoint API per ottenere solo le emergenze in corso dal database
 2. Assicurarsi che i dati siano aggiornati in tempo reale
3. Implementare l'espansione dei dettagli
 1. Aggiungere un pulsante accanto a ogni emergenza per visualizzare i dettagli aggiuntivi
 2. Garantire una transizione fluida
4. Gestire l'interazione tra lista e mappa
 1. Assicurarsi che l'interazione avvenga in modo corretto e preciso

6. Visualizzazione mappa

Importanza	3 - Importante
Story points	S

Descrizione:

Come utente voglio poter visualizzare su una mappa interattiva le emergenze in corso con la relativa posizione, in modo da conoscere le aree locali coinvolte

Criteri di accettazione:

- Il sistema deve fornire una mappa aggiornata del comune di Trento, e disporre ogni segnalazione sulla mappa con segnaposti dedicati
- Le segnalazioni non più considerate "in corso" non devono apparire sulla mappa
- La mappa deve supportare funzionalità di zoom e panoramica per consentire agli utenti di esplorare diverse aree della città
- L'utente può visualizzare i dettagli principali di un'emergenza cliccando sul corrispettivo segnaposto

Task:

1. Integrazione API per la mappa
2. Collegare la mappa al database delle segnalazioni
 1. Filtrare solo le emergenze "in corso"
3. Aggiungere interazioni per gli utenti
 1. Permettere agli utenti di cliccare su un'icona per vedere i dettagli di una segnalazione
4. Aggiornamento in tempo reale
 1. Aggiornare la mappa senza bisogno di ricaricare la pagina
 2. Testare il tempo di risposta e la sincronizzazione tra frontend e backend
5. Test su diversi dispositivi: verificare che la mappa e i segnaposti siano facilmente utilizzabili su desktop e dispositivi mobili, inclusi controlli di zoom e panoramica

7. Comunicazioni operatore

Importanza	1 - Essenziale
Story points	M

Descrizione:

Come operatore comunale, voglio poter comunicare velocemente emergenze ai cittadini, così da garantire un'informazione tempestiva e chiara

Criteri di accettazione:

- L'operatore comunale può inserire una nuova comunicazione compilando tutti i campi obbligatori richiesti:
 - Inserire un titolo e la categoria dell'emergenza
 - Fornire la posizione dell'evento
 - Fornire una descrizione dell'emergenza (opzionale, ma raccomandata)
- Il sistema deve permettere all'operatore di modificare o eliminare una comunicazione (anche già pubblicata) in caso di errore commesso in fase di pubblicazione o anche nel caso in cui l'emergenza dovesse subire delle modifiche
- La comunicazione pubblicata deve essere inserita direttamente nel database delle emergenze
- L'operatore comunale, una volta terminata un'emergenza, deve poter eliminare la comunicazione dalla vista principale

Task:

1. Creare un'interfaccia dedicata per gli operatori

1. Implementare una dashboard con opzioni per creare, modificare o eliminare una segnalazione
2. Gestire la pubblicazione delle segnalazioni
 1. Progettare la dashboard per l'inserimento di tutti i campi obbligatori
 2. Implementare le API per la creazione e gestione delle segnalazioni
3. Gestire i casi in cui non vengono inseriti tutti i campi obbligatori
 1. Mostrare un messaggio d'errore appropriato per i campi mancanti
4. Gestire i casi in cui i campi inseriti non siano conformi a quelli richiesti
 1. Mostrare un messaggio d'errore dedicato segnalando i campi errati
5. Testare il sistema di comunicazione
 1. Verificare la corretta pubblicazione e visualizzazione delle segnalazioni

8. Invio segnalazioni utenti

Importanza	2 - Necessario
Story points	M

Descrizione:

Come utente, voglio poter inviare segnalazione di emergenze o situazioni a rischio, in modo da contribuire alle segnalazioni inviate alla piattaforma e far sì che le autorità possano intervenire tempestivamente

Criteri di accettazione:

- L'utente può accedere a un modulo per inviare una segnalazione
- Il modulo include campi obbligatori come descrizione e posizione
- Se i dati sono incompleti, viene mostrato un messaggio di errore dedicato
- Dopo l'invio, l'utente riceve una conferma visiva che la segnalazione è stata registrata
- L'operatore riceve la segnalazione in tempo reale

Task:

1. Creare l'interfaccia del modulo di segnalazione
 1. Progettare un modulo intuitivo con i campi richiesti
2. Implementare la validazione dei dati in ingresso
 1. Assicurarsi che tutti i campi obbligatori richiesti siano compilati prima dell'invio
3. Collegare il modulo al backend
 1. Creare un endpoint API per ricevere e salvare le segnalazioni del database
 2. Associare le segnalazioni all'utente che l'ha inviata
4. Conferma dopo l'invio della segnalazione
 1. Mostrare un messaggio visivo di conferma dell'invio

5. Inviare le segnalazioni agli operatori in tempo reale
 1. Assicurarsi che le segnalazioni vengano salvate correttamente e visualizzate nel backend
6. Testare il processo di invio segnalazione
 1. Verificare che il modulo funzioni correttamente
 2. Simulare segnalazioni incomplete o errate per verificare la gestione degli errori

9. Gestione segnalazioni per l'operatore

Importanza	3 - Importante
Story points	M

Descrizione:

Come operatore, voglio poter visualizzare le segnalazioni inviate dagli utenti e cambiarne lo stato, così da poter approvare o rifiutare le segnalazioni e mantenere il sistema aggiornato

Criteri di accettazione:

- L'operatore può accedere a una lista di segnalazioni ricevute
- Ogni segnalazione mostra i dettagli inseriti dall'utente in fase di compilazione
- L'operatore può modificare lo stato della segnalazione da "in attesa" a "approvata" o "rifiutata"

Task:

1. Creare l'interfaccia per la gestione delle segnalazioni
 1. Progettare una sezione nella dashboard dell'operatore per l'elenco delle segnalazioni ricevute
 2. Mostrare le segnalazione con i dettagli e un pulsante per modificarne lo stato
2. Implementare il cambio di stato delle segnalazioni
 1. Aggiungere opzioni per modificare lo stato delle segnalazioni
 2. Salvare le modifiche nel database
3. Testare la gestione delle segnalazioni
 1. Verificare che l'operatore possa visualizzare tutte le segnalazioni
 2. Testare il cambio di stato e il corretto aggiornamento nel database

10. Gestione segnalazioni per l'utente

Importanza	3 - Importante
Story points	M

Descrizione:

Come utente, voglio poter vedere tutte le segnalazioni che ho inviato e il loro stato attuale, così da poter monitorare l'andamento delle mie segnalazioni e sapere se sono state approvate o rifiutate

Criteri di accettazione:

- L'utente può accedere a una sezione del proprio profilo dedicata alle segnalazioni inviate
- Ogni segnalazione mostra i relativi dettagli e stato attuale
- L'utente visualizza le modifiche dello stato apportate dall'operatore in tempo reale

Task:

1. Creare la sezione "Le mie segnalazioni" nel profilo utente
 1. Aggiungere un'area dedicata nella pagina del profilo
 2. Mostrare un elenco con tutte le segnalazioni inviate dall'utente
2. Recuperare le segnalazioni dal database
 1. Implementare una chiamata alle API del backend per ottenere le segnalazioni dell'utente autenticato
 2. Mostrare i dettagli per le varie segnalazioni
3. Aggiornare dinamicamente lo stato delle segnalazioni
 1. Assicurarsi che il cambiamento di stato venga visualizzato in tempo reale
4. Testare la funzionalità
 1. Verificare che l'utente possa visualizzare tutte le segnalazioni inviate
 2. Testare la corretta visualizzazione dello stato attuale

11. Gestione profilo personale

Importanza	2 - Necessario
Story points	L

Descrizione:

Come utente, voglio poter accedere alla mia pagina profilo per visualizzare la mia email, visualizzare le segnalazioni che ho inviato, modificare la password e effettuare il log out, così da avere il controllo del mio account

Criteri di accettazione:

