



UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in
Informatica

PROGETTO INGEGNERIA DEL SOFTWARE

SISTEMA DI MONITORAGGIO AMBIENTALE

D5 G19

Anno accademico 2021/2022

Indice

Scopo del documento	3
1 User Stories	4
2 Features	5
3 User Flows	6
4 Application Implementation and Documentation	8
4.1 Project Structure	8
4.2 Project Dependencies	8
4.3 Project Data or DB	9
4.4 Project APIs	11
4.4.1 Elenco informazioni di un animale	11
4.4.2 Aggiornamento rischio ambientale in tempo reale	11
4.4.3 Aggiunta di un sensore	12
4.4.4 Rimozione di un sensore	12
5 API Documentation	13
6 Front End Implementation	14
7 GitHub Repository Info	19
8 Testing	20

Scopo del documento

Il documento presentato riporta tutte le informazioni necessarie per lo sviluppo dell'applicativo Sistema Monitoraggio Ambientale. In particolare, esso presenta tutti gli artefatti necessari per sviluppare le funzionalità dell'account di tipo Amministratore, definito nei precedenti documenti. Partendo dalla descrizione delle user stories, features ed user flow legate al ruolo di amministratore, il documento procede con la presentazione delle API necessarie per poter visualizzare, modificare ed inserire i dati dei sensori GPS utilizzati dall'applicazione. A seguire troveremo un capitolo dove sarà possibile vedere come è stata pensata l'interfaccia utente, con l'obiettivo di essere la più semplice e chiara possibile. Per ogni API presentata, oltre alla descrizione delle funzionalità fornite, il documento presenta la documentazione ed i casi di test correlati.

1 | User Stories

Una "User Story" è una descrizione informale e generica di una caratteristica del software presentata dal punto di vista dell'utente finale.

Il punto focale di ogni User Story presentata nel documento viene posto sull'amministratore, ossia l'utente principale da noi selezionato per l'implementazione dell'applicazione. Esso è in grado sia di visualizzare i dati relativi al parco in termini di popolazione, dei rischi ambientali e delle descrizioni dei singoli animali, sia di gestire i sensori GPS dei singoli animali.

Ogni User Story viene definita con un linguaggio non tecnico, volta a fornire allo sviluppatore un'idea del chi (Who?), cosa (What?) e del perché(Why?) si sta sviluppando una specifica funzionalità del software. Le **User Stories** legate all'Amministratore dell'applicazione "Sistema di monitoraggio ambientale" sono riportate nella tabella seguente. Nel caso non sia presente un requisito funzionale o non funzionale, viene presentata una "X" ad indicare che non esiste tale correlazione.

	Descrizione User Story	Requisiti funzionali	Requisiti non funzionali
US1	Come amministratore voglio poter selezionare il parco in cui mi trovo, per vedere le informazioni di quest'ultimo	RF 2.1	X
US2	Come amministratore voglio poter visualizzare le specie di flora e fauna, per vedere le informazioni degli animali e delle piante	RF 2.1	X
US3	Come amministratore voglio poter visualizzare le statistiche del parco in termini di pericolo ambientale così da poterlo tenere sotto controllo	RF 2.3	RNF 3.3
US4	Come amministratore voglio poter visualizzare, per la specie animale selezionata, un istogramma rappresentante la sua popolazione nel tempo e una mappa delle posizioni degli animali di quella specie in tempo reale, in modo da tenerli sotto controllo	X	RNF 2.5, RNF 3.3
US5	Come amministratore, voglio essere in grado di selezionare un sensore GPS del parco per poi eliminarlo, in modo da poterlo gestire	RF 2.5	X
US6	Come amministratore, voglio essere in grado di aggiungere un sensore GPS nel parco, in modo da poterlo gestire	RF 2.5	X
US7	Come amministratore, voglio essere in grado di selezionare un sensore GPS del parco per poi visualizzarne i dati, in modo da poter ottenere informazioni sul singolo elemento	RF 2.5	X

