

# Sviluppo

Progetto	VolontariON	
Gruppo	T01	
Documento	D3	rev: 1.2

## Indice

<b>Scopo del documento.....</b>	<b>3</b>
<b>User stories.....</b>	<b>3</b>
User Story 1: Login.....	3
User Story 2: Registrazione di un Volontario.....	4
User Story 3: Registrazione di un'associazione.....	5
User Story 4: Visualizzazione profilo associazione.....	5
User Story 5: Visualizzazione profilo Volontario.....	6
User Story 6: Modifica del Profilo.....	7
User Story 7: Modifica Password.....	8
User Story 8: Creazione Evento.....	9
User Story 9: Eliminazione Evento.....	10
User Story 10: Modifica Evento.....	11
User Story 11: Visualizzazione Evento.....	12
User Story 12: Visualizzazione lista eventi.....	13
User Story 13: Visualizzazione eventi di un'associazione.....	13
User Story 14: Iscrizione ad un Evento.....	14
User Story 15: Disiscriversi da un evento.....	15
User Story 16: Visualizzazione iscritti a un evento.....	16
User Story 17: Visualizzazione lista delle associazioni registrate.....	16
User Story 18: Seguire associazioni.....	17
User Story 19: Visualizzazione lista di eventi creati da associazioni seguite.....	17
<b>User flow.....</b>	<b>18</b>
<b>Web APIs.....</b>	<b>21</b>
<b>Implementation.....</b>	<b>21</b>
Repository organization.....	21

Organizzazione backend.....	22
Organizzazione frontend.....	22
Gestione Workflow.....	23
Dependencies backend.....	23
Dependecies frontend.....	24
Database.....	25
<b>Testing.....</b>	<b>27</b>
<b>Frontend.....</b>	<b>29</b>
Sezioni comuni.....	29
Sezioni Volontario.....	32
<b>Deployment.....</b>	<b>38</b>

# Scopo del documento

Il seguente documento riporta tutte le informazioni necessarie per descrivere lo sviluppo di una parte, quasi completa, di VolontariOn.

Nel primo capitolo vengono riportate le user stories, segue poi l'user flow. Successivamente viene riportata la struttura della repository, le dipendenze usate, i modelli realizzati, le API implementate e informazioni relative al testing. Infine è presente una sezione sul deployment e sul frontend.

## User stories

### User Story 1: Login

Come utente registrato (volontario o associazione), voglio accedere alla piattaforma inserendo le mie credenziali, in modo che possa gestire il mio profilo e interagire con le funzionalità disponibili.

#### Criteri di Accettazione:

- Il sistema deve permettere il login tramite email e password.
- Se le credenziali sono corrette, l'utente è autenticato.
- Se le credenziali sono errate, deve essere mostrato un messaggio di errore.

#### Task:

- Creare una pagina di login con i campi email e password.
- Implementare la validazione delle credenziali.
- Implementare un meccanismo di autenticazione sicuro ( token JWT).

- Implementare la gestione degli errori in caso di credenziali errate.

## User Story 2: Registrazione di un Volontario

Come utente non autenticato, voglio registrarmi come volontario sulla piattaforma, in modo che possa partecipare agli eventi e interagire con le associazioni.

### Criteri di Accettazione:

- Il modulo di registrazione deve richiedere i campi: nome, cognome, email, password, descrizione di sé, esperienze passate e competenze.
- Il numero di telefono e l'immagine del profilo devono essere opzionali.
- L'email deve essere univoca tra volontari.
- Deve essere inviata un'email di conferma per la registrazione.

### Task:

- Creare il modulo di registrazione per volontari.
- Implementare la validazione dei campi.
- Verificare l'unicità dell'email.
- Inviare un'email di conferma di avvenuta registrazione
- Implementare meccanismo di salvataggio dati nel database.

## **User Story 3: Registrazione di un'associazione**

Come utente non autenticato, voglio registrarmi come associazione sulla piattaforma, in modo che possa promuovere le mie iniziative di volontariato e gestire gli eventi.

### **Criteri di Accettazione:**

- Il modulo di registrazione deve richiedere i campi: nome dell'associazione, email, password, descrizione, obiettivi dell'associazione.
- Il numero di telefono e l'immagine del profilo/logo devono essere opzionali.
- L'email deve essere univoca tra associazioni.
- Deve essere inviata un'email di conferma per la registrazione.

### **Task:**

- Creare il modulo di registrazione per associazioni.
- Implementare la validazione dei campi.
- Verificare l'unicità dell'email.
- Inviare un'email di conferma.
- Implementare meccanismo di salvataggio dati nel database.

## **User Story 4: Visualizzazione profilo associazione**

Come utente (volontario, associazione, o utente non autenticato), voglio visualizzare il profilo completo di una specifica associazione, perché desidero conoscere meglio le informazioni sull'associazione e gli eventi che gestisce.

### **Criteri di Accettazione:**

- L'utente deve poter visualizzare la pagina profilo di un'associazione selezionata.

- La pagina del profilo dell'associazione deve contenere almeno le seguenti informazioni: nome dell'associazione, descrizione dell'associazione, immagine del profilo o logo, email, numero di telefono, obiettivi principali dell'associazione.
- L'utente deve poter visualizzare la lista degli eventi pubblicati dall'associazione.

**Task:**

- Creazione della pagina profilo dell'associazione
- Recuperare e mostrare i dati dell'associazione dal database.
- Implementare meccanismo per visualizzare la lista degli eventi pubblicati dall'associazione.

## **User Story 5: Visualizzazione profilo Volontario**

Come associazione,Voglio visualizzare il profilo completo di un volontario che si è iscritto a uno dei miei eventi,Perché desidero conoscere meglio il volontario, le sue esperienze e capire se le sue abilità sono adatte alle attività richieste.

**Criteri di Accettazione:**

- L'associazione deve poter visualizzare il profilo completo di un volontario che si è iscritto a un evento.
- La pagina del profilo del volontario deve includere almeno le seguenti informazioni: nome e cognome del volontario, immagine del profilo, email, numero di telefono, Descrizione personale, esperienze passate di volontariato, competenze specifiche del volontario.
- L'associazione deve poter visualizzare il profilo del volontario solo se il volontario si è iscritto ad almeno uno dei suoi eventi.

**Task:**

- Creazione della pagina profilo del volontario.
- Recuperare e mostrare i dati del volontario dal database.

## **User Story 6: Modifica del Profilo**

Come un utente registrato (volontario o associazione), Voglio poter modificare le informazioni del mio profilo, In modo che possa mantenere aggiornate le mie informazioni.

**Criteri di Accettazione:**

- Gli utenti devono poter modificare i propri dati.
- Le modifiche devono essere salvate correttamente nel database.

**Task:**

- Creare una pagina di modifica del profilo.
- Implementare la validazione dei campi modificabili.
- Implementare meccanismo di aggiornamento dati nel database.