- L'utente può accedere alla propria pagina profilo
- La pagina mostra email, pulsante per cambiare password e per effettuare il logout
- Dopo il cambio password viene mostrato un messaggio di conferma e l'utente deve riaccedere con la nuova password
- Il pulsante di logout permette all'utente di uscire dalla piattaforma e reindirizza alla pagina principale con la dashboard

Task:

1. Creare la pagina di profilo utente
 1. Implementare un'interfaccia accessibile dal menu utente
 2. Mostrare le informazioni in modo chiaro
2. Mostrare l'indirizzo email dell'utente
 1. Recuperare e visualizzare l'email associata all'account
3. Implementare la funzionalità di cambio password
 1. Creare un form per inserire la vecchia password e una nuova password
 2. Implementare la logica di backend per aggiornare la password nel database
 3. Richiedere all'utente di effettuare nuovamente il login dopo la modifica
4. Aggiungere il pulsante di logout
 1. Implementare la logica per terminare la sessione dell'utente
 2. Reindirizzare l'utente alla pagina di login dopo il logout
5. Testare la gestione del profilo
 1. Verificare che le informazioni siano caricate correttamente
 2. Testare la modifica della password con scenari validi e non validi
 3. Assicurarsi che il logout funzioni correttamente e l'utente venga reindirizzato alla pagina di login

12. Gestione utenti per l'operatore

Importanza	4 - Utile
Story points	M

Descrizione:

Come operatore, voglio poter visualizzare una lista di tutti gli utenti registrati e avere la possibilità di eliminare gli account degli utenti, così da poter gestire in modo efficace la piattaforma e intervenire in caso di necessità

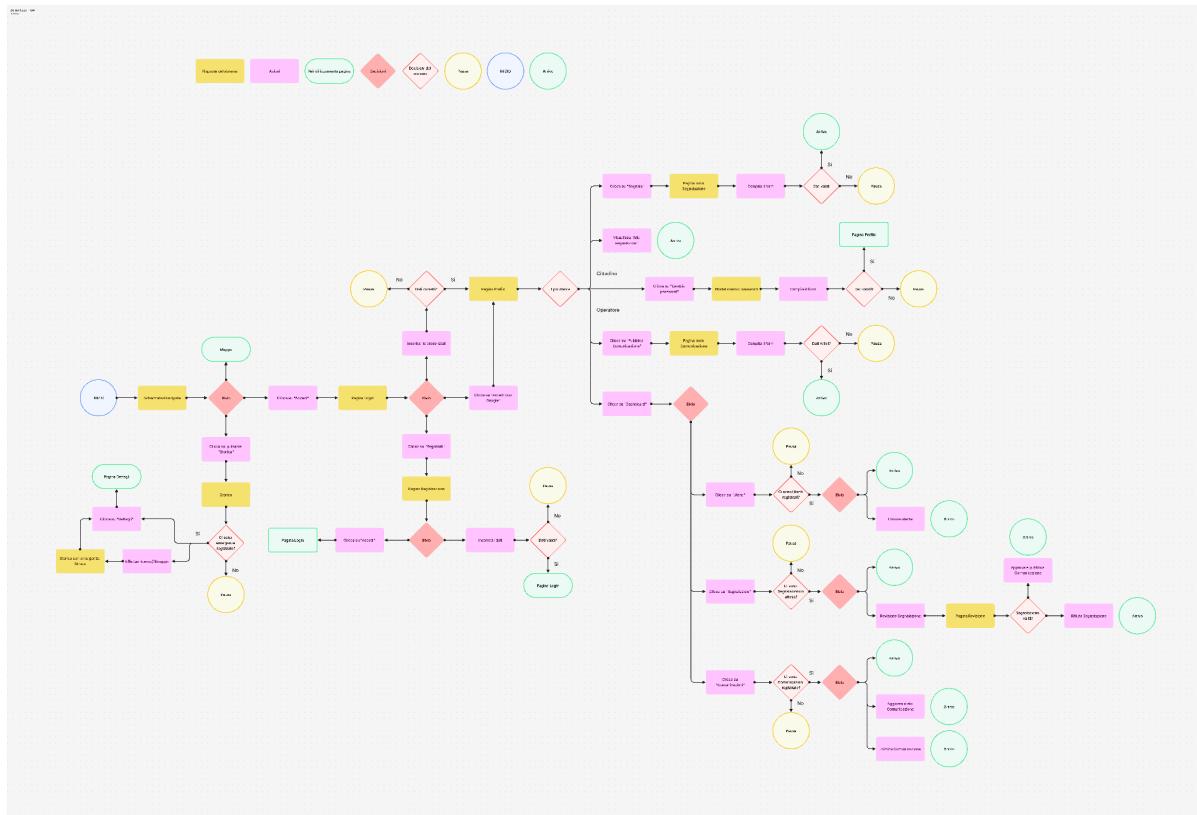
Criteri di accettazione:

- L'operatore ha accesso a una pagina di gestione utenti
- La pagina mostra un elenco di tutti gli utenti registrati contenente le email associate
- L'operatore può eliminare l'account di un utente, impedendogli di accedere alla piattaforma
- Quando un account viene eliminato, non può più essere utilizzato per effettuare il login

Task:

1. Creare l'interfaccia di gestione utenti per gli operatori
 1. Progettare e implementare una pagina accessibili solo agli operatori
 2. Mostrare una tabella con l'elenco di tutti gli utenti registrati
2. Implementare la logica di recupero degli utenti dal database
 1. Creare un endpoint backend per ottenere la lista di utenti
 2. Mostrare i dettagli degli utenti nell'interfaccia operatore
3. Implementare la funzione di eliminazione degli account
 1. Aggiungere un pulsante per disattivare l'account di un utente
 2. Mostrare un messaggio di conferma prima dell'eliminazione
 3. Assicurarsi che l'utente disattivato non possa più accedere alla piattaforma
4. Testare la gestione utenti
 1. Verificare che l'elenco degli utenti sia caricato correttamente
 2. Assicurarsi che l'eliminazione funzioni come previsto
 3. Testare il comportamento dell'account eliminato per garantire che non possa accedere

2. User Flow



A causa dell'elevata risoluzione dell'immagine, abbiamo incluso un link a *postimg* (host di immagini gratuito) per scaricare il file in alta risoluzione. Aperto il link in alto a sinistra c'è il tasto "Scarica l'immagine originale" per la qualità maggiore: <https://postimg.cc/tZyVBfvb>

3. Web APIs

Le API seguono un'architettura REST e sono state descritte secondo le specifiche OpenAPI3, la documentazione è accessibile pubblicamente attraverso il seguente link
<https://app.swaggerhub.com/apis/MIRCOSTELZER/Ocio/1.2.1>.

Sono stati individuati tre endpoint principali in questa demo, alla quale sono state aggiunte le opportune sotto-risorse e le relative operazioni CRUD utilizzate. In aggiunta sono stati specificati dei Security Schemes che fanno riferimento alla protezione delle rotte riservate.