2 | Features

User Story	Feature index	Feature
US1	F1	Mostrare i parchi selezionabili utilizzando una lista.
US1	F2	Estrarre tutti i parchi memorizzati tramite un'API locale.
US1	F3	Mostrare la pagina del parco selezionato.
US2	F4	Mostrare le specie presenti nel parco selezionato utilizzando un elenco.
US2	F5	Estrarre tutte le specie presenti nel parco tramite un'API locale.
US3	F6	Mostra pagina contenente le statistiche ambientali.
US3	F7	Estrarre le statistiche ambientali memorizzate tramite un'API locale.
US4	F8	Mostrare le specie presenti nel parco selezionato utilizzando un elenco.
US4	F9	Estrarre tutte le specie presenti nel parco tramite un'API locale.
US4	F10	Mostrare pagina con la mappa contenente le posizioni degli animali e l'istogramma sullo storico della popolazione.
US4	F11	Estrarre tutte le posizioni dell'animale selezionato tramite un'API locale.
US4	F12	Estrarre lo storico dell'animale selezionato tramite un'API locale.
US5	F13	Mostrare pagina con i dati dei sensori GPS tramite un elenco.
US5	F14	Estrarre i dati dei sensori tramite un'API locale filtrati con il parco selezionato dall'amministratore.
US5	F15	Mostrare un'area contenente i dati del sensore selezionato.
US5	F16	Predisporre un bottone per eliminare il sensore.
US5	F17	Aggiornare il database tramite un'API locale.
US6	F18	Mostrare una pagina con i dati dei sensori tramite un elenco.
US6	F19	Estrarre i dati dei sensori tramite un'API locale, filtrati con il parco selezionato dall'amministratore.
US6	F20	Predisporre un bottone per aggiungere un sensore.
US6	F21	Visualizzare un form con dei campi da compilare per aggiungere un sensore.
US6	F22	Memorizzare sul database i dati del nuovo sensore tramite un'API locale.
US7	F23	Mostrare una pagina contenente i dati dei sensori tramite un elenco.
US7	F24	Estrarre i dati dei sensori tramite un'API locale filtrati con il parco selezionato dall'amministratore.
US7	F25	Visualizzare un'area contenente i dati del singolo sensore.

3 | User Flows

In questa sezione vengono presentati gli User Flows correlati alla figura dell'Amministratore da noi scelta. La figura 3.1 descrive lo User Flow relativo alla gestione ed ottenimento delle informazioni del parco, già introdotte nel capitolo introduttivo. In qualsiasi momento l'amministratore può gestire e visualizzare i sensori presenti nel parco e visualizzare informazioni inerenti la flora e la fauna del suddetto.

Nella figura 3.1 vengono anche presentate le relazioni tra le azioni effettuate da parte dell'amministratore e le features descritte nel capitolo precedente. In aggiunta viene posta una legenda per facilitare la comprensione del diagramma sottostante.

