Document: *Sviluppo*

Revision: *1.1*

UNIVERSITY OF TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Progetto:

OasiTN

Titolo del documento:

Analisi e progettazione

Document Info

Doc. Name	*D3-oasitnSviluppo*	Doc. Number	D3
Description	Documento di sviluppo dell'applicazione		

INDICE

Sommario

Scopo del documento

Il presente documento riporta tutte le informazioni necessarie per lo sviluppo di una parte dell'applicazione OasiTN. In particolare, presenta tutti gli artefatti necessari per realizzare i servizi di gestione dei libri e dei prestiti con l'applicazione OasiTN.
Partendo dalla descrizione delle User Stories, il documento prosegue con gli User Flow, e la presentazione delle web APIs del servizio web, proseguendo con l'organizzazione del codice, il modello dati, e ulteriori informazioni su testing. Infine, una sezione dedicata al front-end e una agli aspetti di deployment.

1. User stories

Nel presente capitolo vengono riportate le user stories individuate del progetto OasiTN partendo dai requisiti individuati nel documento precedente (D1).

User Story 1: Visualizzazione del Tutorial di Navigazione

Come utente, voglio poter visualizzare un tutorial interattivo, in modo da comprendere facilmente il funzionamento della piattaforma.

Criteri di accettazione:

- L'utente può avviare il tutorial cliccando su "Help" nella schermata iniziale.
- Il sistema oscura progressivamente le componenti della pagina, fornendo una descrizione interattiva.
- L'utente può avanzare nel tutorial o interromperlo in qualsiasi momento.
- Il tutorial deve funzionare correttamente su tutti i browser supportati.

Tasks:

1. Creare il pulsante "Help" nella UI.
2. Implementare la logica per oscurare progressivamente gli elementi della pagina.
3. Aggiungere il testo descrittivo nelle varie fasi del tutorial.
4. Testare il tutorial su diversi browser per verificarne la compatibilità.

User Story 2: Mappa Interattiva

Come utente, voglio poter visualizzare i parchi su una mappa interattiva, in modo da ottenere facilmente informazioni dettagliate su di essi.

Criteri di accettazione:

- La mappa deve mostrare dei marker per ciascun parco.
- Cliccando su un marker, si deve aprire un pop-up con le informazioni base del parco.
- L'utente può cliccare sul pop-up per ottenere ulteriori dettagli.
- La mappa deve essere caricata correttamente su tutti i browser supportati.

Tasks:

1. Integrare la mappa con marker interattivi.
2. Collegare i marker ai dati del database dei parchi.
3. Implementare la logica per visualizzare il pop-up con le informazioni.
4. Testare il corretto funzionamento della mappa su browser e dispositivi diversi.

User Story 3: Ricerca Parchi con Filtri

Come utente, voglio poter filtrare i parchi per categorie specifiche, in modo da trovare rapidamente quelli che rispondono alle mie esigenze.

Criteri di accettazione:

- L'utente può selezionare filtri dal menù della pagina.
- Il sistema aggiorna la lista dei parchi in base ai filtri selezionati.
- L'utente può rimuovere i filtri e tornare alla visualizzazione completa.

Tasks:

1. Creare il menù dei filtri con le categorie disponibili.
2. Implementare la logica di filtraggio dei parchi.
3. Collegare il sistema di filtri al database.
4. Testare il corretto funzionamento della ricerca con filtri multipli.

User Story 4: Ricerca Parchi tramite Barra di Ricerca

Come utente, voglio poter cercare un parco inserendo il nome nella barra di ricerca, in modo da trovarlo rapidamente.

Criteri di accettazione:

- L'utente può inserire il nome o una parte di esso nella barra di ricerca.

- Il sistema mostra i parchi corrispondenti in tempo reale.
- Se il parco non è presente, viene mostrato un messaggio informativo.

Tasks:

1. Creare la barra di ricerca con suggerimenti dinamici.
2. Implementare la logica di ricerca parziale e completa.
3. Collegare la barra di ricerca al database.
4. Gestire i casi di errore o assenza di risultati.

User Story 5: Segnalazioni Utente

Come utente, voglio poter segnalare errori o problemi relativi a un parco, in modo da contribuire al miglioramento delle informazioni disponibili.

Criteri di accettazione:

- L'utente può cliccare sul pulsante "Segnala un problema".
- Viene visualizzato un form per la segnalazione.
- Il form deve permettere di selezionare il parco e descrivere il problema.
- Il sistema salva la segnalazione nel database per l'analisi da parte dell'admin.

Tasks:

1. Creare il pulsante "Segnala un problema".
2. Implementare il form di segnalazione.
3. Collegare il form al database.
4. Implementare una notifica di conferma all'utente.

User Story 6: Gestione Parchi (Admin)

Come amministratore, voglio poter aggiungere, modificare e rimuovere i parchi dal sistema, in modo da garantire che le informazioni siano sempre aggiornate.

Criteri di accettazione:

- L'admin può accedere a una dashboard di gestione.
- Può aggiungere un nuovo parco compilando un modulo.
- Può modificare i dettagli di un parco esistente.
- Può eliminare un parco dal sistema.

Tasks:

1. Creare la dashboard di gestione per l'admin.
2. Implementare il modulo di inserimento nuovi parchi.

---

3. Implementare la funzionalità di modifica ed eliminazione dei parchi.
4. Collegare la gestione parchi al database.

User Story 7: Gestione Segnalazioni (Admin)

Come amministratore, voglio poter visualizzare e gestire le segnalazioni ricevute dagli utenti, in modo da correggere eventuali problemi.

Criteri di accettazione:

- L'admin può accedere alla sezione "Segnalazioni".
- Può visualizzare la lista delle segnalazioni attive.
- Può eliminare segnalazioni risolte o irrilevanti.

Tasks:

1. Creare l'interfaccia per la visualizzazione delle segnalazioni.
2. Implementare il recupero delle segnalazioni dal database.
3. Aggiungere la possibilità di eliminare le segnalazioni gestite.
4. Testare la gestione delle segnalazioni su diversi scenari.