La specifica è visualizzabile anche dal repository di Github al link https://github.com/mircostelzer/IS-Project/blob/main/swagger_code/oas3.yaml. Il contenuto del file è riportato qui per esteso:

```
openapi: 3.0.0
info:
  title: Ocio API
  description: API for emergencies communication and users management.
  version: 1.2.1
servers:
  - url: https://localhost:5000/api
    description: Localhost
  - url: https://ocio-backend.onrender.com/api
    description: Main backend deployment
  - url: https://ocio-frontend.onrender.com
    description: Main frontend deployment

paths:
  /login:
    post:
      summary: Sends a request for the JWT
      requestBody:
        description: User's credentials
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                email:
                  type: string
                password:
                  type: string
      responses:
        '201':
          description: JWT generated successfully
          content:
            application/json:
              schema:
                type: object
                properties:
                  token:
                    type: string
```

```
        description: JWT for authentication
'404':
    description: User not found
'401':
    description: Wrong credentials
'500':
    description: Error in token creation
tags:
- Login
/emergencies:
get:
    summary: Gets the list of the emergencies (in progress and ended), with the
possibility to filter them
    parameters:
- name: state
    in: query
    description: Filter for the state of the emergencies
    required: false
    schema:
        type: string
        enum: [In corso, Terminato]
responses:
'200':
    description: (Filtered) list of the emergencies
    content:
        application/json:
            schema:
                type: array
                items:
                    $ref: '#/components/schemas/Emergency'
'500':
    description: Error in emergencies recovery
tags:
- Emergencies

post:
    summary: Creates a new emergency
    security:
- bearerAuth: []
    requestBody:
        required: true
        content:
            application/json:
```

```
schema:
  $ref: '#/components/schemas/Emergency'
responses:
  '201':
    description: Emergency created. Link in the Location header
    headers:
      Location:
        description: Link to the new emergency
        style: simple
        explode: false
        schema:
          type: string
  '400':
    description: Error in emergency creation
tags:
  - Emergencies

/emergencies/{id}:
get:
  summary: Gets a single emergency
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: string
  responses:
    '200':
      description: Details of the emergency
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Emergency'
    '404':
      description: Emergency not found
    '500':
      description: Error in emergency recovery
tags:
  - Emergencies

put:
  summary: Updates the details of an emergency
  security:
```

```
- bearerAuth: []

parameters:
- name: id
  in: path
  required: true
  schema:
    type: string

requestBody:
  required: true
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/Emergency'

responses:
  '200':
    description: Emergency updated successfully
  '404':
    description: Emergency not found
  '400':
    description: Error in emergency update

tags:
- Emergencies

delete:
  summary: Deletes an emergency
  security:
  - bearerAuth: []

parameters:
- name: id
  in: path
  required: true
  schema:
    type: string

responses:
  '204':
    description: Emergency deleted successfully
  '404':
    description: Emergency not found
  '500':
    description: Error in emergency deletion

tags:
- Emergencies
```

```
/users:  
  get:  
    summary: Get a list of all the users  
    security:  
      - bearerAuth: []  
    responses:  
      '200':  
        description: List of the users  
        content:  
          application/json:  
            schema:  
              type: array  
              items:  
                $ref: '#/components/schemas/User'  
      '500':  
        description: Error in users recovery  
    tags:  
      - Users  
  
  post:  
    summary: Creates a new user  
    requestBody:  
      required: true  
      content:  
        application/json:  
          schema:  
            $ref: '#/components/schemas/User'  
    responses:  
      '201':  
        description: User created. Link in the Location header  
        headers:  
          Location:  
            description: Link to the new user  
            style: simple  
            explode: false  
            schema:  
              type: string  
      '400':  
        description: Error in user creation  
    tags:  
      - Users  
  
/users/{id}:
```

```
get:
  summary: Gets a single user
  security:
    - bearerAuth: []
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: string
  responses:
    '200':
      description: Details of the user
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/User'
    '404':
      description: User not found
    '400':
      description: Error in user recovery
  tags:
    - Users
delete:
  summary: Deletes a user
  security:
    - bearerAuth: []
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: string
  responses:
    '204':
      description: User deleted successfully
    '404':
      description: User not found
    '500':
      description: Error in user deletion
  tags:
    - Users
```

```
/users/{id}/password:  
put:  
  summary: Updates the password of a user  
  security:  
    - bearerAuth: []  
  parameters:  
    - name: id  
      in: path  
      required: true  
      schema:  
        type: string  
  requestBody:  
    required: true  
    content:  
      application/json:  
        schema:  
          type: object  
          properties:  
            oldPassword:  
              type: string  
            newPassword:  
              type: string  
  responses:  
    '200':  
      description: Password updated successfully  
    '404':  
      description: User not found  
    '400':  
      description: Incorrect old password  
    '403':  
      description: You cannot change another user's password  
    '500':  
      description: Server error  
  tags:  
    - Users  
  
/reports:  
get:  
  summary: Retrieve all reports  
  security:  
    - bearerAuth: []  
  responses:  
    '200':  
      description: List of the reports
```

```
content:
  application/json:
    schema:
      type: array
      items:
        $ref: '#/components/schemas/Report'
'500':
  description: Error in reports recovery
tags:
- Reports
post:
  summary: Creates a new report
  security:
  - bearerAuth: []
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Report'
responses:
'201':
  description: Report created. Link in the Location header
  headers:
    Location:
      description: Link to the new report
      style: simple
      explode: false
      schema:
        type: string
'400':
  description: Error in report creation
tags:
- Reports

/reports/{id}:
get:
  summary: Gets a single report
  security:
  - bearerAuth: []
parameters:
- name: id
  in: path
```

```
    required: true
    schema:
      type: string
  responses:
    '200':
      description: Details of the report
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Report'
    '404':
      description: Report not found
    '500':
      description: Error in report recovery
  tags:
    - Reports
put:
  summary: Updates a report
  security:
    - bearerAuth: []
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: string
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/Report'
  responses:
    '200':
      description: Report updated successfully
    '404':
      description: Report not found
    '400':
      description: Error in report update
  tags:
    - Reports
delete:
  summary: Deletes a report
```

```
  security:
    - bearerAuth: []
  parameters:
    - name: id
      in: path
      required: true
      schema:
        type: string
  responses:
    '204':
      description: Report deleted successfully
    '404':
      description: Report not found
    '500':
      description: Error in report deletion
  tags:
    - Reports

  /reports/myReports:
    get:
      summary: Retrieve all reports created by the user
      security:
        - bearerAuth: []
      responses:
        '200':
          description: List of the reports
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/Report'
        '500':
          description: Error in reports recovery
      tags:
        - Reports

  components:
    schemas:
      User:
        type: object
        required:
          - id
```

```
- email
- password

properties:
  id:
    type: string
    description: ID of the user
    example: "65f4b8a1d2e3a5b6c7f8d7w1"

  email:
    type: string
    description: Email of the user
    example: "mario.rossi@gmail.com"

  password:
    type: string
    description: Password of the user
    example: "password123"

  Emergency:
    type: object
    required:
      - id
      - title
      - category
      - startDate
      - location
      - state

    properties:
      id:
        type: string
        description: ID of the emergency
        example: "65f4b8a1d2e3a5b6c7f8d3a8"

      title:
        type: string
        description: Title of the emergency
        example: "Alluvione del sottopassaggio"

      category:
        type: string
        enum: [frana, alluvione, ghiaccio, crollo_strutturale, incendio, neve,
tempesta, caduta_albero]
        description: Category of the emergency
        example: "frana"

      startDate:
        type: string
        format: date-time
```

```
        description: Timestamp indicating when the emergency started
        example: "2024-10-26T10:02:00Z"
      endDate:
        type: string
        format: date-time
        description: Timestamp indicating when the emergency finished
        example: "2024-10-26T15:30:00Z"
      location:
        type: string
        description: Location of the emergency
        example: "Via Sommarive 9"
      coordinates:
        type: object
        properties:
          lat:
            type: number
            description: Latitude of the emergency location
            example: 46.066422
          lon:
            type: number
            description: Longitude of the emergency location
            example: 11.149765
      state:
        type: string
        enum: [In corso, Terminato]
        default: "In corso"
      description:
        type: string
        description: Description of the emergency
        example: "I forti venti hanno causato la caduta di un albero sulla strada
in via Sommarive, la viabilità è stata di conseguenza sospesa temporaneamente."
    Report:
      type: object
      required:
        - id
        - startDate
        - location
        - state
        - description
      properties:
        id:
          type: string
```

```
        description: ID of the report
        example: "65f4b8a1d2e3a5b6c7f8d9e0"
      startDate:
        type: string
        format: date-time
        description: Timestamp indicating when the emergency was reported
        example: "2024-10-25T10:00:00Z"
      location:
        type: string
        description: Location of the reported emergency
        example: "Via Sommarive 9"
      coordinates:
        type: object
        properties:
          lat:
            type: number
            description: Latitude of the emergency location
            example: 46.066422
          lon:
            type: number
            description: Longitude of the emergency location
            example: 11.149765
      state:
        type: string
        enum: [pending, approved, rejected]
        default: "pending"
      description:
        type: string
        description: Description of the report
        example: "A landslide has partially blocked the road, making it dangerous
to drive."
      createdBy:
        type: string
        format: uuid
        description: "ID of the reporting user"
        example: "65b23f18d4f1c3e58a9a7c12"
    securitySchemes:
      bearerAuth:
        type: http
        scheme: bearer
        bearerFormat: JWT
```

4. Implementazione

L'applicazione è stata sviluppata utilizzando Node.js per il backend e Vue per il frontend. La scelta di queste tecnologie si basa sostanzialmente sul materiale fornito durante il corso, non avendo esperienza uniforme e approfondita all'interno del gruppo per quanto riguarda il web development abbiamo deciso di attenerci a quanto consigliato dai docenti.