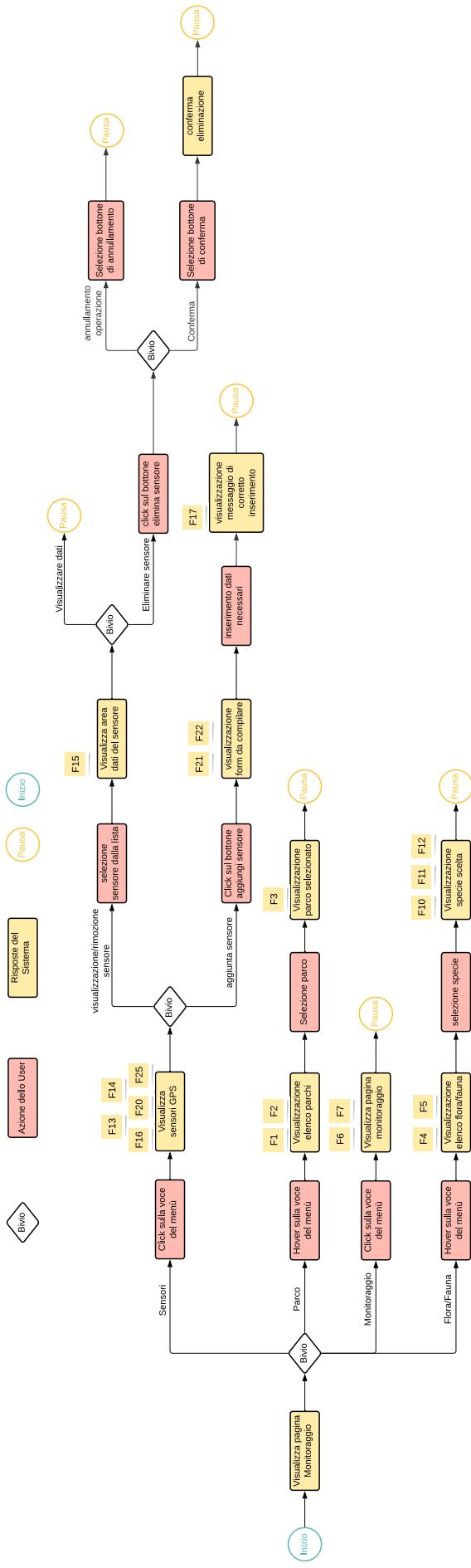


Figura 3.1: User flow

4 | Application Implementation and Documentation

Nei capitoli precedenti sono state presentate le features che devono essere implementate per il corretto funzionamento dell'applicazione Sistema Monitoraggio Ambientale nel caso in cui dovesse essere utilizzata da un utente di tipo Amministratore. L'applicazione è stata sviluppata con **NodeJS** e **Bootstrap**. Per la gestione dei dati abbiamo utilizzato **MongoDB**.

4.1 Project Structure

Nella figura 4.1 possiamo vedere la struttura del progetto. Esso è composto da una cartella api per la gestione delle API locali, una cartella css che contiene un file di stile, una cartella images che contiene tutte le immagini utilizzate nel progetto, una cartella js che raggruppa tutti i file javascript e infine una cartella ui che contiene le le pagine html. In fondo si può vedere anche la pagina html principale dell'applicazione e il file README.md utilizzato da github per fornire una descrizione del progetto contenuto nella repository e come eseguirlo correttamente.

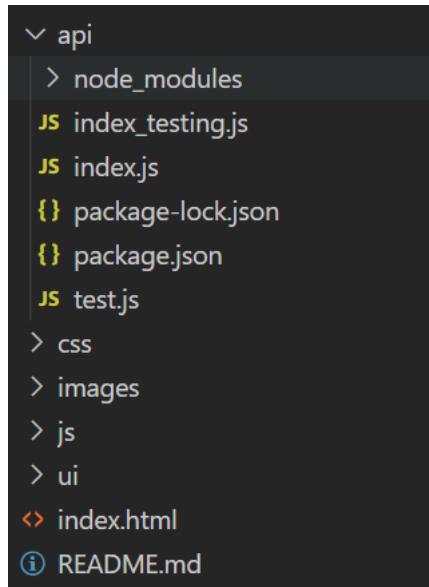


Figura 4.1: Struttura del progetto

4.2 Project Dependencies

I seguenti moduli Node sono stati utilizzati ed aggiunti al file `package.json`:

- Express
- Body-Parser
- Cors
- MongoDB
- Swagger-jsDoc
- Swagger-ui-Express
- sweetalert2

Inoltre sono stati utilizzati dei moduli per sviluppo:

- supertest
- tap-spec
- tap

In aggiunta alle *dependencies* presentate, sono stati utilizzati anche Bootstrap, jQuery, MapBox e Google Charts come script HTML.

4.3 Project Data or DB

Per la gestione dei dati necessari al corretto funzionamento dell'applicativo abbiamo definito 6 strutture dati come illustrato in Figura 4.2. Le suddette sono "Fauna", "Flora", "Parco", "SensoreGPS", "StoricoFauna", "RischioAmbientale".