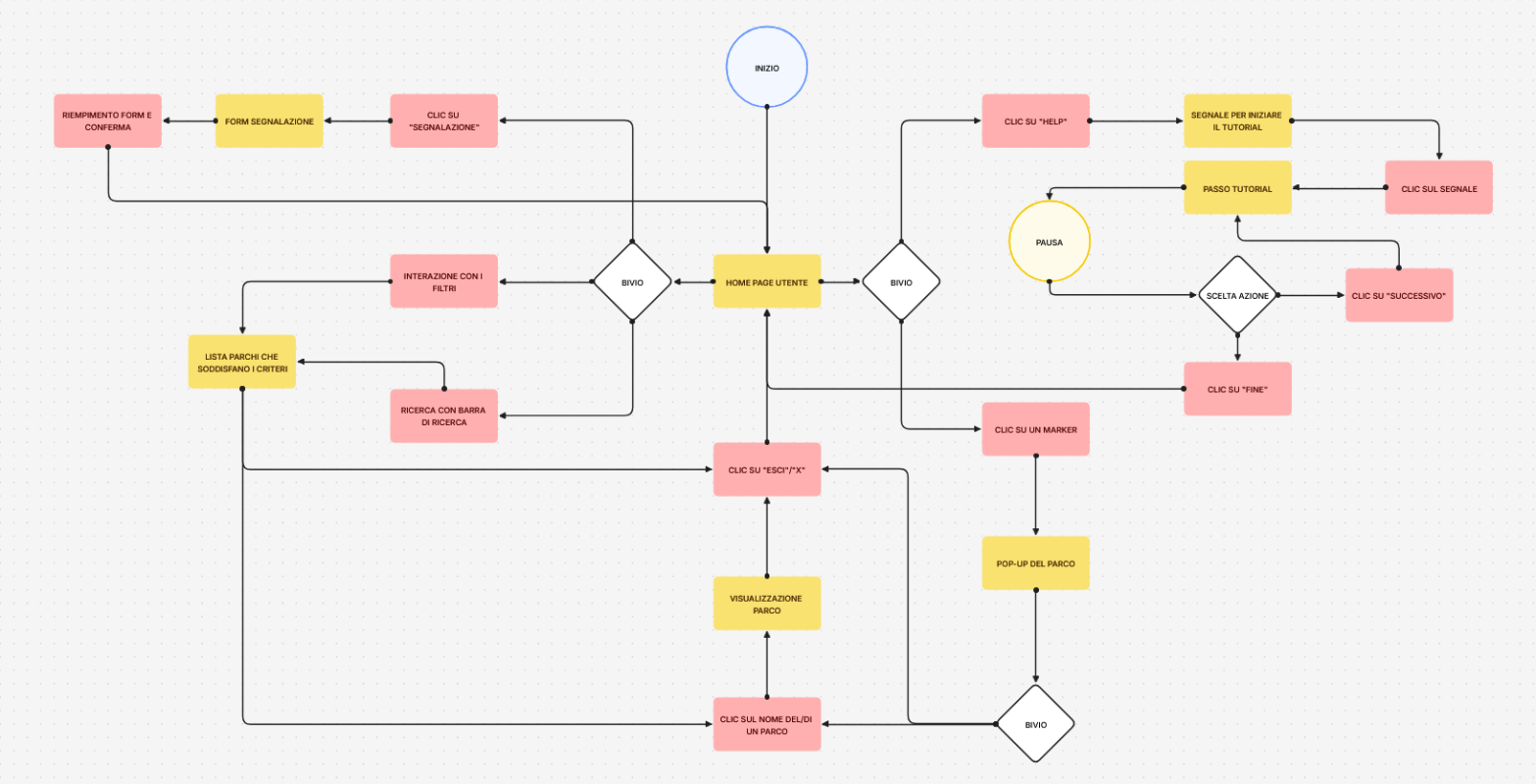
2. User Flow

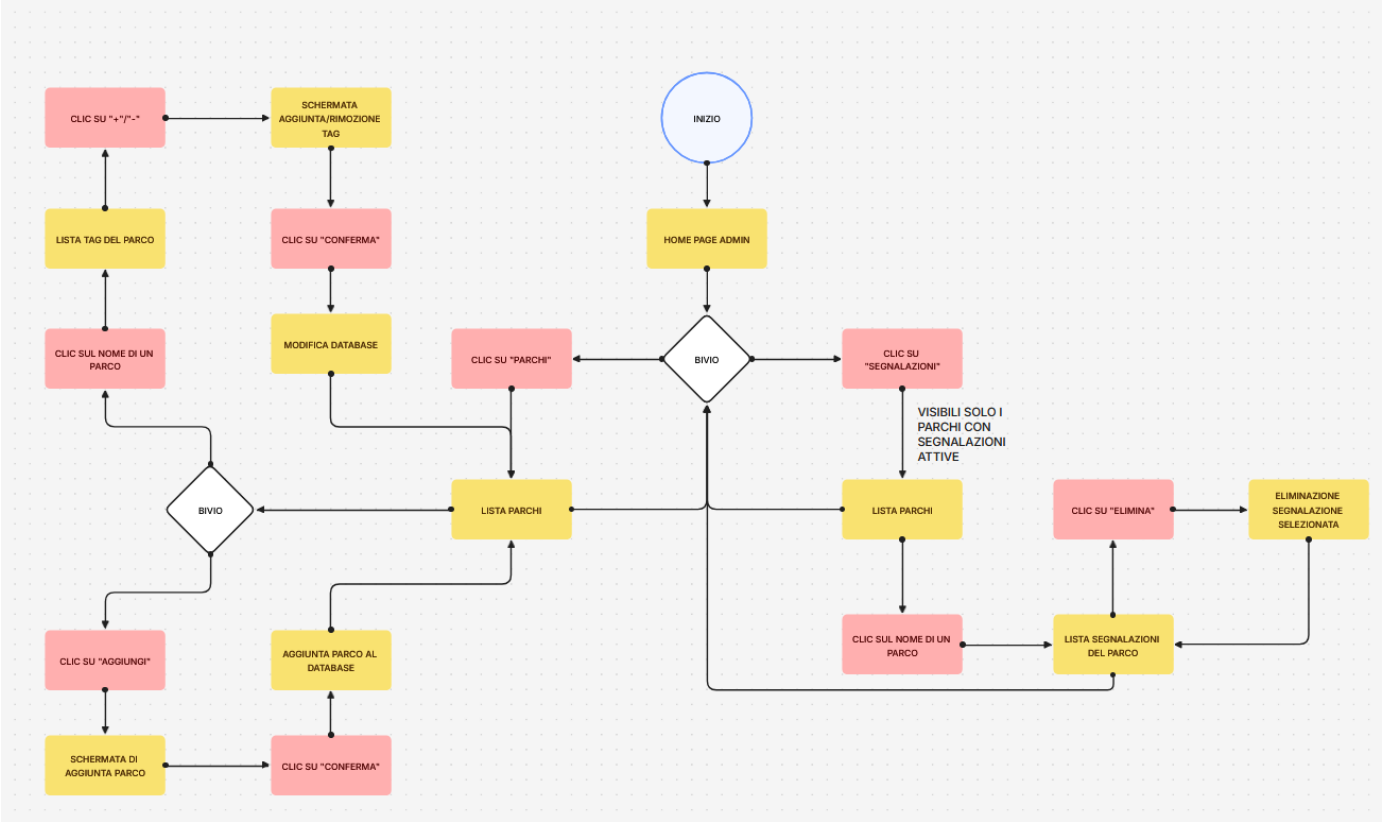
In questa sezione del documento di sviluppo riportiamo gli “user flows” per i ruoli di utente e admin della nostra applicazione.
La figura 0 presenta la legenda dei diagrammi.
La figura 1 presenta lo user flow dell’utente.
La figura 2 presenta lo user flow dell’admin.