4.1 Organizzazione del codice

Il codice dell'intero progetto, comprensivo di backend e frontend, è accessibile su Github (<https://github.com/mircostelzer/IS-Project>) ed è organizzato nel seguente modo:

- ./github/workflows	configurazione CI/CD con Github Actions
- /backend	cartella principale per il backend
- ./jest	configurazione variabili per il testing
- /authentication	middleware per autenticazione
- /config	configurazione del server
- /controllers	controller per le API
- /models	modelli mongoose dei dati
- /routes	gestione delle rotte per le API
- /testing	test-suites
- app.js	applicazione Express.js
- index.js	avvio del server
- jest.config.js	configurazione dell'ambiente di testing
- package.json	file di configurazione npm
- /frontend	cartella principale per il frontend
- /public	immagini e icone
- /src	cartella file sorgenti frontend
- /assets	configurazione principale CSS
- /components	componenti Vue
- /data	dati utilizzate dalla WebApp
- /router	gestione indirizzamento pagine
- /states	gestione dati utente
- /views	viste della WebApp
- App.vue	applicazione principale Vue
- main.js	file per avvio applicazione Vue
- index.html	file html principale
- jsconfig.json	configurazione per la compilazione
- package.json	file di configurazione npm
- postcss.config.js	configurazione PostCSS
- tailwind.config.js	configurazione Tailwind
- vite.config.js	configurazione Vite
- /swagger_code	documentazione API
- .gitignore	configurazione repository git

4.2 Branching strategy e organizzazione

A fronte dei diversi livelli di competenza nell'ambito dello sviluppo di applicazioni per il Web all'interno del gruppo, abbiamo deciso di dividere il carico di lavoro in questo modo: due membri hanno lavorato principalmente al backend, definendo le API e creando il setup per il server; il terzo membro del gruppo si è occupato principalmente del frontend, data la sua esperienza pregressa. Il lavoro veniva in ogni caso revisionato da tutti i membri del gruppo, dopodichè durante i meeting settimanali venivano approvate le modifiche e venivano effettuati i commit. Anche per questo motivo in alcuni casi si possono notare degli sbilanciamenti con il numero di commit, poiché spesso alcune modifiche venivano apportate in presenza con tutti i membri e il commit veniva effettuato da chi in quel momento stava scrivendo. In totale sono stati eseguiti più di 150 commit suddivisi equamente tra i membri del gruppo.

Per quanto riguarda l'organizzazione del repository remoto del progetto, abbiamo deciso di adottare una strategia di tipo Feature Branch Workflow, così da mantenere il branch principale sempre funzionante e al contempo poter lavorare indipendentemente su feature diverse. Sia backend che frontend si trovano nella stessa repository. I principali branch su cui sono state sviluppate le varie funzionalità del progetto, sempre derivati da quello principale (main), sono i seguenti:

- swagger: branch per la documentazione delle API su file .yaml
- routes: principale sviluppo delle API e gestione delle rotte nel backend
- backend: sviluppo delle componenti principali del backend
- schemas: definizione degli schemi di mongoose
- authentication: implementazione dei servizi di autenticazione
- frontend: sviluppo delle componenti principali del frontend
- testing: sviluppo dei casi di test per l'applicazione

4.3 Dependencies

Il progetto npm si basa sui seguenti moduli esterni:

- **Backend**
 - **Express**: framework Web per il backend
 - **Cors**: supporto alle chiamate CORS
 - **Google-auth-library**: supporto al login con Google
 - **Jsonwebtoken**: autenticazione tramite JWT
 - **Mongoose**: supporto per la gestione dei dati e le interazioni con MongoDB
- **Frontend**

- **Vue**: framework per il frontend
- **Vue-router**: supporto al routing delle pagine
- **Leaflet**: libreria per mappe Javascript interattive
- **Algolia Search**: ricerca e filtraggio real-time
- **JS-cookies**: gestione di salvataggio e rimozione cookie

E le seguenti dipendenze di sviluppo:

- **Backend**
 - **Dotenv**: gestione delle variabili d'ambiente tramite file .env
 - **Jest**: framework per il testing
 - **Mongodb-server-memory**: supporto per testing tramite database in memoria
 - **Supertest**: supporto al testing degli endpoint express
- **Frontend**
 - **Vite**: tool di supporto frontend
 - **Tailwindcss**: libreria di classi CSS prefabbricate
 - **Heroicons**: icone SVG apposite per Tailwind
 - **DaisyUI**: libreria di componenti per la GUI
 - **PostCSS**: libreria per scrivere CSS dinamico
 - **Autoprefixer**: plugin PostCSS

4.4 Database

Abbiamo individuato due strutture principali per i dati della nostra applicazione, definite tramite mongoose:

- **User:**

```
•
•
{
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  role: {
    type: String,
    enum: ["citizen", "operator"],
    default: "citizen",
  },
}
•
```

- **Emergency:**

```
•
{
  title: { type: String, required: true },
  category: { type: String, required: true },
  startDate: { type: Date, default: Date.now, required: true },
  endDate: { type: Date },
  location: { type: String, required: true },
  coordinates: {
    type: {
      lat: { type: Number },
      lon: { type: Number }
    },
    _id: false
  },
  state: {
    type: String,
    enum: ["In corso", "Terminato"],
    default: "In corso",
  },
  description: { type: String }
}
```

- **Report:**

```
•
•
• {
    startDate: { type: Date, default: Date.now, required: true },
    location: { type: String, required: true },
    coordinates: {
        type: {
            lat: { type: Number },
            lon: { type: Number }
        },
        _id: false
    },
    state: {
        type: String,
        enum: ["pending", "approved", "rejected"],
        default: "pending",
        required: true
    },
    description: { type: String, required: true },
    createdBy: {
        type: Schema.Types.ObjectId,
        ref: "User",
        required: true
    }
}
```

4.5 Testing

Abbiamo sviluppato 5 test-suites per le nostre API che permettono di verificare il corretto funzionamento delle funzionalità implementate. I test sono utilizzati all'interno della configurazione CI/CD. I vari casi di test si basano sui requisiti funzionali individuati nei documenti precedenti che sono stati implementati nella demo finale, i numeri fanno infatti riferimento a questi ultimi.

L'implementazione delle test-suites è stata organizzata in una cartella dedicata, contenente i vari casi di test divisi a seconda dell'endpoint mirato. Il testing è stato effettuato tramite le librerie di Jest e il package npm Supertest. Poiché l'applicazione è basata su ES Modules (ESM), l'integrazione con Jest ha richiesto delle configurazioni aggiuntive e alcune componenti risultano essere ancora in fase sperimentale, ma comunque funzionanti. Dei 30 casi di test descritti, 23 sono stati implementati, mentre i restanti sono facilmente verificabili dal frontend.