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Fauna	9	843.4 B	7.6 KB	1	36.9 KB	
Flora	7	931.3 B	6.5 KB	1	36.9 KB	
Parco	3	465.3 B	1.4 KB	1	36.9 KB	
SensoreGPS	56	135.6 B	7.6 KB	1	36.9 KB	
StoricoFauna	40	119.2 B	4.8 KB	1	36.9 KB	
rischioAmbientale	3	119.0 B	357.0 B	1	36.9 KB	

Figura 4.2: Collezione dati utilizzati dall'applicazione

Per rappresentare le seguenti strutture dati abbiamo definito i seguenti tipi di dati (Vedi figure da 4.3 a 4.8).

```
_id: ObjectId("61a2ae7bb48bb237244bf8a9")
Tipo: "Orso"
Descrizione: "Gli ursidi sono una famiglia
> Immagine: Array
> Parco: Array
Contentimento: true
```

Figura 4.3: Tipo di dato "Fauna"

```
_id: ObjectId("61a2af7db48bb237244bf8c2")
Posizione: "45° 28' 23 N 7° 20' 35 E"
TipoAnimale: "Orso"
Parco: "Gran Paradiso"
SenId: 1
```

Figura 4.6: Tipo di dato "SensoreGPS"

```
_id: ObjectId("61aa563e528b2ed0b34f8843")
Tipo: "Abete Bianco"
Descrizione: "L'abete bianco, una delle con
> Immagine: Array
> Parco: Array
```

Figura 4.4: Tipo di dato "Flora"

```
_id: ObjectId("61a2b0eab48bb237244bf8e6")
Parco: "Gran Paradiso"
TipoAnimale: "Orso"
NumEsemplari: 4
Giorno: 1
Anno: 2021
Mese: 11
```

Figura 4.7: Tipo di dato "StoricoFauna"

```
_id: ObjectId("619ffb40588047e3616941db")
Parco: "Gran Paradiso"
Posizione: "45° 32' 00 N 7° 17' 00 E"
Coordinate: "https://www.google.com/maps/ma
> Immagine: Array
```

Figura 4.5: Tipo di dato "Parco"

```
_id: ObjectId("61aa33b07a2b2ecdc0a733a")
allagamento: 42
incendio: 11
meteo: 13
parco: "Gran Paradiso"
risorseIdriche: 33
siccita: 7
```

Figura 4.8: Tipo di dato "rischioAmbientale"

4.4 Project APIs

4.4.1 Elenco informazioni di un animale

Utilizzando questa API l'applicativo può ottenere le informazioni inerenti ad un singolo animale contenuto al momento della chiamata nel sistema. L'API mediante il metodo `find({})` ritorna il singolo animale inserito come parametro nella funzione. In caso venga rilevato un errore, esso verrà stampato a console.

```
app.get('/api/fauna/:animale', (request, response) => {
  database.collection("Fauna").find({
    Tipo: request.params.animale
  }).toArray((error, result) => {
    if (error) {
      console.log(error);
    }

    response.send(result);
  })
})
```

4.4.2 Aggiornamento rischio ambientale in tempo reale

La seguente API permette all'applicazione di simulare la ricezione ed inserimento nel database dei dati elaborati dal "Sistema Di Calcolo Probabilistico" (nominato molteplici volte nei documenti precedenti).

Con il metodo `updateOne({}, {})`, il sistema è in grado di modificare i valori di rischio di un parco in modo tale che le variazioni non siamo troppo elevate.

```
app.put('/api/rischioAmbientale', (request, response) => {

  database.collection("rischioAmbientale").updateOne(
    // Filter Criteria
    {
      "parco": request.body['parco']
    },
    // Update
    {
      $set:
      {
        allagamento: request.body['allagamento'],
        incendio: request.body['incendio'],
        meteo: request.body['meteo'],
        siccita: request.body['siccita'],
        risorseIdriche: request.body['risorseIdriche'],
      }
    }
  );
})
```

4.4.3 Aggiunta di un sensore

Questa API viene utilizzata ognqualvolta si desidera inserire un nuovo sensore nel sistema.

Permette di creare un nuovo oggetto di tipo `SensoreGPS` composto da `Posizione`, `TipoAnimale`, `Parco` e `SenId` inviati come body della richiesta. Nel caso di avvenuto inserimento, viene visualizzato a console un messaggio di conferma, in caso contrario viene stampato un messaggio di errore.

```
app.post('/api/sensoreGPS', (request, response) => {

    database.collection("SensoreGPS").count({}, function (error) {
        if (error){
            console.log(error);
        }

        database.collection("SensoreGPS").insertOne({
            Posizione: request.body['posizione'],
            TipoAnimale: request.body['tipoAnimale'],
            Parco: request.body['parco'],
            Contenimento: request.body['contenimento'],
            SenId: request.body['senId']
        });

        response.json("Added Successfully");
    })
})
```