## User Story 7: Modifica Password

Come utente registrato (volontario o associazione), voglio modificare la mia password, in modo che possa garantire la sicurezza del mio account.

### Criteri di Accettazione:

- L'utente deve poter accedere alla sezione "Profilo" per modificare la password.
- Deve essere richiesto l'inserimento della vecchia password per procedere.
- La nuova password deve essere inserita due volte per conferma.
- Se la vecchia password è errata, deve essere mostrato un messaggio di errore.
- Se la modifica va a buon fine, l'utente deve ricevere una notifica di conferma.

### Task:

- Creare un'interfaccia nella sezione "Profilo" per la modifica della password.
- Implementare la verifica della vecchia password.
- Implementare la conferma della nuova password.
- Aggiornare la password nel database.
- Mostrare messaggi di errore o conferma in base all'esito dell'operazione.

## User Story 8: Creazione Evento

Come un'associazione registrata, voglio creare un evento sulla piattaforma, in modo che i volontari possano iscriversi e partecipare.

### Criteri di Accettazione:

- Solo le associazioni possono creare eventi.
- L'evento deve includere titolo, luogo, data e ora di inizio e di fine , descrizione e immagine (facoltativa).
- L'evento deve essere visibile nella lista degli eventi.

### Task:

- Creare il modulo di creazione evento.
- Implementare la validazione dei campi.
- Salvare l'evento nel database.

## **User Story 9: Eliminazione Evento**

Come un'associazione registrata, voglio poter eliminare un evento che ho creato, in modo che possa rimuovere eventi non più validi o cancellati.

### **Criteri di Accettazione:**

- Solo le associazioni devono poter eliminare i propri eventi.
- Una volta eliminato, l'evento deve essere rimosso definitivamente dal sistema.
- I volontari iscritti all'evento non devono più vederlo nella loro lista di eventi iscritti.

### **Task:**

- Aggiungere un pulsante "Elimina" per gli eventi nella sezione "I miei eventi" dell'associazione.
- Creare la logica per la rimozione definitiva dell'evento dal database.
- Rimuovere l'evento dalla lista degli eventi iscritti dei volontari.
- Inviare un messaggio di notifica di avvenuta eliminazione all'associazione.

## **User Story 10: Modifica Evento**

Come un'associazione registrata, voglio poter modificare i dettagli di un evento che ho creato, in modo che possa aggiornare informazioni come data, luogo o descrizione se necessario.

### **Criteri di Accettazione:**

- Solo le associazioni devono poter modificare i propri eventi.
- Devono essere modificabili: titolo, luogo, data, descrizione e immagine.
- Le modifiche devono essere salvate correttamente.

### **Task:**

- Aggiungere un pulsante "Modifica" per gli eventi nella sezione "I miei eventi" dell'associazione.
- Implementare un modulo di modifica con i campi dell'evento.
- Salvare le modifiche nel database.

## **User Story 11: Visualizzazione Evento**

Come un utente della piattaforma (volontario, associazione o utente non autenticato), voglio poter visualizzare i dettagli di un evento, in modo che possa ottenere informazioni su di esso.

### **Criteri di Accettazione:**

- Tutti gli utenti (volontari, associazioni e non autenticati) devono poter visualizzare gli eventi.
- I dettagli visibili devono includere: titolo, data, luogo, descrizione e immagine.
- L'utente deve poter essere reindirizzato alla pagina profilo dell'associazione che ha creato l'evento tramite un bottone.
- Se l'utente è un volontario autenticato, deve poter iscriversi all'evento.

### **Task:**

- Creare una pagina di dettaglio per ogni evento.
- Recuperare e mostrare i dettagli dell'evento dal database.
- Implementare meccanismo per l'iscrizione per i volontari autenticati.
- Implementare meccanismo di reindirizzamento alla pagina profilo dell'associazione.

## **User Story 12: Visualizzazione lista eventi**

Come utente della piattaforma (volontario, associazione o utente non autenticato), Voglio poter visualizzare un elenco di tutti gli eventi sulla piattaforma, in modo che possa scoprire opportunità di volontariato.

**Criteri di Accettazione:**

- La lista degli eventi deve essere visibile a tutti gli utenti, anche non autenticati.
- Ogni evento nella lista deve mostrare titolo, data, luogo e l'associazione che organizza l'evento.
- Deve essere possibile filtrare gli eventi per data.
- Deve essere possibile cercare un evento per titolo.
- L'utente deve poter essere reindirizzato alla pagina di dettaglio dell'evento tramite un bottone.

**Task:**

- Creare una pagina con la lista degli eventi.
- Recuperare e mostrare gli eventi dal database.
- Implementare filtro per data.
- Implementare una barra di ricerca per trovare eventi specifici.
- Implementare meccanismo di reindirizzamento alla pagina dell'evento.

## User Story 13: Visualizzazione eventi di un'associazione

Come utente della piattaforma (volontario, associazione o utente non autenticato), voglio visualizzare gli eventi organizzati da una specifica associazione, perché desidero scoprire le attività di volontariato offerte da tale associazione.

**Criteri di Accettazione:**

- Dalla pagina profilo dell'associazione, l'utente può dirigersi alla pagina contenente la lista di eventi di tale associazione.
- La lista degli eventi deve essere visibile a tutti gli utenti, anche non autenticati.
- Ogni evento nella lista deve mostrare titolo, data, luogo e l'associazione che organizza l'evento.

- Deve essere possibile filtrare gli eventi per data.
- Deve essere possibile cercare un evento per titolo.
- L'utente deve poter essere reindirizzato alla pagina di dettaglio dell'evento tramite un bottone.

**Task:**

- Aggiungere bottone “Visualizza eventi” alla pagine profilo delle associazioni.
- Creazione interfaccia per la lista degli eventi.
- Recuperare e mostrare gli eventi dal database.
- Implementare filtro per data.
- Implementare una barra di ricerca per trovare eventi specifici.
- Implementare bottone di reindirizzamento alla pagina dell'evento.

## **User Story 14: Iscrizione ad un Evento**

Come volontario registrato, voglio iscrivermi a un evento, in modo che possa partecipare a tale evento.

**Criteri di Accettazione:**

- Solo i volontari possono iscriversi.
- L'evento deve essere aggiunto alla lista di eventi a cui il volontario è iscritto.

**Task:**

- Creare un pulsante di iscrizione alla pagina dell'evento.
- Implementare funzionalità di iscrizione.
- Salvare l'iscrizione nel database.
- Inviare un messaggio di conferma di avvenuta iscrizione.
- Implementare controllo se un volontario è già iscritto all'evento

## **User Story 15: Disiscriversi da un evento**

Come volontario registrato, voglio poter disiscriversi da un evento a cui mi sono precedentemente iscritto, perché potrei cambiare idea o non essere più in grado di partecipare all'evento.