Figura 0


LEGENDA
INIZIO
PAUSA
ARRIVO
AZIONI
RISPOSTA DEL SISTEMA
BIVIO

Figura 1


INIZIO
RIEMPIMENTO FORM E CONFERMA
FORM SEGNALAZIONE
CLIC SU "SEGNALAZIONE"
CLIC SU "HELP"
SEGNALE PER INIZIARE IL TUTORIAL
PASSO TUTORIAL
CLIC SUL SEGNALE
PAUSA
INTERAZIONE CON I FILTRI
BIVIO
HOME PAGE UTENTE
BIVIO
SCELTA AZIONE
CLIC SU "SUCCESSIVO"
LISTA PARCHI CHE SODDISFANO I CRITERI
RICERCA CON BARRA DI RICERCA
CLIC SU "FINE"
CLIC SU UN MARKER
CLIC SU "ESCI"/"X"
POP-UP DEL PARCO
VISUALIZZAZIONE PARCO
CLIC SUL NOME DEL/DI UN PARCO
BIVIO



3. Web APIs

Le api sono state documentate secondo le specifiche openapi 3 e la documentazione è consultabile

La specifica delle API è disponibile nel repository al link https://github.com/AleFacciUnitn/OasiTN/blob/main/BackEnd/openapi/oas3.yaml.

Di seguito é riportato il file swagger compilato contenente tutte le API.

Admin - GET Operazioni di lettura per l'Admin

- GET `/admin/Crowding` Get all crowding data
- GET `/admin/Categoria` Get all categories
- GET `/admin/Categoria/{id}` Get category by ID
- GET `/admin/Tag` Get all tags
- GET `/admin/Tag/{id}` Get tag by ID
- GET `/admin/Parco` Get all parks
- GET `/admin/Parco/{id}` Get park by ID
- GET `/admin/Segnalazioni` Get all reports

Admin - POST Operazioni di creazione per l'Admin

- POST `/admin/Login` Admin login
- POST `/admin/Categoria` Add a new category
- POST `/admin/Tag` Add a new tag
- POST `/admin/Parco` Add a new park

Admin - DELETE Operazioni di eliminazione per l'Admin

- DELETE `/admin/Categoria/{id}` Delete category by ID
- DELETE `/admin/Tag/{id}` Delete tag by ID
- DELETE `/admin/Parco/{id}` Delete park by ID
- DELETE `/admin/Segnalazioni/{id}` Resolve report

Admin - PUT Operazioni di modifica per l'Admin

- PUT `/admin/Categoria/{id}` Update category by ID
- PUT `/admin/Tag/{id}` Update tag by ID
- PUT `/admin/Parco/{id}` Update park by ID
- PUT `/admin/Segnalazioni` Update report status

User - GET Operazioni di lettura per l'Utente

- GET `/user/init` Initial fetch for user

User - POST Operazioni di creazione per l'Utente

- POST `/user/Crowding` Add a new crowding data
- POST `/user/Segnalazioni` Add a new report

4. Implementation

L'applicazione è stata sviluppata utilizzando le seguenti tecnologie: NextJS e ExpressJS, per la gestione dei dati abbiamo utilizzato MongoDB. Lo stack tecnologico è stato motivato dal materiale fornito nel corso e dalle conoscenze pregresse degli strumenti di NextJS e ExpressJS.

4.1 Repository organization

Il codice del progetto, che si può trovare al in questa repository, è stato sviluppato secondo la seguente struttura:

- /Backend:
  - /exported_data — una cartella con dei data sample e le istruzioni per usarli
  - /openapi — documentazione delle api
  - /src:
    - /Controllers — contiene la logica della API
    - /middleware — contiene alcune funzioni di middleware
    - /models — contiene gli schema dei dati nel database
    - /routes — contiene la definizione degli endpoint delle api
- /Frontend:
  - /assets — immagini per il sito
  - /src/app: — contiene le pagine della sezione utente e della sezione admin
    - /admin — contiene le pagine della sezione admin

4.2 Branching strategy e organizzazione del lavoro

Lo sviluppo del progetto ha visto un'intensa collaborazione tra i membri del team, con una suddivisione chiara dei compiti e un'organizzazione efficace del lavoro. Abbiamo suddiviso le attività in modo da garantire un'evoluzione parallela delle componenti principali, ossia il backend e il frontend.

Per il versionamento del codice, abbiamo adottato la strategia GitFlow Workflow, che ci ha permesso di mantenere un'organizzazione chiara delle varie fasi dello sviluppo. In particolare:

Lo sviluppo iniziale del backend e del frontend è stato gestito su branch separate (backend, frontend).

Una volta completate le prime fasi di sviluppo e superati i primi test, queste branch sono state unite per favorire l'integrazione.
Le correzioni minori (minor fixes) sono state applicate direttamente sulla branch di sviluppo principale (develop). Per l'implementazione di nuove funzionalità, abbiamo creato branch

dedicate che, dopo essere state testate e integrate, sono state eliminate per mantenere il repository pulito e ordinato.