N. test case	Descrizione	Input Data	Precondizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
2.1	Registrazione con credenziali valide	Email e password valide	Utente non registrato	-	Viene creato l'account con successo	Viene creato l'account con successo	
2.2	Registrazione tramite email già esistente nel database	Email già utilizzata e password valida	Email già registrata nel database	Caso di test da effettuare dopo il 2.1 con la stessa mail	Il sistema mostra un messaggio di errore nella creazione dell'account	Il sistema mostra un messaggio di errore nella creazione dell'account	
2.3	Registrazione inserendo password e conferma password diverse	Email e password valide, conferma password diversa	Utente non registrato	-	Viene mostrato un messaggio d'errore nella conferma della password	Viene mostrato un messaggio d'errore nella conferma della password	Gestito lato frontend
2.4	Registrazione inserendo una mail non valida	Email con formato non valido, password valida	-	-	Il sistema mostra un messaggio di errore nell'inserimen	Il sistema mostra un messaggio di errore nell'inserime	

					to dei dati	nto dei dati	
3.1	Login con credenziali valide	Email e password valide, già registrate nel database	Utente già registrato	-	Viene effettuato l'accesso correttamente e viene fornito un token	Viene effettuato l'accesso correttamente e viene fornito un token	
3.2	Login inserendo indirizzo email non esistente	Email non registrata nel sistema	Utente non registrato	-	Il sistema mostra un messaggio di errore nella procedura di login	Il sistema mostra un messaggio di errore nella procedura di login	
3.3	Login inserendo password errata	Email registrata nel sistema, password errata	Utente già registrato	-	Il sistema mostra un messaggio di errore nella procedura di login	Il sistema mostra un messaggio di errore nella procedura di login	
4.1	Visualizzazione emergenze in corso	-	Emergenze in corso	-	Il sistema mostra correttamente le emergenze sulla mappa	Il sistema mostra correttamente le emergenze sulla mappa	
4.2	Visualizzazione dettagli emergenza	Click su una spunta presente sulla mappa	Emergenze in corso	-	Il sistema reindirizza alla pagina dei dettagli	Il sistema reindirizza alla pagina dei dettagli	Requisito gestito lato frontend
4.3	Nessuna emergenza in corso	-	Nessuna emergenza registrata o emergenze tutte terminate	-	Il sistema non mostra alcuna spunta sulla mappa	Il sistema non mostra alcuna spunta sulla mappa	
5.1	Visualizzazione di tutte le emergenze registrate	-	Emergenze registrate nel sistema	-	Il sistema mostra una lista con tutte le emergenze registrate nel sistema	Il sistema mostra una lista con tutte le emergenze registrate nel	

						sistema	
5.2	Visualizzazione di un'emergenza eliminata	ID di un'emergenza eliminata	Emergenza precedente mente eliminata	Da effettuare in seguito al caso di test 10.4, utilizzando l'emergenza eliminata	Il sistema mostra un messaggio di errore nella visualizzazione dell'emergenza	Il sistema mostra un messaggio di errore nella visualizzazione dell'emergenza	
5.3	Ricerca emergenze	Testo di ricerca	Emergenze registrate nel sistema	-	Il sistema mostra le emergenze corrispondenti alla ricerca		Gestito lato frontend tramite API Algolia
7.1	Accesso alla pagina utente correttamente	-	Effettuare l'accesso al sistema	-	Il sistema fornisce i dati personali dell'utente	Il sistema fornisce i dati personali dell'utente	Nella demo l'accesso alla pagina utente viene effettuato attraverso cookies
7.2	Accesso alla pagina utente senza aver effettuato il login	-	L'utente non deve aver effettuato l'accesso	-	Il sistema mostra un messaggio di errore nell'accedere alla pagina dell'utente	Il sistema mostra un messaggio di errore nell'accedere alla pagina dell'utente	
7.3	Modifica password corretta tramite account esistente	ID utente registrato, token valido, vecchia e nuova password	L'utente deve aver effettuato l'accesso	-	Il sistema modifica correttamente la password e mostra un messaggio di successo	Il sistema modifica correttamente la password e mostra un messaggio di successo	
8.1	Invio di una segnalazione correttamente	Tutti i dati richiesti dal form per l'invio, token valido	Effettuare l'accesso al sistema come cittadino	-	Il sistema registra correttamente la segnalazione con stato "in attesa di verifica"	Il sistema registra correttamente la segnalazione con stato "in attesa di verifica"	

8.2	Compilazione incompleta dei campi obbligatori	Descrizione della segnalazione, token valido	Effettuare l'accesso al sistema come cittadino	-	Il sistema mostra un messaggio di errore nella creazione della segnalazione	Il sistema mostra un messaggio di errore nella creazione della segnalazione	
8.3	Invio di una segnalazione come operatore	Dati richiesti dal form per l'invio, token operatore	Effettuare l'accesso al sistema come operatore	-	Il sistema mostra un messaggio di errore nell'autorizzazione	Il sistema mostra un messaggio di errore nell'autorizzazione	
10.1	Pubblicazione di una comunicazione con dati completi	Tutti i dati richiesti dal form per la pubblicazione, token valido	Login tramite un account operatore	-	Il sistema registra correttamente l'emergenza con stato "in corso" e mostra un messaggio di successo nella pubblicazione	Il sistema registra correttamente l'emergenza con stato "in corso" e mostra un messaggio di successo nella pubblicazione	
10.2	Modifica dello stato di un'emergenza	ID emergenza registrata nel sistema, token valido	Login tramite account operatore, emergenza registrata	Da effettuare in seguito al caso di test 10.1, specificando la stessa emergenza	Il sistema registra correttamente la modifica e aggiorna la data di fine dell'emergenza	Il sistema registra correttamente la modifica e aggiorna la data di fine dell'emergenza	
10.3	Pubblicazione di una comunicazione con dati obbligatori parziali	Titolo e categoria validi, token valido	Login tramite un account operatore	-	Il sistema mostra un messaggio di errore nella creazione dell'emergenza	Il sistema mostra un messaggio di errore nella creazione dell'emergenza	
10.4	Eliminazione di una comunicazione registrata	ID emergenza già registrata,	Login tramite un account operatore, emergenza	Da effettuare in seguito al caso di test	Il sistema elimina correttamente l'emergenza	Il sistema elimina correttamente	

	nel sistema	token valido	già registrata	10.2, specificando sempre la stessa emergenza	dal database e mostra un messaggio di successo	l'emergenza dal database e mostra un messaggio di successo	
11.1	Visualizzazione di tutte le segnalazioni registrate	Token valido	Segnalazioni registrate nel sistema, accesso come operatore	-	Il sistema mostra una lista con le segnalazioni registrate nel sistema	Il sistema mostra una lista con le segnalazioni registrate nel sistema	
11.2	Revisione di una segnalazione	ID di una segnalazione registrata nel sistema, token valido	Segnalazioni registrate nel sistema, login tramite account operatore	Da effettuare in seguito al caso di test 8.1, utilizzando la stessa segnalazione	Il sistema registra correttamente la modifica nello stato della segnalazione	Il sistema registra correttamente la modifica nello stato della segnalazione	Nella demo l'utente non verrà notificato della revisione
11.3	Revisione di una segnalazione come cittadino	ID di una segnalazione registrata nel sistema, token cittadino	Segnalazione registrata nel sistema, login tramite account cittadino	Da effettuare in seguito al caso di test precedente, utilizzando la stessa segnalazione	Il sistema mostra un messaggio di errore nell'autorizzazione	Il sistema mostra un messaggio di errore nell'autorizzazione	
11.4	Eliminazione di una segnalazione	ID di una segnalazione registrata nel sistema, token valido	Segnalazione registrata nel sistema, login tramite account operatore	Da effettuare in seguito al caso di test precedente, utilizzando la stessa segnalazione	Il sistema elimina correttamente la segnalazione dal database e mostra un messaggio di successo	Il sistema elimina correttamente la segnalazione dal database e mostra un messaggio di successo	
12.1	Visualizzazione lista degli account registrati	Token valido	Login tramite un account operatore	-	Il sistema mostra una lista degli account registrati nel sistema	Il sistema mostra una lista degli account registrati nel sistema	
12.2	Eliminazione di un account	ID di un account	Login tramite un account	Da effettuare in	Il sistema elimina	Il sistema elimina	Nella demo l'utente

	registrato	registrato nel sistema, token valido	operatore, account utente registrato nel sistema	seguito al caso di test 7.3, utilizzando gli stessi dati	correttamente l'account dal database e mostra un messaggio di successo	correttamente l'account dal database e mostra un messaggio di successo	viene effettivamente eliminato dal database anzichè essere disattivato, inoltre l'operatore non può allegare un messaggio
12.3	Visualizzazione di un account non registrato nel sistema	ID di un account non registrato nel sistema, token valido	Login tramite account operatore, account non registrato nel sistema	Da effettuare in seguito al caso di test precedente, utilizzando lo stesso ID	Il sistema mostra un messaggio di errore nella visualizzazione dell'account	Il sistema mostra un messaggio di errore nella visualizzazione dell'account	

5. Frontend

Il frontend fornisce tutte le funzionalità di visualizzazione principali, comprese la mappa con le emergenze in corso, lo storico delle comunicazioni, la pagina del proprio profilo personale e la dashboard per l'operatore comunale. Sono state inoltre implementate le funzionalità principali di inserimento, modifica ed eliminazioni dei dati dell'applicazione (comunicazioni, segnalazioni e utenti).