### **Criteri di Accettazione:**

- Il volontario deve poter visualizzare una lista degli eventi a cui è attualmente iscritto.
- Ogni evento nella lista deve avere un'opzione per permettere al volontario di disiscriversi.
- Dopo aver cliccato su "Disiscriviti", l'evento deve essere rimosso dalla lista degli eventi a cui il volontario è iscritto. Il volontario deve ricevere un messaggio che confermi la disiscrizione avvenuta con successo.

### **Task:**

- Creazione delle pagine contenente la lista degli eventi a cui il volontario è iscritto
- Implementare funzionalità di disiscrizione.
- Salvare operazione nel database.
- Inviare un messaggio di conferma di avvenuta disiscrizione.

## **User Story 16: Visualizzazione iscritti a un evento**

Come associazione , voglio visualizzare una lista di tutti i volontari che si sono iscritti a un evento organizzato dalla mia associazione, perché desidero avere un quadro chiaro delle persone che parteciperanno.

### **Criteri di Accettazione:**

- Solo l'associazione che ha creato l'evento deve poter visualizzare la lista di tutti i volontari che si sono iscritti.

- La lista degli iscritti deve mostrare, per ogni volontario: nome, cognome, email, cellulare e un bottone per andare alla pagina profilo del volontario

**Task:**

- Creazione interfaccia per la lista degli iscritti.
- Recuperare e mostrare gli iscritti dal database.
- Implementare meccanismo per dirigersi alla pagina profilo del volontario

## **User Story 17: Visualizzazione lista delle associazioni registrate**

Come utente della piattaforma (volontario, associazione o utente non autenticato), voglio visualizzare una lista di tutte le associazioni registrate sulla piattaforma, perché desidero scoprire le organizzazioni che sono attive.

**Criteri di Accettazione:**

- L'utente deve poter visualizzare una lista completa delle associazioni registrate sulla piattaforma.
- Ogni associazione nella lista deve mostrare almeno le seguenti informazioni: foto profilo, nome.
- L'utente deve poter dirigersi alla pagina profilo delle associazioni.
- L'utente deve avere la possibilità di cercare specifiche associazioni tramite una barra di ricerca, utilizzando il nome dell'associazione.

**Task:**

- Creazione interfaccia per la lista delle associazioni.
- Recuperare e mostrare le associazioni dal database.
- Implementare meccanismo per dirigersi alla pagina profilo dell'associazione.
- Implementare una barra di ricerca per trovare associazioni per nome.

## **User Story 18: Seguire associazioni**

Come volontario, voglio poter seguire una o più associazioni, perché desidero ricevere aggiornamenti sui loro eventi e rimanere informato sulle loro attività.

### **Criteri di Accettazione:**

- Il volontario deve poter seguire una o più associazioni.
- Il volontario deve poter visualizzare una lista delle associazioni che segue, con la possibilità di navigare al profilo completo dell'associazione, smettere di seguire associazioni e cercare specifiche associazioni tramite una barra di ricerca, utilizzando il nome dell'associazione.

### **Task:**

- Implementazione della logica per permettere a un volontario di seguire un'associazione:
- Creare una sezione o una pagina che mostri tutte le associazioni che il volontario segue.
- Implementare meccanismo per “unfollow” di associazioni
- Implementare una barra di ricerca per trovare associazioni per nome.

## **User Story 19: Visualizzazione lista di eventi creati da associazioni seguite**

Come volontario, voglio visualizzare l'elenco degli eventi creati dalle associazioni che segue, perché desidero rimanere informato sugli eventi pertinenti alle mie preferenze.

### **Criteri di Accettazione:**

- Il volontario deve poter visualizzare un elenco degli eventi creati dalle associazioni che segue.
- L'elenco deve includere almeno le seguenti informazioni per ciascun evento: Immagine ,Titolo dell'evento, Data e ora di inizio e fine dell'evento, Luogo dell'evento.
- Il volontario deve poter dirigersi alla pagina dell'evento.
- Il volontario deve poter filtrare gli eventi per data.

- Il volontario deve poter cercare un evento per titolo.

**Task:**

- Creazione dell'interfaccia per visualizzare gli eventi delle associazioni seguite
- Recuperare e mostrare gli eventi dal database.
- Implementare filtro per data.
- Implementare una barra di ricerca per trovare eventi specifici.
- Implementare bottone di reindirizzamento alla pagina dell'evento.

## User flow

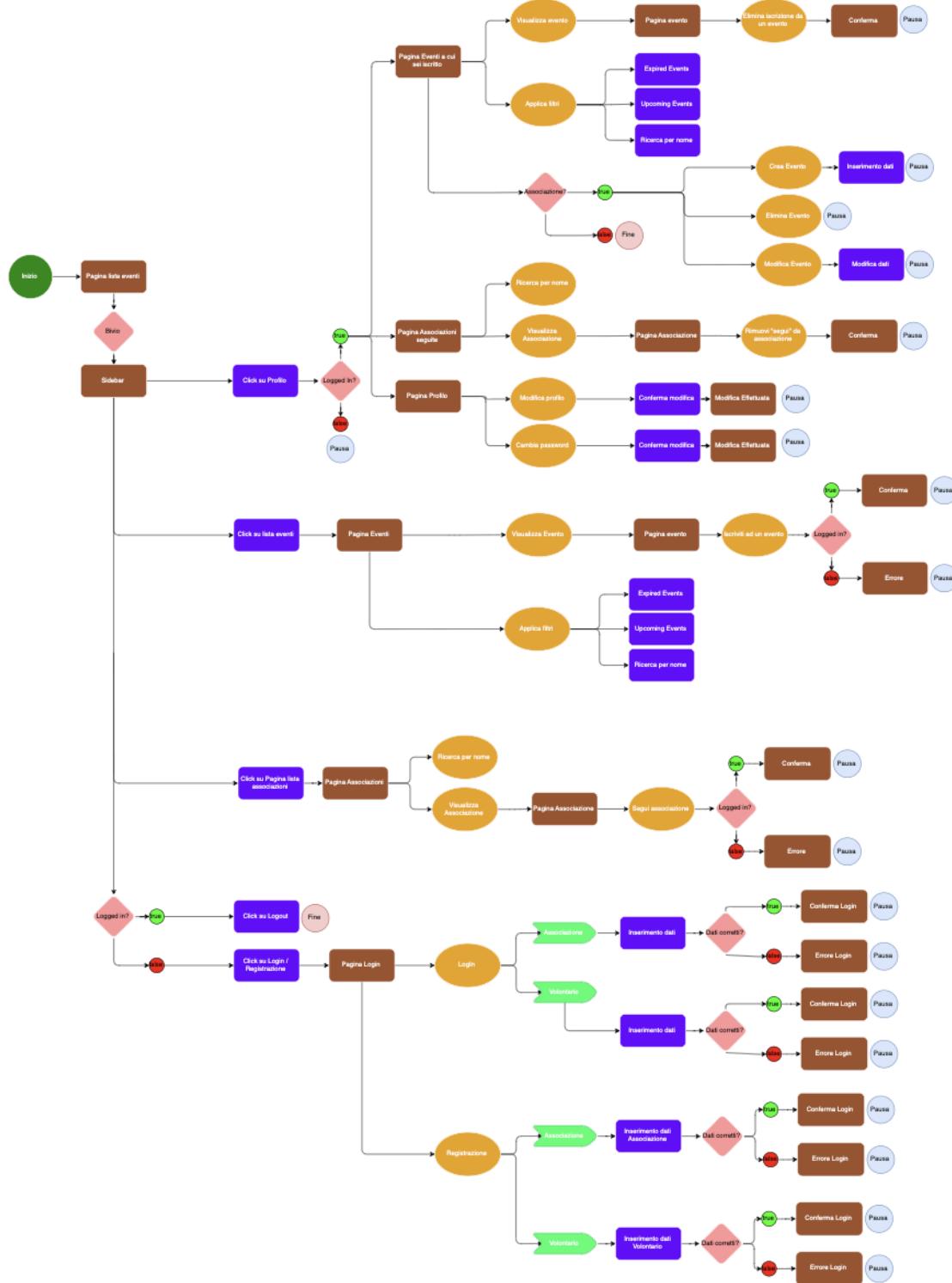
In questa sezione viene illustrato lo **userflow** dell'applicazione, evidenziando le azioni che gli utenti possono compiere nell'implementazione descritta in dettaglio in questo documento. Di seguito, è riportata una legenda dei vari componenti utilizzati nello userflow:



Nello userflow sono presenti due bivii: uno basato sulle scelte dell'utente (quello senza true/false) e uno che dipende dallo stato del sistema.

Il componente “Azione possibile” rappresenta le azioni che possono essere eseguite all’interno di una pagina, mentre il componente “Soggetto” è stato utilizzato per distinguere i due tipi di account presenti nel sistema: volontario e associazione.

In seguito è allegato uno screenshot dello userflow. In alternativa, è possibile visualizzarlo al seguente link: [Visualizza lo userflow](#).



## Web APIs

Le API sono state documentate secondo lo standard **OpenAPI 3.0**, e la documentazione è stata creata con **Swagger.js**. È possibile consultarla qui: [Documentazione API](#)

La specifica delle API è disponibile nel file **swagger.json**, accessibile al seguente link: [Specifiche API \(swagger.json\)](#)

## Implementation

L'applicazione è stata sviluppata utilizzando le seguenti tecnologie:

- **Node.js** – runtime JavaScript lato server.
- **Vue.js** – framework frontend per la creazione di interfacce utente reattive.
- **MongoDB** – database NoSQL orientato ai documenti.

Queste tecnologie sono state scelte per la loro semplicità e familiarità, in quanto trattate durante il corso. Inoltre, **Node.js** è uno dei runtime JavaScript più diffusi, offrendo un ampio supporto e una vasta gamma di librerie.

## Repository organization

Il codice del progetto è disponibile all'indirizzo: [Volontarion](#). I commit sono visibili nella storia del progetto.

Per organizzare al meglio il lavoro, abbiamo creato un Organization su GitHub e suddiviso il progetto in tre repository separate: frontend, backend e deliverables.

## Organizzazione backend

```
project-root/
  src/
    controllers/
      ...
    models/
      ...
    routes/
      ...
    middlewares/
      ...
    utils/
      ...
    app.js
    index.js
  ...
  .env
  config.json
  .gitignore
  package.json
  swagger.json
  README.md
```

Nella cartella **controllers** sono presenti tutte le funzioni richiamate dagli endpoint. La cartella **models** contiene i modelli utilizzati da Mongoose, mentre **routes** gestisce la suddivisione dei vari endpoint.

I **middleware** includono la gestione dell'autenticazione tramite JWT e un middleware di debug che calcola la dimensione della request.

Infine, la cartella **utils** raccoglie funzioni utili per il debugging, come il logger.

## Organizzazione frontend

```
frontend/
  src/
    assets/
      ...
    components/
      account/
        profilo/
          ...
        eventi/
          ...
        login/
          ...
        registrazione/
          ...
        util/
          ...
      ...
    views/
      ...
    router/
      index.js
    store/
      index.js
    endpoint.js
    main.js
    App.vue
  public/
    favicon.ico
    images/
      ...
  .env
  .gitignore
  package.json
  vite.config.js
  README.md
  node_modules/
```

Nella cartella **components** sono presenti le diverse pagine dell'applicazione, suddivise in sezioni per account, eventi, login, registrazione e componenti riutilizzabili.

## Gestione Workflow

Abbiamo suddiviso il lavoro nel seguente modo: abbiamo effettuato oltre 90 commit per il backend e più di 80 per il frontend, seguendo lo standard **Conventional Commits** ([conventionalcommits.org](https://conventionalcommits.org)).

La differenza nel numero di commit è dovuta ad errori nel merge, differenze negli ambienti di sviluppo, problemi con le dipendenze e modifiche necessarie per il deploy. Per questo motivo, sono stati creati numerosi commit di piccole dimensioni per correggere dettagli legati al deploy, mentre i commit più grandi contengono principalmente l'aggiunta di nuove funzionalità all'applicazione.

## Dependencies backend

### Moduli esterni di npm:

- **bcrypt**: Serve per l'hashing sicuro delle password.
- **cookie-parser**: Middleware per analizzare i cookie nelle richieste HTTP.
- **cors**: Middleware per abilitare il supporto CORS (Cross-Origin Resource Sharing).
- **dotenv**: Permette la gestione delle variabili d'ambiente tramite un file .env.
- **express**: Framework web per la gestione del server HTTP.
- **express-jwt**: Middleware per l'autenticazione basata su JSON Web Tokens (JWT).
- **express-session**: Middleware per gestire le sessioni utente.
- **https**: Modulo per effettuare richieste HTTPS in Node.js.
- **jsonwebtoken**: Libreria per creare e verificare JSON Web Tokens (JWT).
- **mongoose**: ODM (Object Data Modeling) per gestire MongoDB in modo strutturato.
- **node-mailjet**: SDK per inviare email tramite il servizio Mailjet.
- **pino-pretty**: Formatter per rendere più leggibili i log di **Pino**.

- **swagger-jsdoc**: Generatore di documentazione OpenAPI (Swagger) dai commenti JSDoc.
- **swagger-ui-express**: Middleware per visualizzare la documentazione API in un'interfaccia web Swagger.

#### Dependencies per il development:

- **nodemon**: Tool per riavviare automaticamente il server Node.js quando i file cambiano.