4.4.4 Rimozione di un sensore

La seguente API permette all'applicazione di rimuovere un sensore dal sistema. Essa prende come input l'ID del sensore da eliminare e mediante il metodo `deleteOne({})` elimina il sensore desiderato. In caso di corretta rimozione, viene visualizzato a console un messaggio di conferma.

```
app.delete('/api/sensoreGPS/:id', (request, response) => {
    let senId = new ObjectId(request.params.id);
    database.collection("SensoreGPS").deleteOne({
        _id: senId
    });

    response.json("Delete successfully!");
})
```

5 | API Documentation

Le API Locali fornite dall'applicazione Sistema Monitoraggio Ambientale e precedentemente descritte sono state documentate mediante il modulo NodeJS chiamato **Swagger-UI-Express**. Abbiamo definito le API utilizzando **JSDoc**, uno strumento volto alla generazione della documentazione utilizzando i commenti nel codice sorgente di un'applicazione. Così facendo la documentazione inerente alle API è visibile a chiunque visualizzi il codice sorgente. Per generare l'endpoint, la cui funzione è quella di presentare le API, abbiamo usato **Swagger UI** che genera una pagina web dalle definizioni delle specifiche OpenAPI.

Di seguito mostriamo la pagina web contenente la documentazione che presenta le 15 API (di tipo GET, POST, PUT and DELETE) per la gestione dei dati nel nostro applicativo.

La GET viene utilizzata per ottenere e visualizzare i dati in una pagina HTML. La POST serve ad inserire un nuovo dato nel sistema, la PUT modifica un dato presente nel sistema e la DELETE ne cancella uno.

Dopo aver correttamente avviato le API mediante terminale, l'endpoint da invocare per poter visualizzare la seguente documentazione è: <http://localhost:49146/api-docs>

The screenshot shows the Swagger UI interface for the 'SistemaMonitoraggioAmbientale API'. At the top, there's a green header bar with the 'swagger' logo. Below it, the title 'SistemaMonitoraggioAmbientale API' is displayed, along with a '1.0.0' version indicator and a 'docs' link. Underneath, there's a brief description of the API and links to 'Group19 - Website' and 'Licensed Under MIT'. A 'Server' dropdown is set to 'http://localhost:49146/'. The main content area is titled 'default' and lists various API endpoints categorized by method and path. Most endpoints are listed in blue boxes, while one PUT endpoint for modifying park environmental statistics is highlighted in orange. Other methods shown include GET, POST, and DELETE. The endpoints include paths like /api/fauna, /api/fauna/{animale}, /api/fauna/cont/{contenimento}, /api/flora, /api/pianta, /api/parco, /api/nome, /api/rischioAmbientale, /api/rischioAmbientale, /api/rischioAmbientale/{parco}, /api/sensoreGPS, /api/sensoreGPS/{parco}, /api/sensoreGPS/{animale}/{parco}, /api/sensoreGPS, and /api/istoricoFauna/{animale}/{parco}.

Figura 5.1: Documentazione delle API di Sistema di Monitoraggio Ambientale

6 | Front End Implementation

Il FrontEnd fornisce le funzionalità di visualizzazione, inserimento e cancellazione di dati dall'applicazione Sistema Monitoraggio Ambientale. L'applicazione è composta da 5 pagine: Monitoraggio, Parco, Flora, Fauna, Sensori.