Abbiamo registrato un numero significativo di commit durante lo sviluppo, distribuiti tra i membri del team secondo le rispettive responsabilità.
Il numero di commit per membro può variare di numero principalmente perché alcuni membri hanno preferito commit frequenti e di piccole dimensioni, mentre altri hanno optato per commit più corposi e strutturati. Questo è stato rilevante in fase di testing, dove sono stati eseguiti molti commit per minor fixes.

Questa strategia ci ha permesso di lavorare in modo agile e collaborativo, garantendo un'integrazione graduale delle nuove funzionalità senza compromettere la stabilità del codice.

4.3 Dependecies

Il progetto npm si basa sui seguenti moduli esterni:

- Express — Framework web per il backend
- Mongoose — Per collegare la logica del backend a mongoDB
- OpenLayer — Per la mappa e le funzionalità annesse
- Next — Framework web per il frontend
- crypto — Per l'hashing delle password (implementato parzialmente)
- Joyride — Per il tutorial iniziale sull'utilizzo della piattaforma
- cors — Per le richieste di tipo CORS

E le seguenti dipendenze di sviluppo:

- Postman — Per il debug e il testing delle API
- MongoDB Compass — Per i primi test e configurazione di mongoDB
- mongoimport — Per la creazione e l'utilizzo di diversi data sample

4.4 Database

Per la gestione dei dati utili all'applicazione abbiamo definito 5 principali strutture dati:

- categorie: categorie nelle quali rientrano i tag
- crowdingraw: dati raw relativi alla frequentazione dei parchi
- crowdingdata: dati lavorati relativi alla frequentazione dei parchi
- parchi: dati relativi ai parchi presenti nell'applicazione
- segnalazioni: dati relativi a tutte le segnalazioni
- tags: tutti i tag presenti nell'applicazione

4.5 Testing

Tutti i casi di test specificati sono stati testati con postman, gli screenshot dei test sono presenti presso github.com/AleFacciUnitn/OasiTN/tree/main/Docs/Screenshot_testing

Nota: Tutti i casi il cui risultato atteso é server error sono stati esclusi dagli screenshot e motivati dall'utilizzo dell'Id nell'endpoint, di conseguenza in assenza dello stesso non viene riconosciuto l'endpoint come valido

N. test case	Descrizione test case	test data	Precondizioni	Dipendenze	Risultato Atteso	Risultato Ottenuto	Note
1.1	Creazione di una segnalazione corretta	<Parco> valido <oggetto> valido <descrizione > Valida <prioritá> valida	—	Parco presente nel database	Creazione della segnalazione e messaggio di <Created>	Creazione della segnalazione e messaggio di <Created>	—
1.2	Creazione di una segnalazione senza sepecificare il parco,	<Parco> vuoto <oggetto> valido <descrizione > Valida <prioritá> valida	—	—	Server Error	Server Error	—
1.3	Creazione di una segnalazione specificando un parco non esistente	<Parco> non esistente <oggetto> valido <descrizione > Valida <prioritá> valida	—	Parco assente nel database	Nessuna Creazione e messaggio di Errore <Dati non validi>	Nessuna Creazione e messaggio di Errore <Dati non validi>	—
2.1	Modifica di un parco corretta	<Parco> valido <tags> validi <infoparco> valido <password> corretta	—	Parco presente nel database	Modifica del parco esistente e messaggio <OK> di conferma	Modifica del parco esistente e messaggio <OK> di conferma	—
2.2	Modifica di un parco non esistente	<Parco> non valido <tags> validi <infoparco> valido <password> corretta	—	Parco assente nel database	Nessuna modifica e messaggio di errore <Errore durante l'aggiornamento del parco>	Nessuna modifica e messaggio di errore <Errore durante l'aggiornament o del parco>	—
2.3	Modifica di un parco inserendo la password errata	<Parco> valido <tags> validi <infoparco> valido <password>	—	—	Nessuna modifica e messaggio di errore <Password errata>	Nessuna modifica e messaggio di errore <Password errata>	—

		errata					
2.4	Modifica di un parco senza inserire l'id del parco	<Parco> vuoto <tags> validi <infoparco> valido <password> corretta	—	—	Server Error	Server Error	—
3.1	Aggiunta parco corretta	<Nome> Valido <Location> Valida <Tags> Validi <Password> Corretta	—	Tags esistenti nel database	Aggiunta del parco al database e messaggio <Created>	Aggiunta del parco al database e messaggio <Created>	—
3.2	Aggiunta di un parco preesistente	—	—	—	—	—	questo caso non è stato definito
3.3	Aggiunta di un parco sbagliando password	<Nome> Valido <Location> Valida <Tags> Validi <Password> Errata	—	—	Errore nell'aggiunta del parco e messaggio <Password Errata>	Errore nell'aggiunta del parco e messaggio <Password Errata>	—
3.4	Aggiunta di un parco inserendo tag non validi	<Nome> Valido <Location> Valida <Tags> non validi <Password> Valida	—	I tag inseriti non sono presenti nel database	Errore nell'aggiunta del parco e messaggio <Dati tag non validi>	Errore nell'aggiunta del parco e messaggio <Dati tag non validi>	—
4.1	Eliminazione corretta di un parco	<ID_parco> valido <Password> corretta	—	Parco esistente nel database	Eliminazione corretta del parco e messaggio <OK> di conferma	Eliminazione corretta del parco e messaggio <OK> di conferma	—
4.2	Eliminazione di un parco inserendo la password errata	<ID_parco> valido <Password> errata	—	—	Nessuna eliminazione e messaggio di errore <Password non valida>	Nessuna eliminazione e messaggio di errore <Password non valida>	—
4.3	Eliminazione di un parco inserendo un parco non esistente	<ID_parco> non valido <Password> corretta	—	Parco assente nel database	Nessuna eliminazione e messaggio di errore <Parco non trovato>	Nessuna eliminazione e messaggio di errore <Parco non trovato>	—
4.4	Eliminazione di un	<ID_parco>	—	—	Server Error	Server Error	