### Dependecies frontend

- **@fortawesome/fontawesome-svg-core**: Il cuore della libreria FontAwesome per la gestione delle icone SVG.
- **@fortawesome/free-brands-svg-icons**: Raccolta di icone relative ai brand (es. Facebook, Twitter, GitHub) di FontAwesome.
- **@fortawesome/free-regular-svg-icons**: Set di icone standard della libreria FontAwesome.
- **@fortawesome/free-solid-svg-icons**: Set di icone solide della libreria FontAwesome.
- **@fortawesome/vue-fontawesome**: Integrazione di FontAwesome con Vue.js per l'utilizzo delle icone nel frontend.
- **@primevue/themes**: Raccolta di temi per **PrimeVue**, un framework UI per Vue.js.
- **body-parser**: Middleware per analizzare il corpo delle richieste HTTP, utile per gestire JSON o dati form-url-encoded.
- **cors**: Middleware per abilitare il supporto CORS (Cross-Origin Resource Sharing) e consentire comunicazioni tra domini diversi.
- **dotenv**: Permette la gestione delle variabili d'ambiente tramite un file .env.
- **vue**: Il framework Vue.js per la creazione di interfacce utente reattive.
- **vue-router**: Gestore delle rotte per Vue.js, essenziale per le applicazioni **Single Page Application (SPA)**.

#### Dependencies per il development:

- **@vitejs/plugin-vue**: Plugin per abilitare il supporto di Vue.js all'interno di Vite.

- **autoprefixer**: Tool che aggiunge automaticamente i prefissi ai CSS per garantire compatibilità con diversi browser.
- **daisyui**: Libreria di componenti UI basata su **Tailwind CSS** per uno sviluppo frontend più rapido.
- **postcss**: Strumento per trasformare i CSS con plugin JavaScript.
- **tailwindcss**: Un framework CSS basato su classi utility per uno sviluppo rapido e modulare.
- **vite**: Un build tool veloce e leggero per applicazioni Vue.js.
- **vite-plugin-vue-devtools**: Plugin che abilita strumenti di sviluppo avanzati per Vue.js con Vite.

## Database

Per la gestione dei dati sono stati definiti 3 modelli (con mongoose): Volontario, Associazione ed Evento.

```
const VolontarioSchema = new mongoose.Schema({
  name: { type: String, required: true },
  surname: { type: String, required: true },
  age: { type: Number, min: 0, max: 200, required: true },
  email: { type: String, required: true, unique: true },
  phone: { type: String, required: false },
  password: { type: String, required: true },
  followedAssociations: { type: [String], required: false },
  subscribedEvents: { type: [String], required: false },
  description: { type: String, required: false },
  experience: { type: String, required: false },
  skills: { type: [String], required: false },
  profilePicture: { type: String, required: false },
});
```

Qui sono presenti tutti i campi del volontario, che sono anche definiti (vedere D2 ), followedAssociations e subscribed events sono degli array che contengono gli id delle relative

associazioni ed eventi. C'è un controllo sull'input di age, e anche su email che deve essere unica tra i volontari. La profilePicture viene convertita in base64 e poi salvata come string.

```
const AssociazioneSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  phone: { type: String, required: true },
  password: { type: String, required: true },
  description: { type: String, required: false },
  objectives: { type: String, required: false },
  profilePicture: { type: String, required: false },
  createdEvents: { type: [String], required: false },
  subscribedVolunteers: { type: [String], required: false },
});
```

Anche qui subscribedVolunteers contiene un array di ID dei volontari iscritti all'associazione. Poi è presente un controllo sull'unicità della mail tra le associazioni. profilePicture è codificata in base64.

```
const EventoSchema = new mongoose.Schema({
  name: { type: String, required: true },
  hostAssociation: { type: String, required: false },
  hostAssociationname: { type: String, required: false },
  startDateTime: { type: String, required: true },
  endDateTime: { type: String, required: true },
  place: { type: String, required: true },
  picture: { type: String, required: false },
  description: { type: String, required: false },
  subscribedVolunteers: { type: [String], required: false },
});
```

Qui subscribedVolunteers contiene un array di ID dei volontari iscritti a quel determinato evento. Picture è codificata in base64.

## Note sullo sviluppo

- La conferma di registrazione via email non è completamente implementata: l'email viene inviata, ma non contiene i dati di riepilogo. Riferimento D2, use case 2.
- L'applicazione è responsive, ma non è progettata per l'uso su dispositivi mobili, quindi l'esperienza utente potrebbe non essere ottimale su questi device.
- A causa di vincoli di tempo, la funzione di eliminazione dell'account non è stata implementata nel sistema.

## Testing

Per ragioni di tempo, non è stato possibile implementare il testing con Jest. Di conseguenza, i test presentati di seguito sono stati eseguiti manualmente e non in modo automatizzato.

N. Test case	Descrizione	Test data	Pre Condizioni	Dipendenze	Risultato atteso	Risultato riscontrato	Note
1	Creazione di un account	Inserimento di tutti i dati richiesti, scritti correttamente			Viene creato l'account, il sistema manda un messaggio di conferma	= risultato atteso	
1.1	Creazione di un account ma la mail esiste già	Inserimento di tutti i dati richiesti, scritti correttamente			Il sistema risponde con un messaggio di errore	= risultato atteso	

1.2	Creazione di un account ma non sono stati inseriti tutti i dati obbligatori	Mancano dei dati obbligatori, come email o password			Il sistema risponde con un messaggio di errore	= risultato atteso	
2	Login con dati validi	Inserisco email e password	Dati dell'account già presenti del sistema	1	Vieni autenticato ed il sistema manda un messaggio di conferma	= risultato atteso	
2.1	Login con email o password sbagliati	Email o password sbagliati			Il sistema manda un messaggio di errore	= risultato atteso	
3	Iscrizione ad un evento	- -	Essere loggati nel sistema come Volontario	1, 2	L'evento viene aggiunto agli eventi dell'account e il sistema manda un messaggio di conferma	= risultato atteso	
3.1	Iscrizione ad un evento senza essere loggati	- -			Il sistema non lo permette	= risultato atteso	Non è presente il pulsante per l'iscrizione
4	Iscrizione ad un'associazione	- -	Essere loggati nel sistema come Volontario	1,2	L'associazione viene aggiunta alle associazioni seguite dell'account e il sistema manda un messaggio di conferma	= risultato atteso	
4.1	Iscrizione ad un'associazione senza essere loggati	- -			Il sistema non lo permette	= risultato atteso	Non è presente il pulsante per l'iscrizione

6	Creazione di un evento	- - -	Essere loggati nel sistema come Associazione	1,2	L'evento viene creato e aggiunto agli eventi di un'associazione, il sistema manda un messaggio di conferma	<b>= risultato atteso</b>	
6.1	Creazione di un evento, ma data di fine è minore di data di inizio	- - -	Essere loggati nel sistema come Associazione	1,2	L'evento non viene creato, viene inviato un messaggio di errore dal sistema	<b>L'evento viene creato ugualmente</b>	

## Frontend

L'applicazione frontend è organizzata in diverse sezioni, accessibili tramite la sidebar. Alcune pagine sono disponibili senza necessità di autenticazione, tra cui la lista degli eventi, la lista delle associazioni e la pagina di login e registrazione.