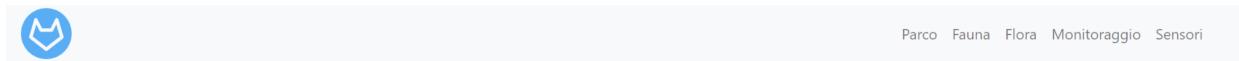
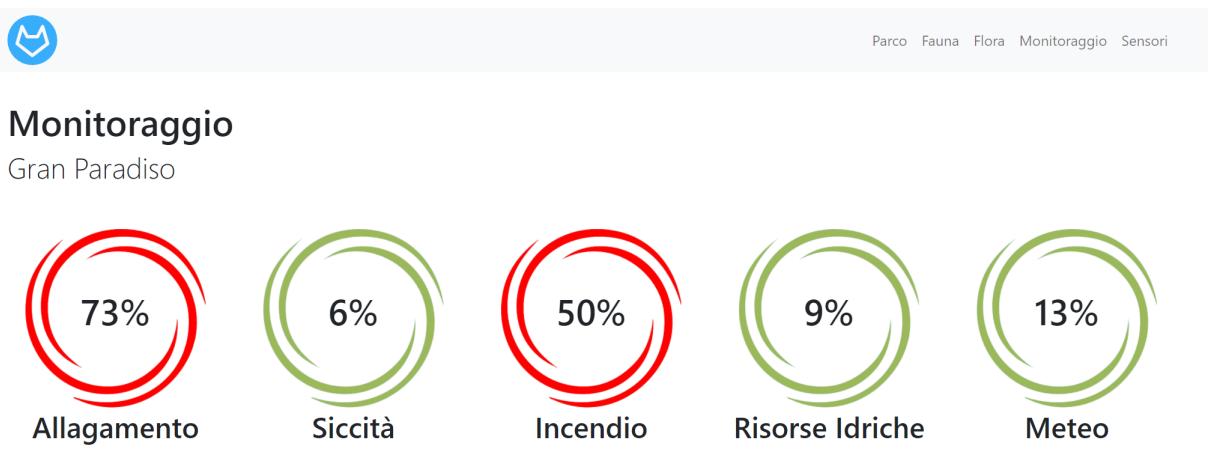


Figura 6.1: Menù dell'applicazione

Per utilizzare l'applicazione è presente un menù che guida la navigazione tra le pagine del progetto. Nelle prime tre voci, al momento della selezione si apre un menù a tendina che permette di selezionare il parco, l'animale o la vegetazione desiderata.



In questa pagina è possibile trovare informazioni riguardanti le percentuali di pericolo registrate dai sensori ambientali posti nel parco.

Figura 6.2: Pagina di Monitoraggio

Nella Figura 6.2 si possono visualizzare i livelli di pericolo delle categorie di monitoraggio del parco selezionato. In base al livello raggiunto il colore cambia tra verde, giallo e rosso.