	parco senza inserire alcun parco	vuoto <Password> corretta					
5.1	Modifica stato di una segnalazione corretta	<ID_segnalazione> valido <stato> valido <Password> corretta	—	Segnalazione esistente nel database	Modifica dello stato della segnalazione e messaggio <OK> di conferma	Modifica dello stato della segnalazione e messaggio <OK> di conferma	—
5.2	Modifica stato di una segnalazione non esistente	<ID_segnalazione> non valido <stato> valido <Password> corretta	—	Segnalazionassente nel database	Nessuna modifica dello stato della segnalazione e mesaggio di errore <segnalazione non trovata>	Nessuna modifica dello stato della segnalazione e mesaggio di errore <segnalazione non trovata>	—
5.3	Modifica stato di una segnalazione senza specificare l'id	<ID_segnalazione> vuoto <stato> valido <Password> corretta	—	—	Nessuna modifica dello stato della segnalazione e mesaggio di errore <Dati non validi>	Nessuna modifica dello stato della segnalazione e mesaggio di errore <Dati non validi>	—
5.4	Modifica stato di una segnalazione con una password errata	<ID_segnalazione> valido <stato> valido <Password> errata	—	—	Nessuna modifica dello stato della segnalazione e mesaggio di errore <Password non valida>	Nessuna modifica dello stato della segnalazione e mesaggio di errore <Password non valida>	—
6.1	Eliminazione corretta di una segnalazione	<ID_segnalazione> valido <password> corretta	—	Segnalazione esistente nel database	Completamento della segnalazione e messaggio <Segnalazione completata> di OK	Completamento della segnalazione e messaggio <Segnalazione completata> di OK	—
6.2	Eliminazione di una segnalazione non esistente	<ID_segnalazione> non valida <password> corretta	—	Segnalazione non esistente nel database	Nessuna eliminazione e messaggio di errore <Segnalazione non trovata>	Nessuna eliminazione e messaggio di errore <Segnalazione non trovata>	—
6.3	Eliminazione di una segnalazione senza specificare l'ID della stessa	<ID_segnalazione> vuoto <password> corretta	—	—	Server Error	Server Error	—
6.4	Eliminazione della segnalazione usando una password errata	<ID_segnalazione> valido <password> errata	—	—	Nessuna eliminazione e messaggio di errore <Password non valida>	Nessuna eliminazione e messaggio di errore <Password non valida>	—

7.1	Modifica di una categoria corretta	<ID_categoria> valido <Descrizione> valida <password corretta>	—	Categoria Esistente nel database	Modifica eseguita e messaggio di OK	Modifica eseguita e messaggio di OK	—
7.2	Modifica di una categoria non esistente	<ID_categoria> non valida <Descrizione> valida <password corretta>	—	Categoria assente nel database	Nessuna modifica e Messaggio di errore <Categoria non trovata>	Nessuna modifica e Messaggio di errore <Categoria non trovata>	—
7.3	Modifica di una categoria senza specificare la stessa	<ID_categoria> vuoto <Descrizione> valida <password> corretta	—	—	Server Error	Server Error	—
7.4	Modifica di una categoria con password errata	<ID_categoria> valido <Descrizione> valida <password> errata	—	—	Nessuna modifica e Messaggio di errore <Password Errata>	Nessuna modifica e Messaggio di errore <Password Errata>	—
8.1	Modifica Tag Corretta	<ID_Tag> valido <Nome> valido <Nome_Categoria> valido <Password> Corretta	—	Tag e categoria presenti nel database	Modifica del tag e messaggio di OK della modifica	Modifica del tag e messaggio di OK della modifica	—
8.2	Modifica Tag usando un Tag non esistente	<ID_Tag> non valido <Nome> valido <Nome_Categoria> valido <Password> Corretta	—	Tag assente nel database	Nessuna modifica e messaggio di errore <Errore durante l'aggiornamento del tag>	Nessuna modifica e messaggio di errore <Errore durante l'aggiornamento del tag>	—
8.3	Modifica Tag usando una categoria non esistente	<ID_Tag> valido <Nome> valido <Nome_Categoria> non valido <Password> Corretta	—	Categoria non esistente	Nessuna modifica e messaggio di errore <Categoria non valida>	Nessuna modifica e messaggio di errore <Categoria non valida>	—
8.4	Modifica Tag senza specificare l'ID dello stesso	<ID_Tag> vuoto <Nome>	—	—	Server Error	Server Error	—