Le restanti sezioni sono accessibili solo agli utenti autenticati e variano in base al ruolo. I volontari possono accedere alla propria pagina profilo, alla sezione degli eventi a cui si sono iscritti e alla pagina delle associazioni seguite. Le associazioni, invece, possono visualizzare la propria pagina profilo e la sezione dedicata agli eventi creati.

## Sezioni comuni

### Pagina eventi

The screenshot shows the 'Events' section of the VolontariOn application. On the left, there is a sidebar with a user profile icon and the following menu items:

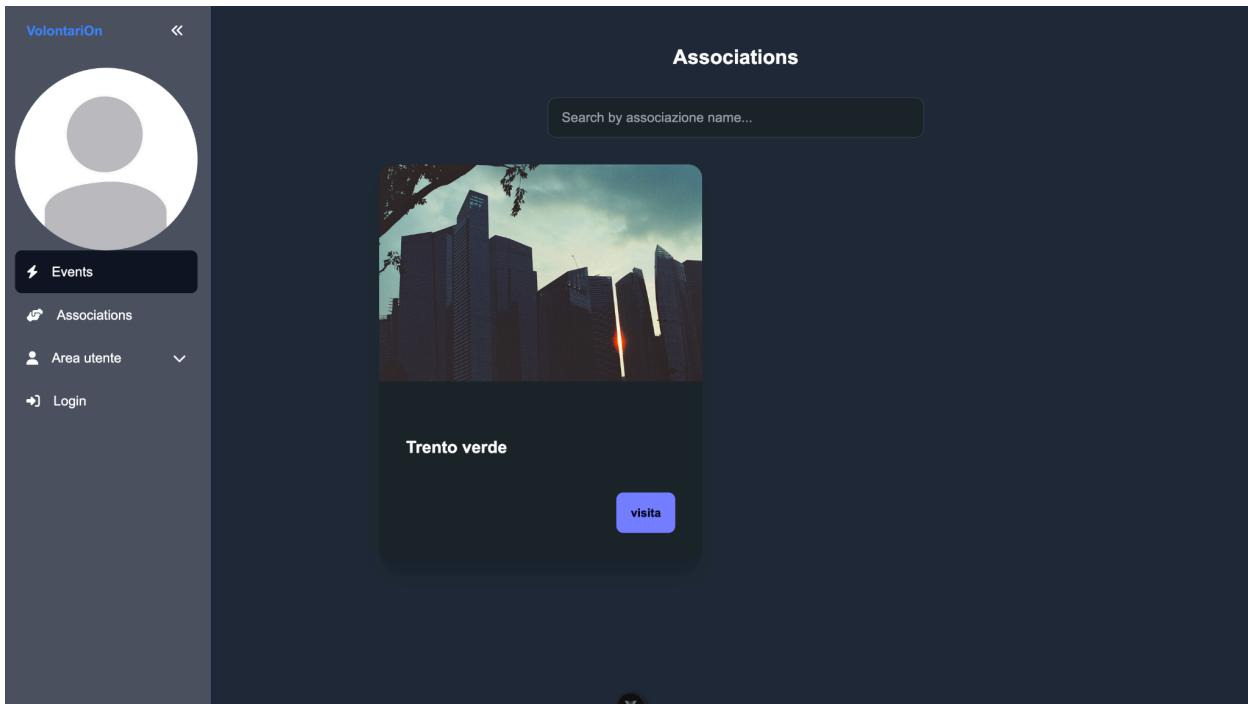
- Events
- Lista associazioni
- Area utente
- Login

The main area is titled 'Events' and features a search bar labeled 'Search by event name...' and a button labeled 'All Events'. Below these are two event cards:

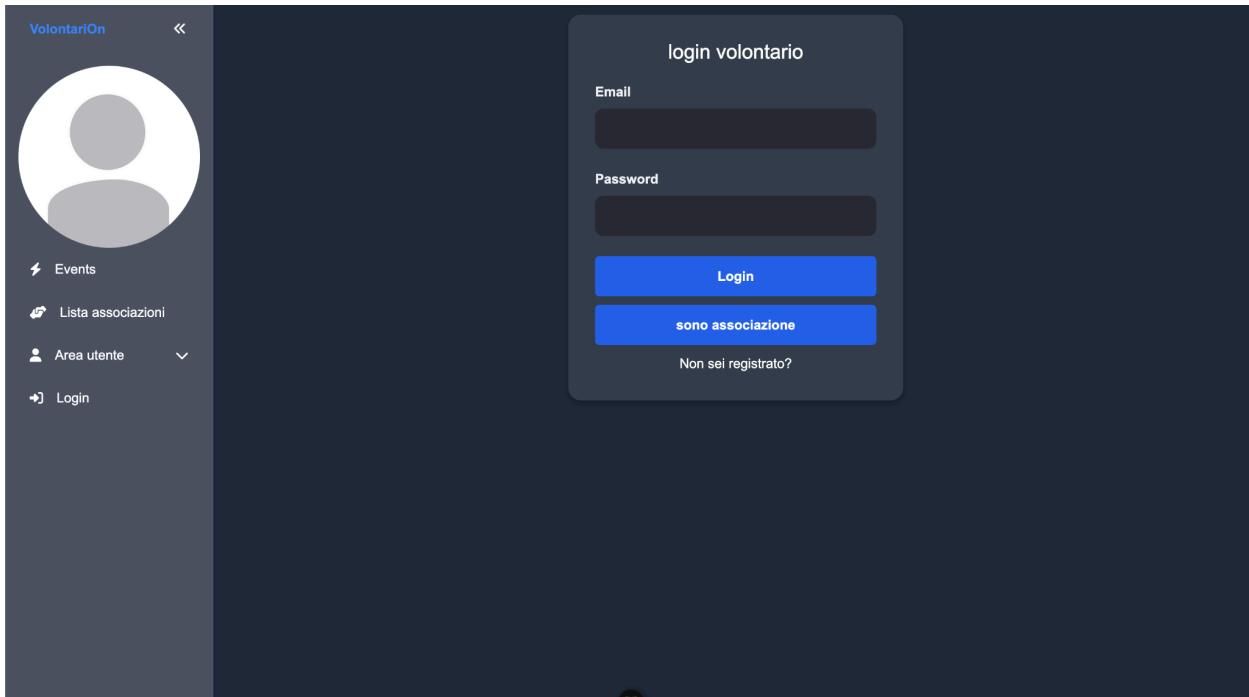
**Puliamo la città**  
startDateTime: 2025-02-22T15:00  
endDateTime: 2025-02-23T18:00  
Location: Trento  
Associazione:  
[Visualizza](#)