Gran Paradiso

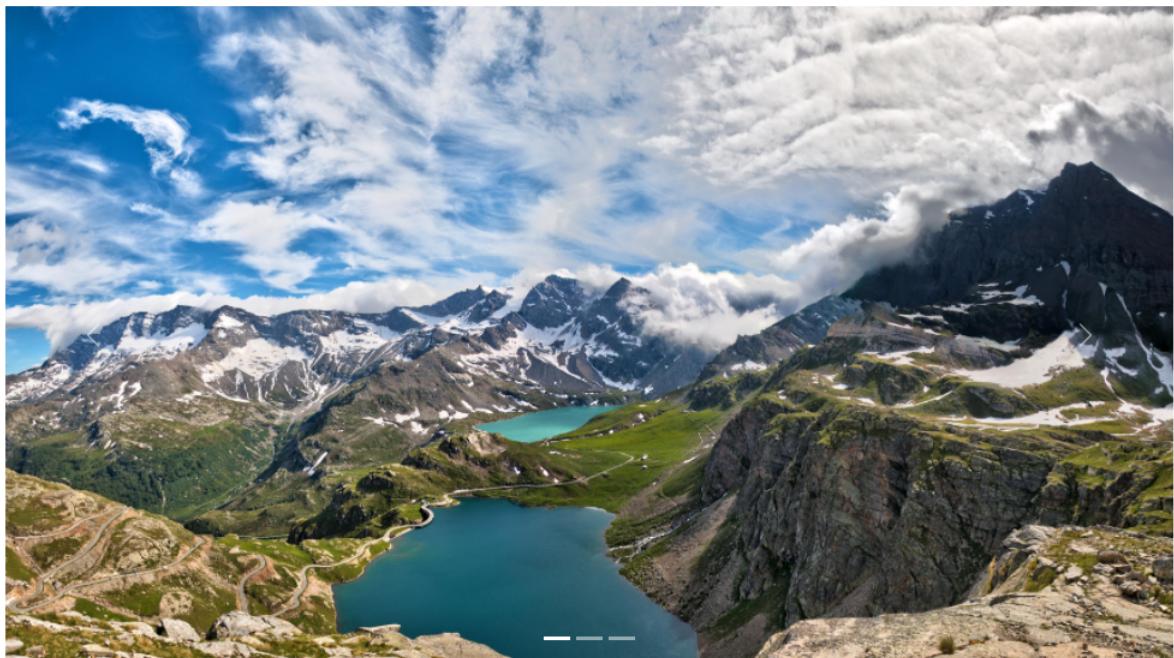


Figura 6.3: Esempio di Parco selezionato

Nella Figura 6.3 è possibile visualizzare delle immagini del parco ed una cartina geografica navigabile del suddetto.



Orso

Gran Paradiso

Gli ursidi sono una famiglia di mammiferi appartenenti al sottordine dei caniformi. Sebbene oggi siano rappresentati solo da otto specie, essi sono molto diffusi e sono presenti in un'ampia varietà di habitat di tutto l'emisfero boreale e, in parte, di quello australe. Si trovano infatti in Nordamerica, Sudamerica, Europa e Asia. Caratteristiche comuni proprie degli orsi moderni sono il grande corpo sorretto da zampe tozze, il lungo muso, le piccole orecchie rotonde, la pelliccia ispida, le zampe plantigrade dotate di cinque artigli non retrattili e la coda corta.



Figura 6.4: Descrizione dell'animale selezionato



Selezione l'anno iniziale e finale

Inizio
2016

Fine
2020

Conferma

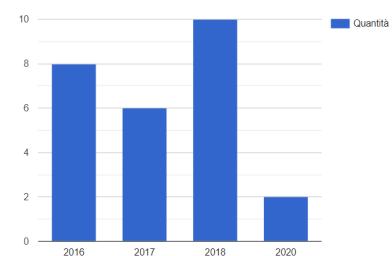
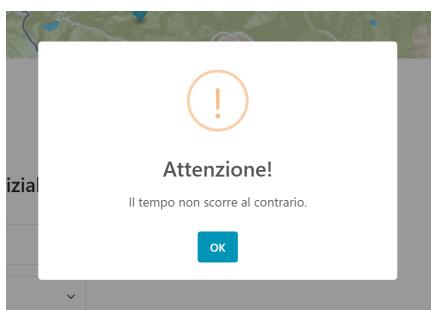


Figura 6.5: Posizione geografica e istogramma della popolazione

Nella figura 6.4 e 6.5 si possono visualizzare la descrizione e le immagini dell'animale selezionato. Inoltre è presente una mappa contenente dei marker che tracciano la posizione degli esemplari scelti. Infine si trova un form per inserire l'intervallo di tempo nel quale visualizzare i dati dello storico della popolazione del suddetto.



Nel caso in cui l'utente selezioni una data di inizio maggiore a quella di fine viene mostrato un pop-up di avvertimento 6.6.

Figura 6.6: Pop-up selezione anni errati



Abete Bianco

Gran Paradiso

L'abete bianco, una delle conifere indigene più frequenti, è una specie ombrivaga che appartiene alla famiglia delle Pinacee. Nel mondo le specie appartenenti al genere *Abies* sono una quarantina. Durante gli ultimi 15 anni in termini numerici, l'abete bianco è diminuito in Svizzera. Attualmente esso rappresenta solo il 10% degli alberi, cifra decisamente inferiore al 38% dell'abete rosso. Su terreni fertili nelle nostre regioni gli abeti bianchi possono raggiungere i 50-55 metri di altezza e vivere fino a 600 anni. Durante questo periodo essi sono in grado di produrre più legname rispetto all'abete rosso. L'abete bianco possiede una ramificazione molto regolare, con rami principali sempreverdi raggruppati in palchi regolari disposti lungo il fusto, lungo e rettilineo. I rametti secondari sono invece disposti lungo il fusto secondo un andamento a spirale. Durante la fase giovanile l'abete presenta una chioma piuttosto conica, mentre con l'aumentare dell'età essa assume un forma più appiattita, assomigliante ad un nido di cicogna. L'apparato radicale, relativamente poco ramificato, è di tipo fittonante e raggiunge la profondità di 1,60 metri. Tra le conifere è una delle specie che meglio si ancora al terreno ed è dunque meno soggetta a rotture o sradicamenti provocati dalle tempeste.