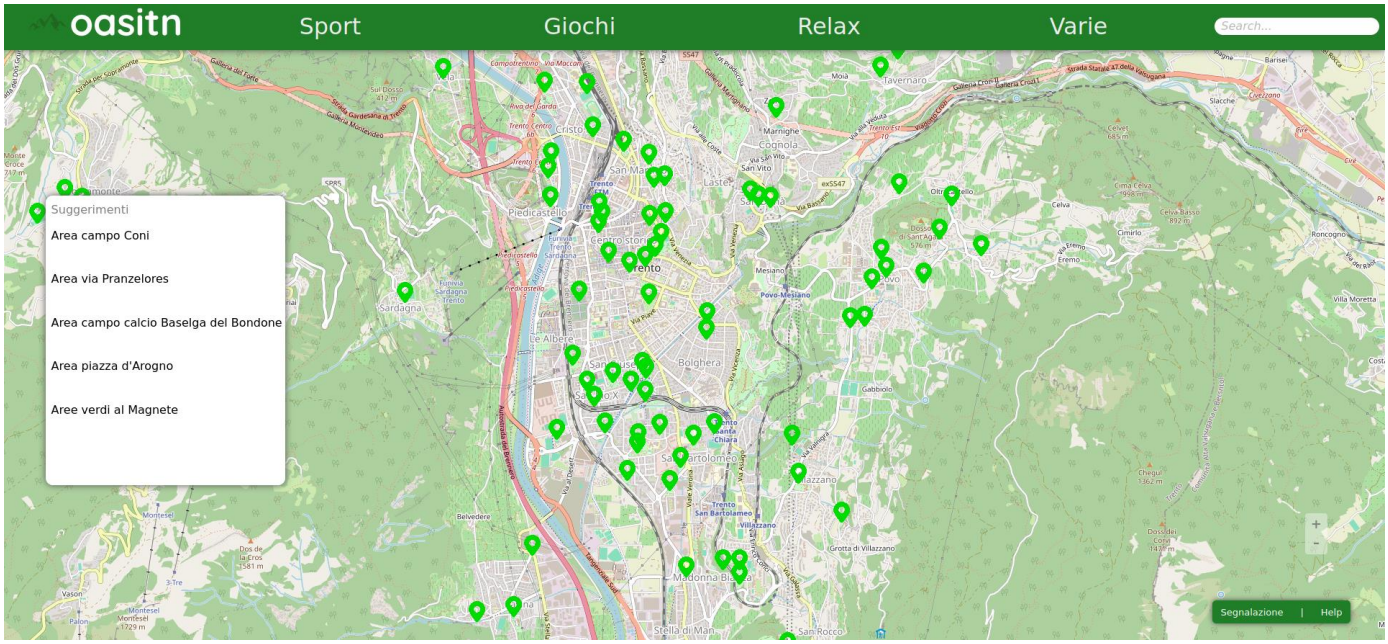
		valido <Nome_Categoria> valido <Password> Corretta					
8.5	Modifica Tag con password errata	<ID_Tag> valida <Nome> valido <Nome_Categoria> valido <Password> Errata	—	—	Nessuna modifica e messaggio di errore <Password non valida>	Nessuna modifica e messaggio di errore <Password non valida>	—
9.1	Aggiunta categoria corretta	<Nome_Categoria> valido <Password> valida	—	—	Categoria aggiunta e messaggio <Categoria aggiunta con successo di conferma>	Categoria aggiunta e messaggio <Categoria aggiunta con successo di conferma>	—
9.2	Aggiunta categoria preesistente	<Nome_Categoria> valido <Password> valida	—	Categoria esistente nel database	Nessuna aggiunta e messaggio di errore <Categoria giá esistente>	Nessuna aggiunta e messaggio di errore <Categoria giá esistente>	—
9.3	Aggiunta categoria con password non valida	<Nome_Categoria> valido <password> non valida	—	—	Nessuna aggiunta e messaggio di errore <Password non valida>	Nessuna aggiunta e messaggio di errore <Password non valida>	—
10.1	Aggiunta tag corretta	<Nome> valido <Descrizione> valida <Nome_categoria> valida <Password> corretta	—	Categoria presente nel database e nome del tag non presente nel database	Aggiunta tag e messaggio <Tag aggiunto con successo> di ok	Aggiunta tag e messaggio <Tag aggiunto con successo> di ok	—
10.2	Aggiunta Tag con categoria non valida	<Nome> valido <Descrizione> valida <Nome_categoria> non valida <Password> corretta	—	Categoria non presente nel database	Nessuna aggiunta e messaggio di errore <categoria non trovata>	Nessuna aggiunta e messaggio di errore <categoria non trovata>	—
10.3	Aggiunta tag preesistente	<Nome> valido <Descrizione> valida <Nome_categoria> valida	—	Nome tag esistente nel database	Nessuna aggiunta e messaggio di errore <Errore nel server>	Nessuna aggiunta e messaggio di errore <Errore nel server>	—

		<Password> corretta					
10.4	Aggiunta tag con password non valida	<Nome> valido <Descrizione> valida <Nome_categoria> valida <Password> Errata	—	—	Nessuna aggiunta e messaggio di errore <Password non valida>	Nessuna aggiunta e messaggio di errore <Password non valida>	—

5. Frontend

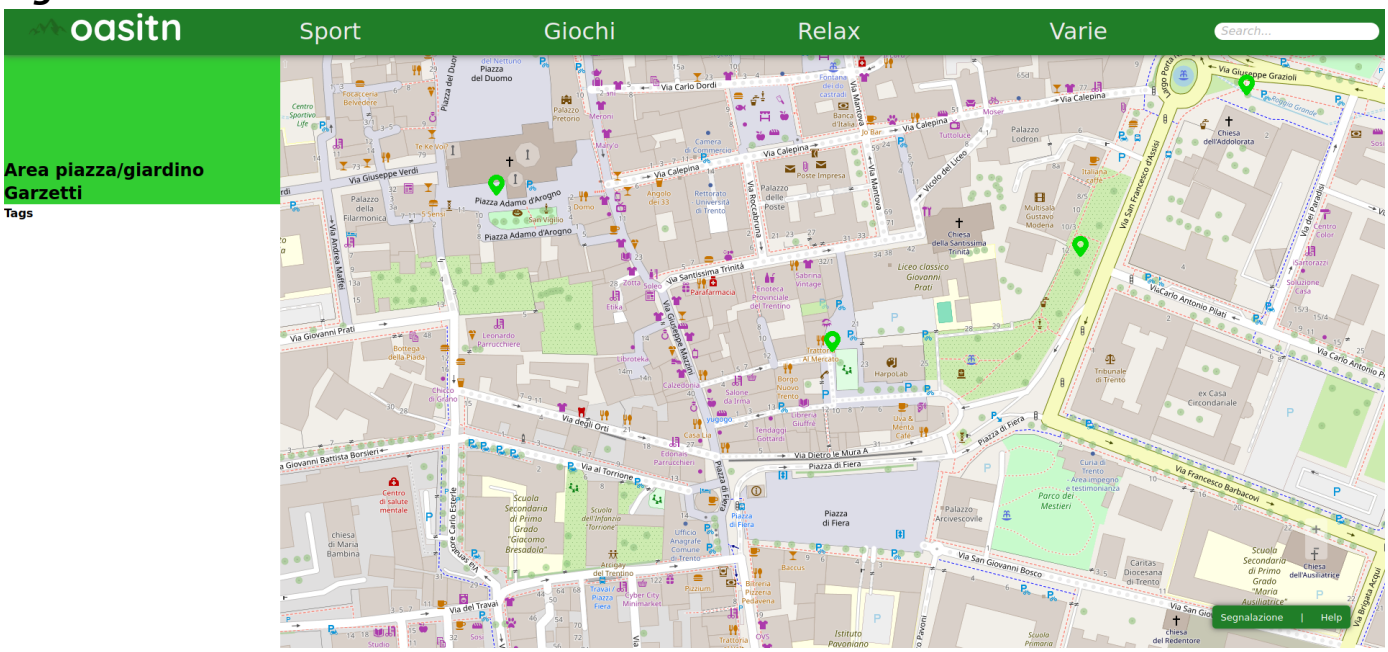
Il frontend fornisce funzionalitá di navigazione dei parchi inserimento modifica e cancellazione degli stessi e la gestione delle segnalazioni in particolare l'applicazione é composta da una homepage, una pagina di visualizzazione dei parchi, una pagina per l'aggiunta di segnalazione e da una pagina destinata all'admin per la gestione dei parchi e delle segnalazioni.