**Pianta un Albero**  
startDateTime: 2025-02-24T12:00  
endDateTime: 2025-02-25T15:00  
Location: Trento - Albere  
Associazione:  
[Visualizza](#)

## Pagina associazioni



**Pagina login:** si può scegliere se autenticarsi come volontario o come associazione



## Pagina Registrazione volontario

The screenshot shows the 'VolontariOn' web application interface. On the left, there's a sidebar with a user profile icon and links for 'Events', 'Lista associazioni', 'Area utente', and 'Login'. The main content area has two tabs at the top: 'Registrazione Volontari' (selected) and 'Registrazione Associazioni'. The 'Registrazione Volontari' tab contains a form for creating a volunteer profile. It includes fields for 'Foto Profilo' (with a placeholder 'Browse... No file selected.'), 'Nome' and 'Cognome' (both redacted), 'Eta' (set to 5), 'Email' (redacted), 'Telefono' and 'Password' (both redacted), 'Descrizione' (redacted), and 'Esperienza' (redacted). Below these, there's a section for adding skills: 'Aggiungi le tue competenze' with a 'Skill:' label and a text input field 'Inserisci una competenza' with an 'Add' button. At the bottom right of the main form is a large blue 'Registrati' button.

## Pagina registrazione Associazione

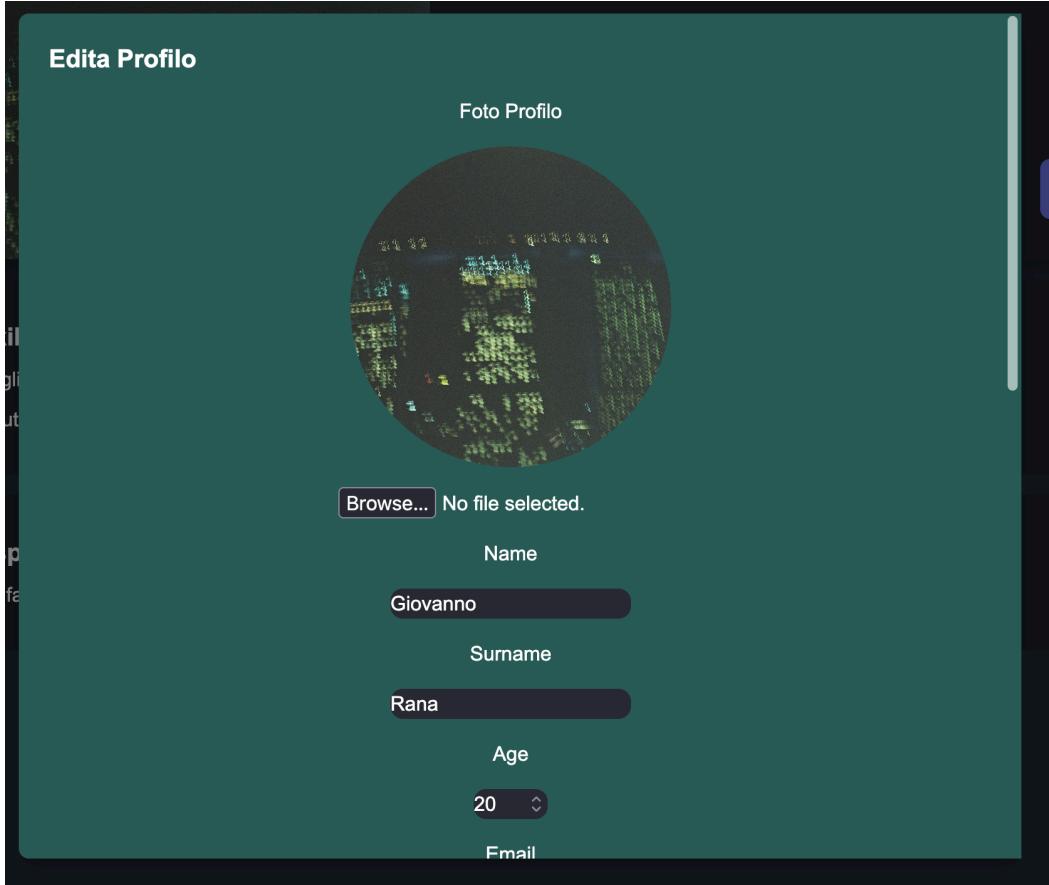
The screenshot shows the 'Associazione' registration form on the VolontariOn platform. On the left, there's a sidebar with a placeholder profile picture and links for 'Events', 'Lista associazioni', 'Area utente', and 'Login'. The main area has two tabs at the top: 'Registrazione Volontari' (grayed out) and 'Registrazione Associazioni' (blue). The registration form contains fields for 'Foto Profilo' (with a 'Browse...' button showing 'No file selected.'), 'Name', 'Email', 'Phone', 'Password', 'Descrizione', and 'Obiettivi'. A large blue 'Registrati' button is at the bottom.

## Sezioni Volontario

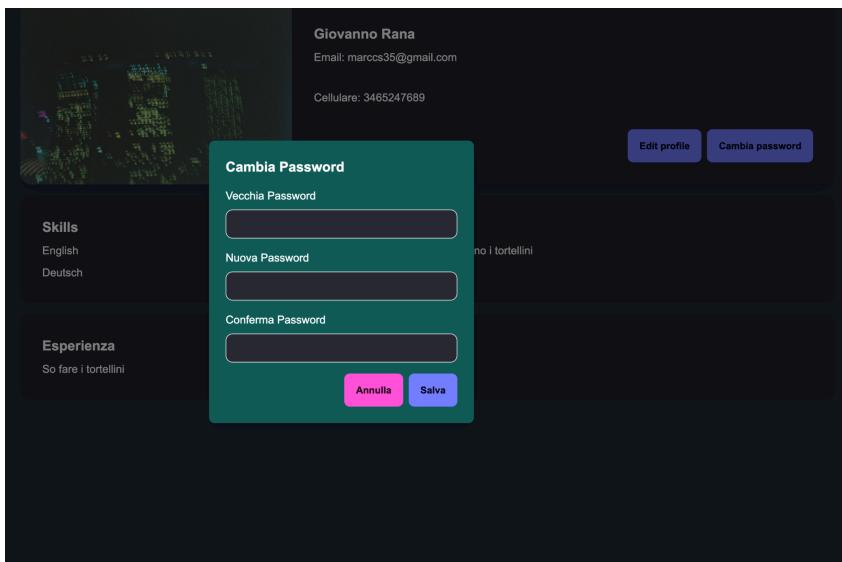
### Pagina profilo

The screenshot shows a volunteer profile page. At the top, there's a placeholder image of a city skyline at night. Below it, the user's name 'Giovanno Rana' is displayed, along with their email 'Email: marcos35@gmail.com' and phone number 'Cellulare: 3465247689'. There are two blue buttons: 'Edit profile' and 'Cambia password'. The page is divided into sections: 'Skills' (English, Deutsch), 'Descrizione' (Ciao! Sono Giovanni, mi piacciono i tortellini), and 'Esperienza' (So fare i tortellini).