Figura 6.7: Pagina Flora dell'applicazione Sistema Monitoraggio Ambientale

Nella figura 6.7 si possono visualizzare descrizione ed immagini della specie vegetale selezionata.



Sensori

Gran Paradiso

Sensori ▾

Elimina sensore

Aggiungi sensore

Sensore

ID sensore parco: 4

Posizione: 45° 33' 35 N 7° 17' 59 E

Tipo Animale: Orso

Parco: Gran Paradiso

Contenimento: TRUE

Figura 6.8: Pagina dei Sensori

Nella figura 6.8 troviamo nella parte sinistra la possibilità di selezionare, eliminare ed aggiungere un sensore, mentre nella parte destra una tabella sulla quale visualizzare i dati del sensore selezionato. L'opzione "Elimina sensore" viene resa disponibile dopo aver selezionato un sensore e permette di eliminare quel preciso sensore dalla lista, notificando l'utente mediante un pop-up.

Figura 6.9: Form aggiungi sensore



Figura 6.10: Pop-up conferma

Nel caso in cui si volesse aggiungere un sensore (figura 6.9), l'amministratore deve inserire il tipo di animale, mentre i campi "Parco" e "ID sensore parco" sono precompilati dal programma, il quale assegna tali valori in base al parco e al numero di sensori presenti in esso nel momento dell'inserimento.

Come definito dallo User Flow al capitolo 3, il sistema prevede l'invio di conferma aggiunta sensore (figura 6.10) come anche un pop-up che richieda se l'utente è sicuro di voler eliminare un sensore (figura 6.11) e di seguito l'avvenuta eliminazione (figura 6.12).

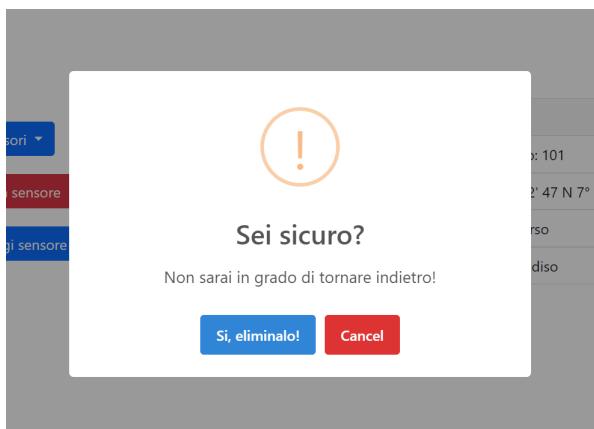


Figura 6.11: Form aggiungi sensore



Figura 6.12: Pop-up conferma

7 | GitHub Repository Info

Il progetto Sistema di Monitoraggio Ambientale è disponibile al seguente link: **Sistema di Monitoraggio Ambientale repository**. Per poter eseguire il progetto è sufficiente avviare il server mediante il comando **npm start** all'interno della cartella **api**. Per quanto riguarda il client, invece, è sufficiente aprire il file **index.html** contenuto nella cartella principale utilizzando un live server (consigliato quello di **vscode**).

8 | Testing

Per effettuare il testing abbiamo definito all'interno della cartella `api` uno script `test.js` che contiene i test delle quattro tipologie di api e uno script `index_testing.js` dove sono definite le api su cui viene eseguito. L'obiettivo di questo testing è quello di verificare la corretta esecuzione delle api `GET`, `POST`, `DELETE` e `PUT`.

Per eseguire il test abbiamo esteso il file di configurazione `package.json` con la seguente riga di codice
`"text" = "node test | tap-spec"`.

```
APIs Running

TEST1: correct fauna returned

Mongo DB Connection Successfull
✓ No error
✓ Fauna as expected

TEST2: correct sensor added

✓ No error
✓ Sensor added correctly

TEST3: sensor deleted

✓ No error
✓ Sensor deleted correctly

TEST4: risk modify

✓ No error
✓ Risk updated correctly
```

Figura 8.1: Esito del testing