Figura 1



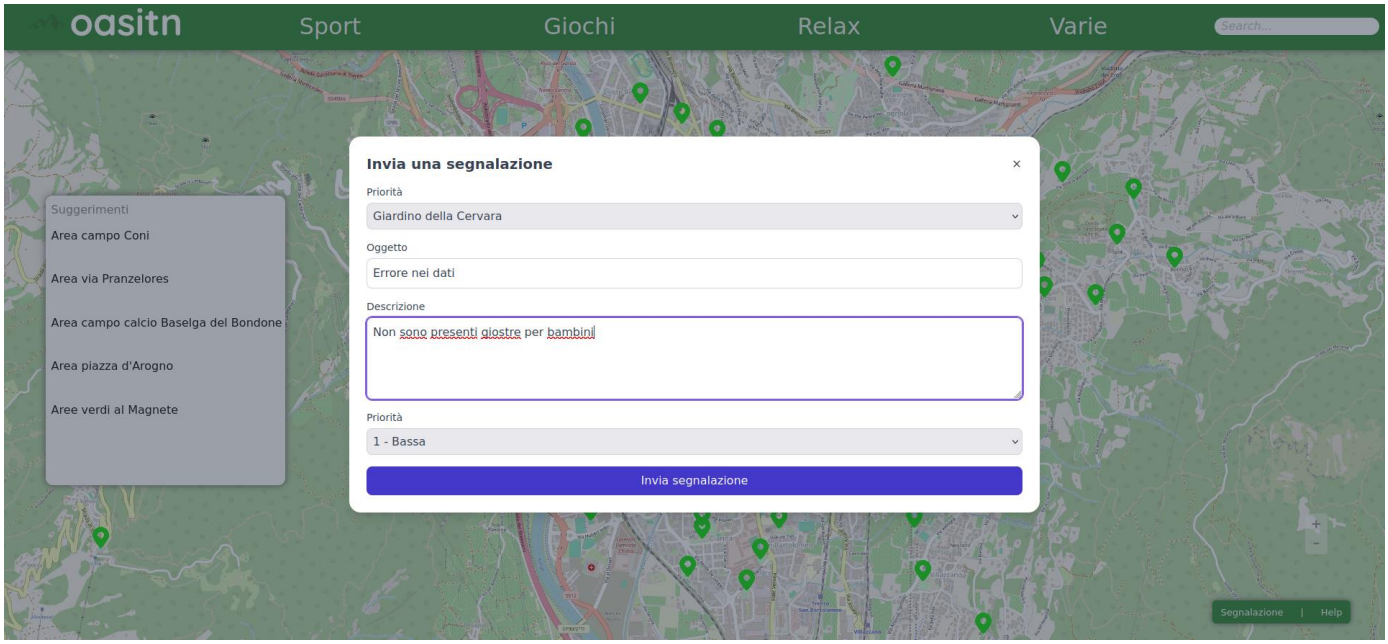
La figura 1 mostra la homepage del sito presentata all'utente, la quale permette attraverso vari metodi di navigare i parchi presenti.

Figura2



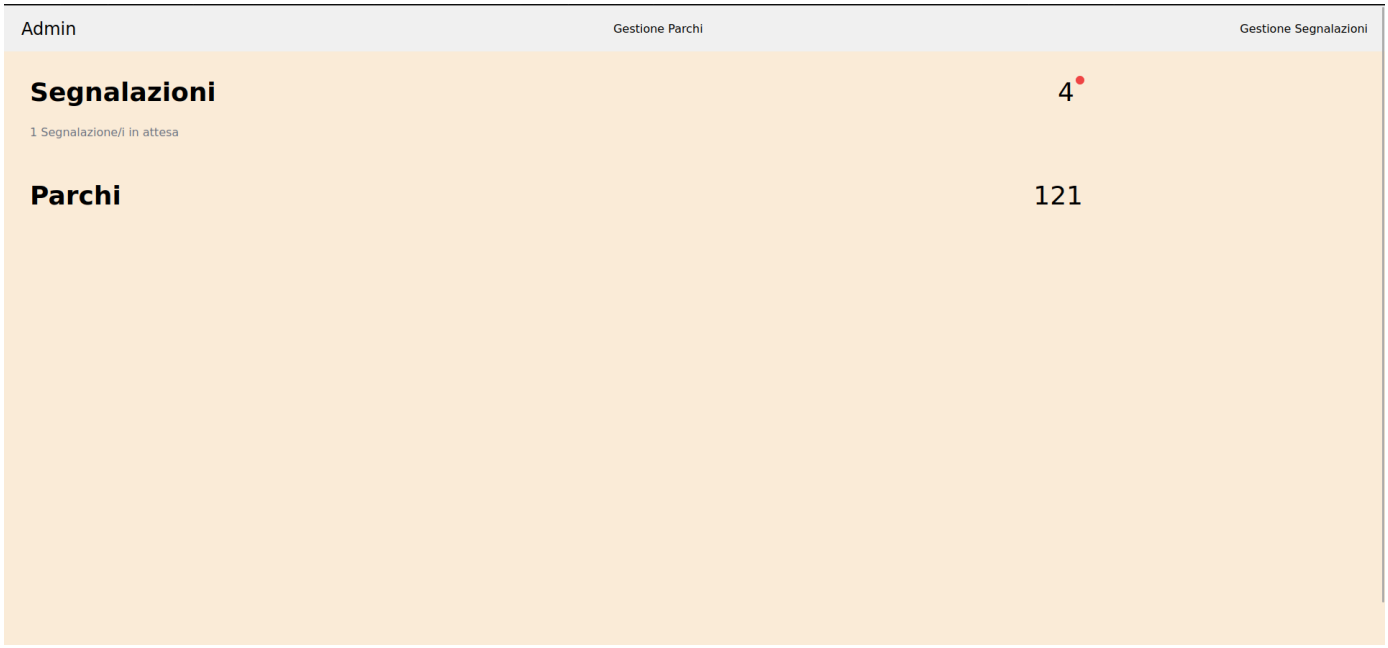
La figura 2 mostra la schermata di visualizzazione del parco dell'utente.