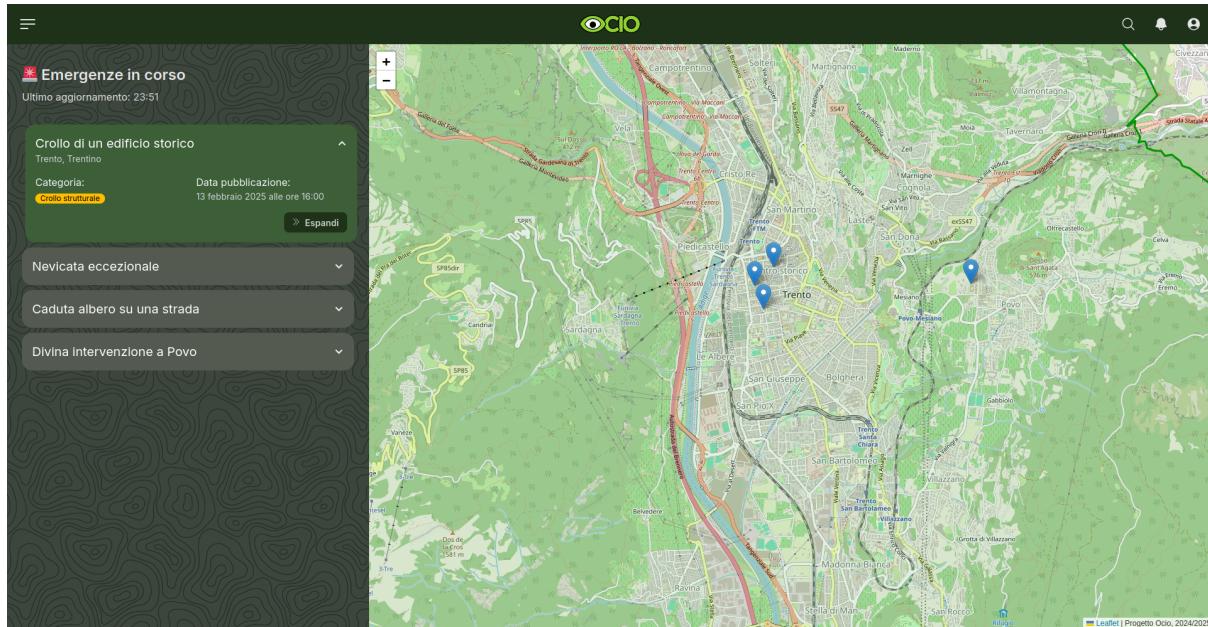


Immagine 1: Mappa con emergenze in corso

Da qui un qualsiasi tipo di utente, anche senza effettuare il login, può visualizzare la mappa con le emergenze in corso ed accedere allo storico.

Categoria:	Area interessata:	Stato:
Ghiaccio	Trento, Trentino	Terminato
Incendio	Trento, Trentino	Terminato
Crollo strutturale	Trento, Trentino	Terminato
Neve	Trento, Trentino	Terminato
Alluvione	Trento, Trentino	Terminato
Caduta albero	Trento, Trentino	Terminato
Franà	Trento, Trentino	Terminato

Immagine 2: Storico emergenze

Da qui un qualsiasi tipo di utente, anche senza effettuare il login, può visualizzare lo storico di tutte le emergenze, passate e in corso, con la possibilità di filtrarle e ordinarle secondo diversi parametri.

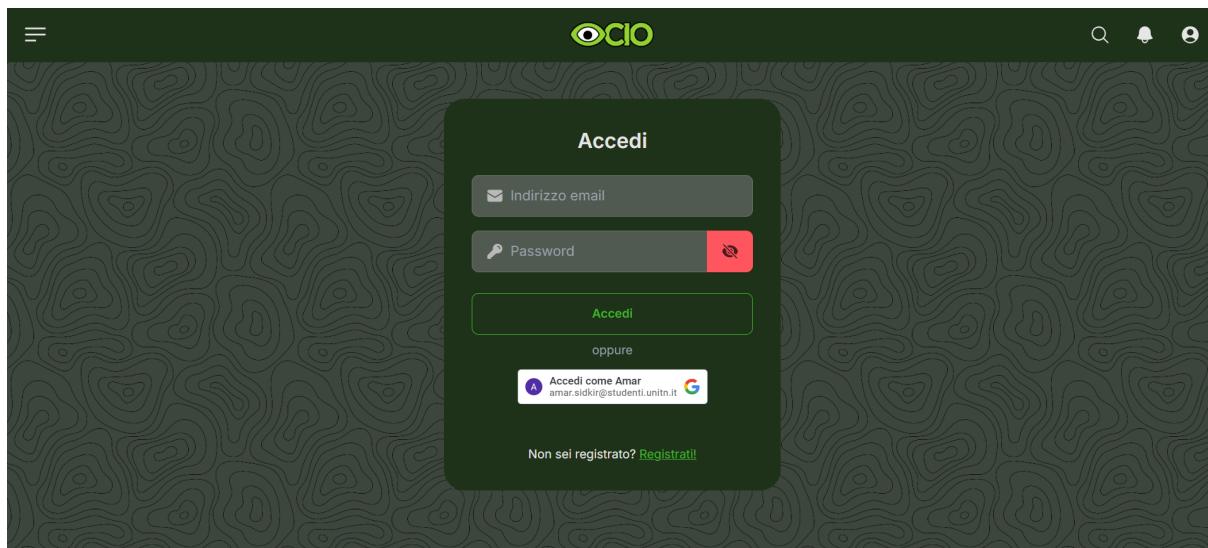


Immagine 3: form accesso

Questa pagina mostra le diverse modalità di accesso: tramite credenziali o tramite il servizio di Google Auth. L'utente ha anche la possibilità di passare alla pagina di registrazione.