## Pagina Profilo - edit profile



## Pagina profilo - change password



## Pagina “I miei eventi”

The screenshot shows a dark-themed web application interface. On the left is a vertical sidebar with icons for navigation: double arrows, user profile, lightning bolt, gear, person, double arrows, and a square. The main area has a header "My Events". Below it is a search bar "Search by event name...". A button "All Events" is visible. A single event card is displayed, featuring a thumbnail image of a modern building with greenery. The event details are: **Puliamo la città**, startDateTime: 2025-02-22T15:00, endDateTime: 2025-02-23T18:00, Location: Trento. Buttons at the bottom of the card are "Disiscriviti" (with a red X icon) and "Visualizza".

## Pagina “associazioni seguite”

The screenshot shows a dark-themed web application interface. On the left is a vertical sidebar with icons for navigation: double arrows, user profile, lightning bolt, gear, person, double arrows, and a square. The main area has a header "Associazioni iscritte". Below it is a search bar "Search by associazione name...". A single association card is displayed, featuring a thumbnail image of a city skyline at night. The association details are: **Trento verde**. Buttons at the bottom of the card are "visita" (blue button) and "unsubscribe" (blue button).

# Sezioni Associazione

## Pagina profilo



**Trento verde**  
Email : trentoverde@gmail.com  
Cellulare : 3452679803

[Edit profile](#) [Cambia password](#)

**Descrizione**  
Trento Verde è un'associazione di volontariato impegnata nella tutela dell'ambiente e nella promozione della sostenibilità attraverso iniziative di sensibilizzazione, cura del verde e partecipazione attiva della comunità.

**Obiettivi**  
Tutela ambientale: Proteggere e valorizzare le aree verdi urbane e naturali. • Sensibilizzazione: Promuovere la cultura della sostenibilità e il rispetto per l'ambiente. • Partecipazione attiva: Coinvolgere la comunità in iniziative di volontariato ecologico. • Riduzione dell'inquinamento: Favorire pratiche sostenibili come il riciclo e la mobilità green.

## Pagina "i miei eventi"

### My Events

[All Events](#)

[\*\*+ Create Event\*\*](#)



**Puliamo la città**  
startDateTime: 2025-02-22T15:00  
endDateTime: 2025-02-23T18:00  
Location: Trento

[✖ Elimina Evento](#)



placeholder

**Pianta un Albero**  
startDateTime: 2025-02-24T12:00  
  
endDateTime: 2025-02-25T15:00  
  
Location: Trento - Albere

[✖ Elimina Evento](#)



**Puliamo la città**  
startDateTime: 2025-02-22T15:00  
endDateTime: 2025-02-23T18:00  
Location: Trento

[✖ Elimina Evento](#)

[📝 Modifica evento](#)

[👤 Visualizza](#)

[✉️ Visualizza Iscritti](#)



placeholder

**Pianta un Albero**  
startDateTime: 2025-02-24T12:00  
  
endDateTime: 2025-02-25T15:00  
  
Location: Trento - Albere

[✖ Elimina Evento](#)

[📝 Modifica evento](#)

[👤 Visualizza](#)

[✉️ Visualizza Iscritti](#)

## Pagina creazione evento

**My Events**

**Crea evento**

Foto Profilo  
Profile Picture  
 No file selected.

Nome

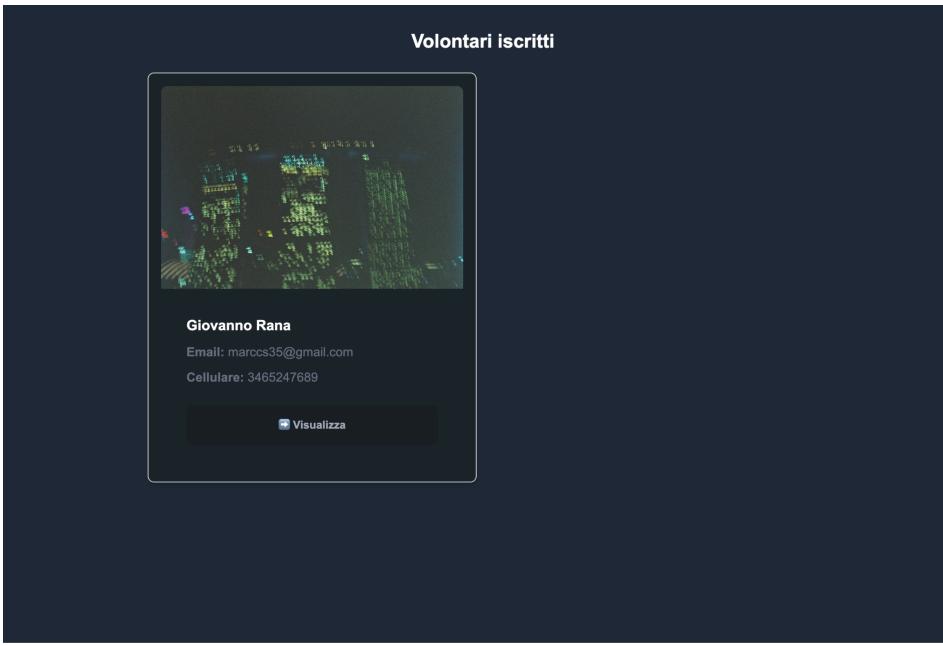
Data e tempo di inizio  
 mm/dd/yyyy, --:-- --

Data e tempo di fine  
 mm/dd/yyyy, --:-- --

Luogo

Descrizione

**Pagina “visualizza iscritti ad un evento”:** cliccando su “visualizza iscritti” nella card di un evento si arriva a questa pagina



# Deployment

L'applicazione web è composta da un backend e un frontend, entrambi ospitati su **Render**.

Il **backend** è hostato all'indirizzo: <https://volontarion-ingegneria-del-software.onrender.com>.

La documentazione degli endpoint può essere consultata su:

<https://volontarion-ingegneria-del-software.onrender.com/api-docs>.

Il **frontend** è ospitato su: <https://volontarionfrontend.onrender.com>.

Il deployment è stato effettuato manualmente. Sono stati eseguiti più deploy per correggere errori legati alla configurazione della comunicazione tra frontend e backend. I dettagli delle modifiche apportate durante i vari rilasci sono documentati nei relativi commit con la dicitura "*deploy: ...*".

Per provare l'applicazione sono disponibili 4 account:

## Account volontario:

Mail: [test@gmail.com](mailto:test@gmail.com) password: pwd

Mail: [marccs35@gmail.com](mailto:marccs35@gmail.com) password: pwd

## Account Associazione:

Mail: [test2@gmail.com](mailto:test2@gmail.com) password: pwd

Mail: [trentoverde@gmail.com](mailto:trentoverde@gmail.com) password: pwd