Figura 3



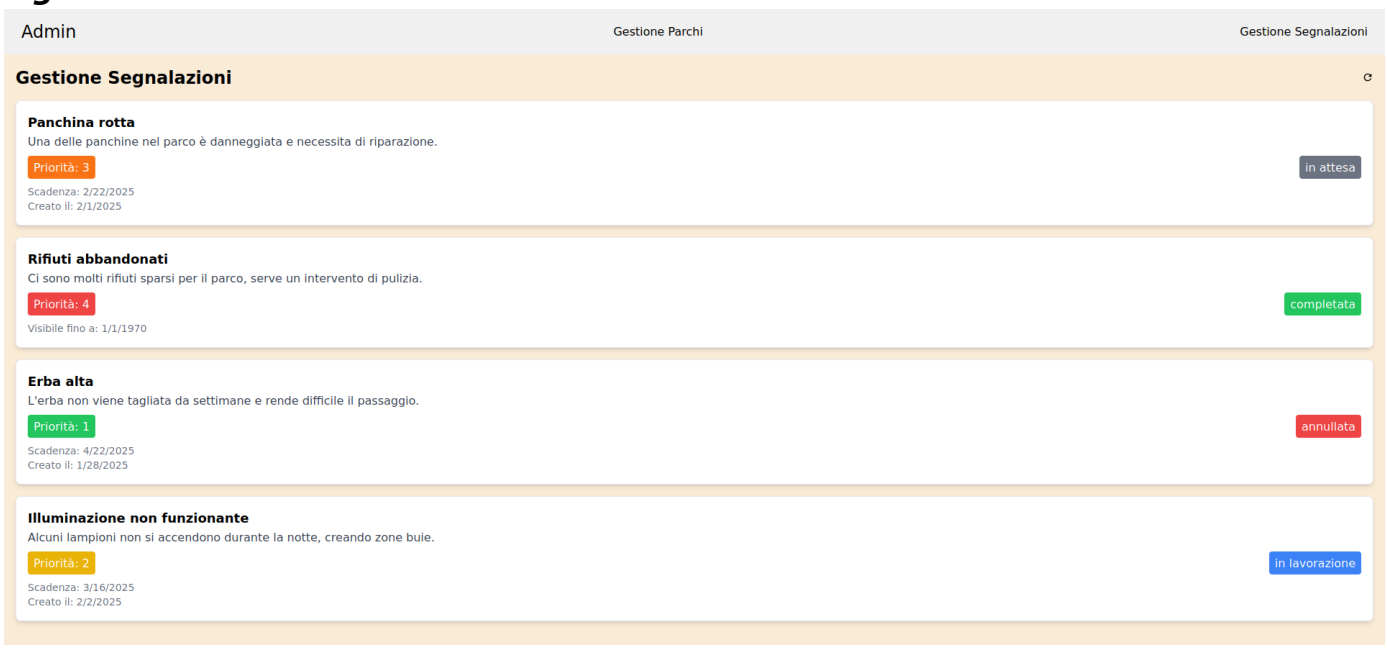
La figura 3 mostra la schermata di inserimento delle segnalazioni dell'utente.

Figura 4



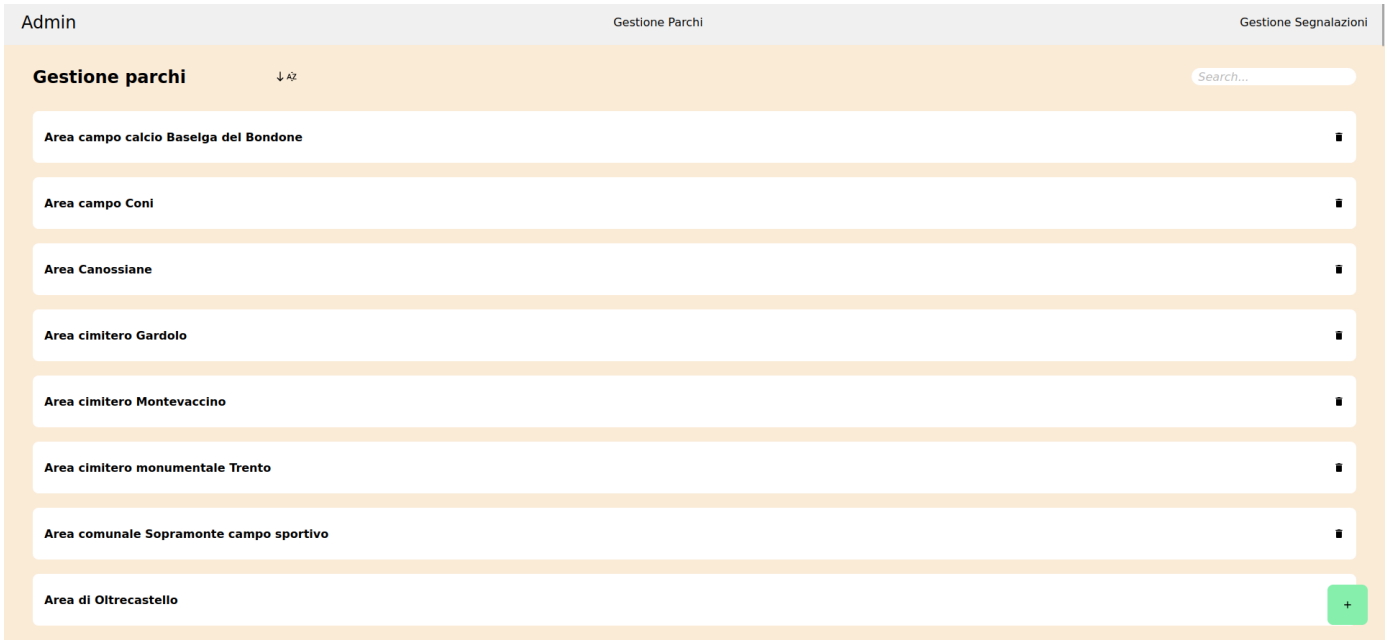
La figura 4 mostra la home page dell'admin che permette di accedere al controllo dei parchi o delle segnalazioni

Figura 5



La figura 5 mostra la schermata di gestione delle segnalazioni da parte dell'admin.

Figura 6



La figura 6 mostra la schermata di gestione e modifica dei parchi.

6. Deployment

Un istanze del frontend è ospitata pubblicamente su questo sito oasitn.onrender.com,
un instanza del backend è ospitata pubblicamente su oasitn-db.onrender.com

Per accedere alla pagina Admin è sufficiente aggiungere /admin all'URL e la password per accedere è 123456789.

Per problemi con il deploy contattare a.faccincani-1@studenti.unitn.it