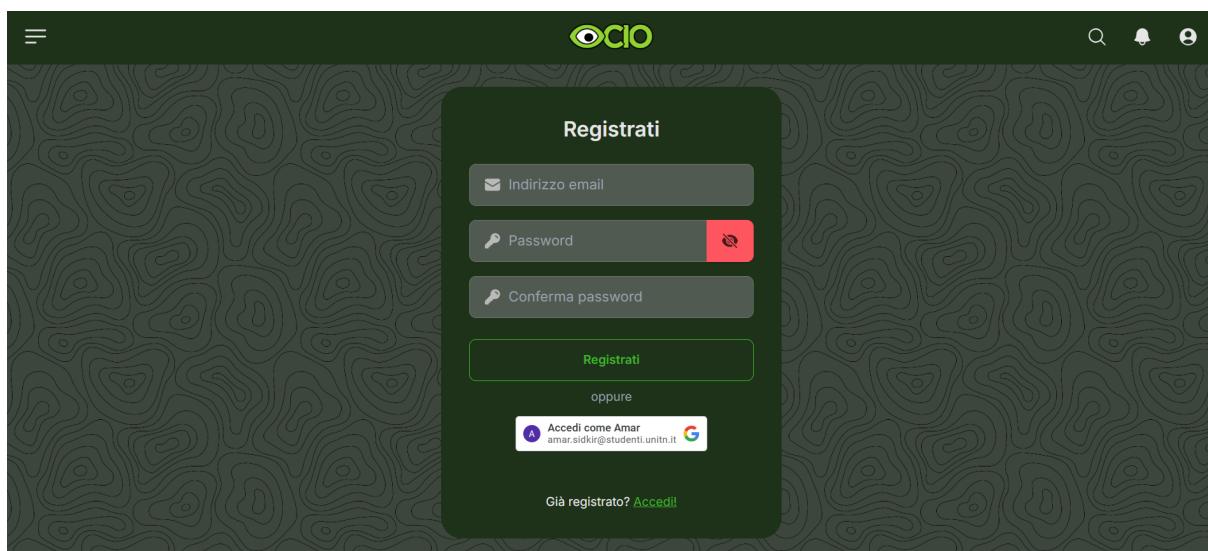
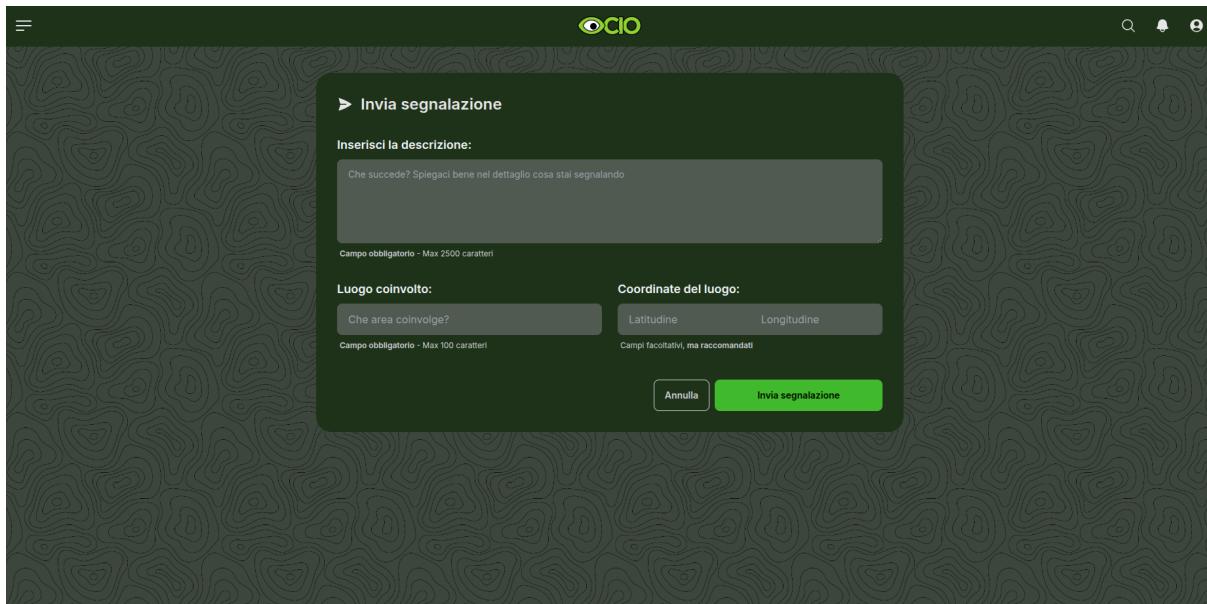


Immagine 4: form registrazione

Questa pagina mostra le diverse modalità di registrazione: tramite credenziali (email e password) o tramite il servizio di Google Auth. L'utente ha anche la possibilità di passare alla pagina di login.



► Invia segnalazione

Inserisci la descrizione:

Che succede? Spiegaci bene nel dettaglio cosa stai segnalando

Campo obbligatorio - Max 2500 caratteri

Luogo coinvolto:

Che area coinvolge?

Campo obbligatorio - Max 100 caratteri

Coordinate del luogo:

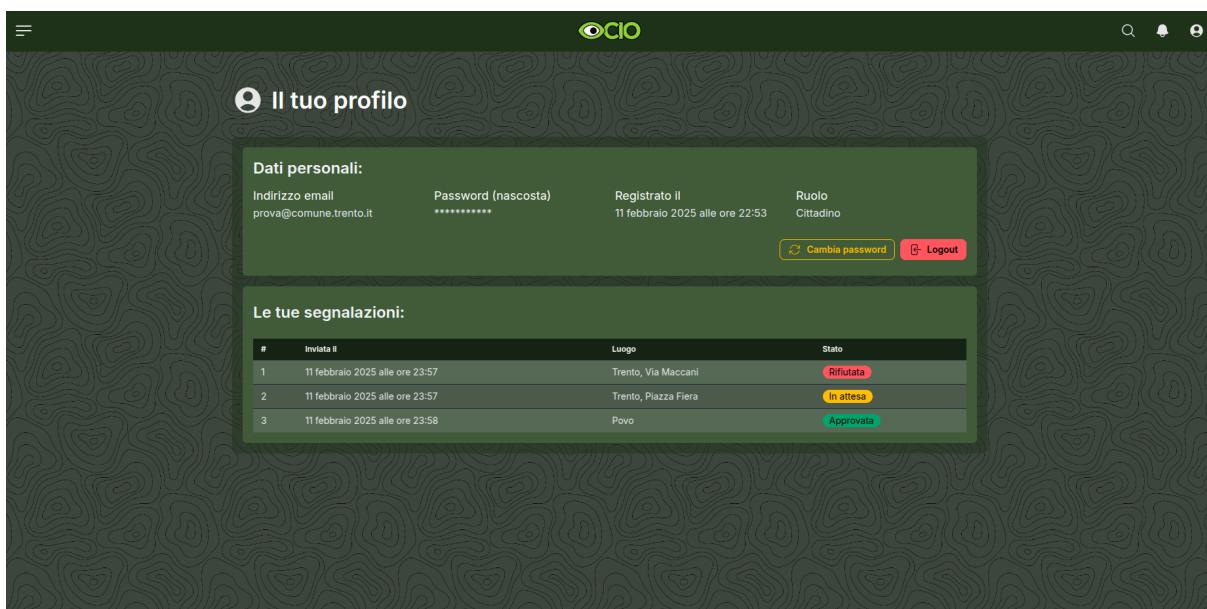
Latitudine Longitudine

Campi facoltativi, ma raccomandati

Annulla Invia segnalazione

Immagine 5: form invio segnalazione

Da qui il cittadino può inviare una segnalazione compilando i campi obbligatori del form.



Il tuo profilo

Dati personali:

Indirizzo email: prova@comune.trento.it
Password (nascosta): *****
Registrato il: 11 febbraio 2025 alle ore 22:53
Ruolo: Cittadino

Cambia password Logout

Le tue segnalazioni:

#	Inviatà il	Luogo	Stato
1	11 febbraio 2025 alle ore 23:57	Trento, Via Maccani	Rifiutata
2	11 febbraio 2025 alle ore 23:57	Trento, Piazza Fiera	In attesa
3	11 febbraio 2025 alle ore 23:58	Povo	Approvata

Immagine 6: profilo cittadino

Da qui il cittadino può visualizzare le informazioni del proprio profilo e le segnalazioni che ha inviato. Con la possibilità di cambiare password o effettuare il logout.

Pubblica comunicazione

Titolo comunicazione:
Riassumi in un titolo la segnalazione
Campo obbligatorio - Max 200 caratteri

Descrizione:
Che succede? Spiega bene nel dettaglio cosa stai segnalando
Max 2500 caratteri

Categoria:
Di che tipo è l'emergenza?
Campo obbligatorio

Data inizio:
gg / mm / aaaa

Luogo coinvolto:
Che area coinvolge?
Campo obbligatorio - Max 100 caratteri

Coordinate del luogo:
Latitudine Longitude

Bottoni: Annulla, Pubblica comunicazione

Immagine 7: form pubblica comunicazione

Da qui l'operatore può pubblicare una comunicazione compilando i campi obbligatori del form.

Dashboard operatore

Statistiche:

- Cittadini: 3 (42.86%)
- Operatori comunali: 4 (57.14%)
- Segnalazioni inviate: 10 (52.63%)
- Emergenze pubblicate: 9 (47.37%)

Operatori comunali:

#	Indirizzo mail	Ruolo	Creato il	Modificato il
1	admin@comune.trento.it	Operatore	25/11/2024 - 16:54	25/11/2024 - 16:54
2	mirco.stelzer@studenti.unin.it	Operatore	10/02/2025 - 22:06	11/02/2025 - 23:24
3	filippo.xausa@studenti.unin.it	Operatore	11/02/2025 - 15:19	11/02/2025 - 23:24
4	amar.sickir@studenti.unin.it	Operatore	11/02/2025 - 21:32	11/02/2025 - 23:21

Cittadini:

#	Indirizzo mail	Ruolo	Creato il	Modificato il	Azione
1	demo_user@gmail.com	Cittadino	11/02/2025 - 18:20	11/02/2025 - 18:20	
2	provaaa@gmail.com	Cittadino	11/02/2025 - 18:23	11/02/2025 - 18:23	
3	prova@comune.trento.it	Cittadino	11/02/2025 - 22:53	11/02/2025 - 23:15	

Immagine 8: dashboard operatore, gestione utenti

Da qui l'operatore può vedere gli utenti registrati ed eventualmente eliminarli.

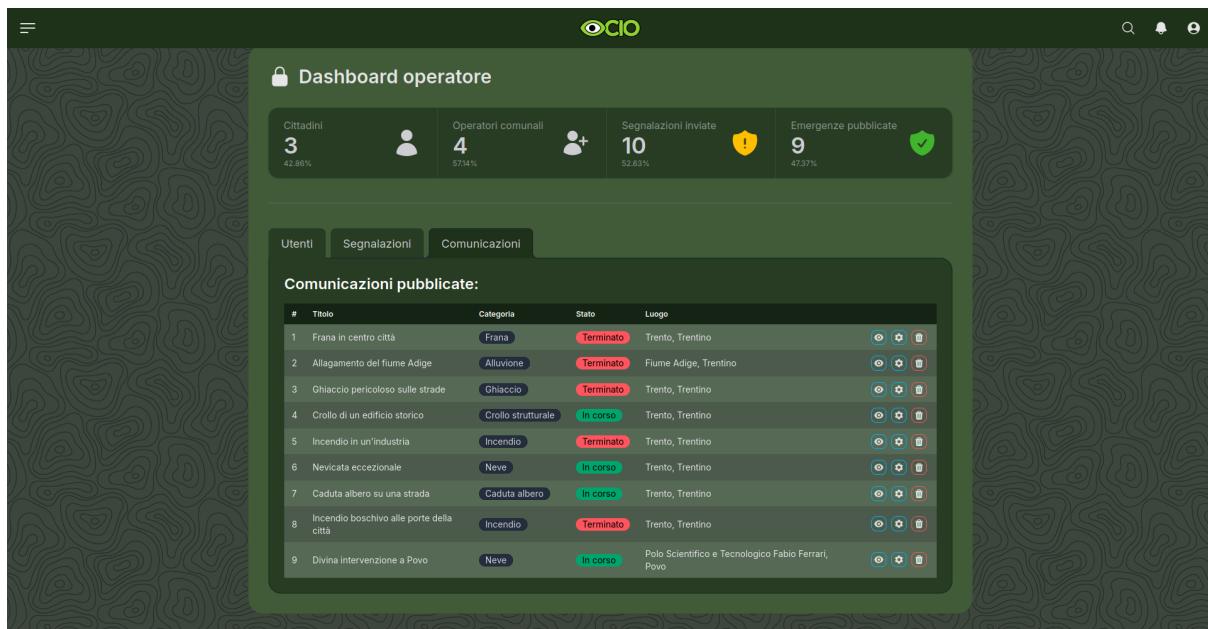


Immagine 9: dashboard operatore, gestione comunicazioni

Da qui l'operatore può vedere l'elenco delle comunicazione pubblicate, con la possibilità di modificarle o eliminarlo.

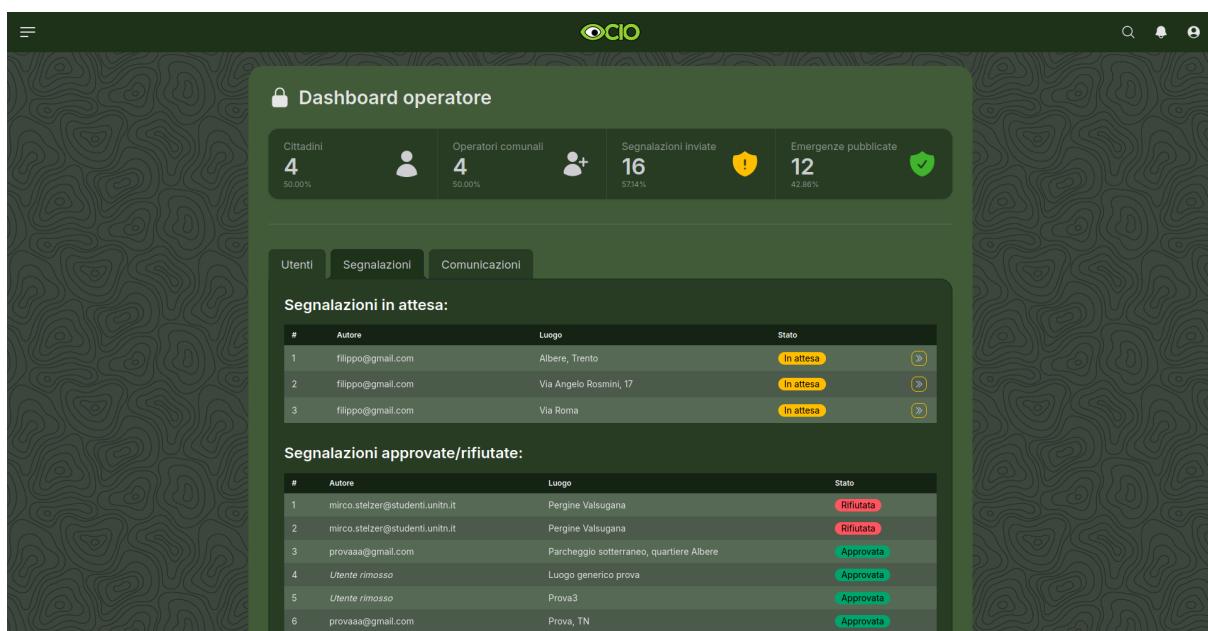


Immagine 10: dashboard operatore, gestione segnalazioni

Da qui l'operatore può vedere le segnalazioni degli utenti, con la possibilità di approvare o rifiutare quelle che sono ancora in attesa.

6. Deployment

Per quanto riguarda il deployment, è stato creato un Web Service per il backend (<https://ocio-backend.onrender.com>), sprovvisto di interfaccia per l'utente poiché direttamente collegato allo Static Site del frontend (<https://ocio-frontend.onrender.com>) tramite le variabili d'ambiente.

La configurazione CI/CD basata sulle GitHub Actions effettua tutti i casi di test definiti nel backend, dopodiché completa la build del frontend e, se tutto è andato a buon fine, procede automaticamente con il deployment.

Per usufruire delle diverse funzionalità offerte dall'applicazione, è possibile:

- navigare senza effettuare l'accesso
- effettuare l'accesso con Google
- effettuare l'accesso tramite l'account cittadino: demo_user@gmail.com, password: demopass123
- effettuare l'accesso tramite l'account operatore: admin@comune.trento.it, password: adminpass123

Per problemi con il deploy contattare:

mirco.stelzer@studenti.unitn.it

filippo.xausa@studenti.unitn.it

amar.sidkir@studenti.unitn.it

Note:

A causa dell'utilizzo di alcune API open-source come Algolia e OpenStreetMap, possono verificarsi dei ritardi di aggiornamento delle viste in caso di aggiornamento dei dati dell'applicazione.

Ad esempio, considerato il piano gratuito offerto da Algolia, l'aggiornamento della ricerca real-time recupera i dati solamente ogni 15 minuti.