UNIVERSITÁ POLITECNICA DELLE MARCHE

FACOLTÁ DI INGEGNERIA

Corso di Laurea Triennale in
Ingegneria Informatica e dell'Automazione

Progettazione e Sviluppo di una Web App per la Gestione di un Orto Botanico

Design and Development of a Web app to manage a Botanic Garden

Relatore:
Ch.mo Prof. Mancini Adriano

Laureando:
Rossi Andrea

Anno Accademico 2019-2020

Indice

Capitolo 1

Introduzione

1.1 Obiettivo

Lo svolgimento di questo progetto è focalizzato sullo sviluppo di una Applicazione Web che consenta di visualizzare piante che sono di interesse nell'ambito di orti botanico in rete. Quest'ultima deve essere in grado di poter elaborare e visionare dati riguardo quest'ambito, utilizzando specifici linguaggi e software studiati durante il percorso di studio.

1.2 Struttura della Tesi

La tesi viene strutturata principalmente come introduzione all'applicazione web. Verranno prima affrontati concetti elementari per poi introdurre più nozioni su tali elementi, ogni parte è propedeutica alla successiva. Viene quindi strutturata come segue:

- nella prima parte verranno introdotti gli strumenti software e i linguaggi utilizzati suddivisi lato front e back end;
- nella seconda parte verrà spiegato quali sono state le fasi della progettazione dell'applicazione web. Si parte dal database logic, poi si analizzano gli script php utilizzati nel back end infine il restante codice del front end;
- nella terza parte vengono espletati tutti gli accorgimenti necessari affinché si abbia un corretto set-up dell'applicazione web;
- nella quarta parte verranno espletati altri accorgimenti affinché si abbia la migliore esperienza di navigazione mobile dal browser Safari. Dopodiché verrà fatta una panoramica dell'applicazione web, guidando l'utente tramite l'uso di screen;

- nell'ultima parte verranno esposte le conclusioni e verranno introdotti gli sviluppi futuri;

1.3 La scelta dell'Applicazione Web

La scelta dell'applicazione web anziché un'applicazione software scritta in Java o C# è motivata dal fatto che essenzialmente un'applicazione web può essere eseguita su qualunque dispositivo dotato di un web browser. Altrimenti il percorso di sviluppo doveva essere suddiviso a seconda del dispositivo, quindi utilizzando Android Studio per un dispositivo Android, Xcode per un dispositivo Apple e via dicendo.

1.4 Indipendenza da software esterni per la gestione del database

Una cosa di fondamentale importanza che è stata implementata con successo in questo progetto è l'indipendenza da un modulo esterno per la gestione del database ad eccezione della creazione del database, tabelle e dati di default. Tale scelta comporta una maggiore flessibilità ed anche facilità per la visualizzazione e manipolazione dei dati del database senza dover ogni volta accedere a un client di gestione di basi di dati o al modulo PhpMyAdmin.

1.5 La scelta di impiegare due Front-end

Sono stati realizzati due front-end che consumano lo stesso back-end.
Il front-end **basic** come suggerisce il nome, è più basilare, è privo di effetti grafici e rispecchia una interfaccia web light.
Il front-end **full** invece è completo anche dal punto di vista grafico. Rispecchia gli standard attuali *Material Design* dell'odierna programmazione web.

Concludendo si può dire che sviluppare due front end è stato un ottimo esercizio per raggiungere gli stessi obiettivi utilizzando linguaggi di programmazione, costrutti ed approcci diversi per ogni front end.

Capitolo 2

Strumenti utilizzati

2.1 Front-end e back-end

I termini front end (in sigla **FE**) e back end (in sigla **BE**) denotano, rispettivamente, la parte visibile all'utente e con cui egli può interagire (interfaccia utente) e la parte che permette l'effettivo funzionamento di queste interazioni. Il front end, è responsabile dell'acquisizione dei dati di ingresso e della loro elaborazione tali da renderli utilizzabili dal back end.
Nella programmazione e sviluppo dei siti web viene definito front end la parte visibile da chiunque e raggiungibile all'indirizzo web del sito e viene definita back end la parte di amministrazione di un sito accessibile solo da amministratori del sito web. Front end e back end si utilizzano solamente quando il sito web è dinamico [8].

2.2 Back End

2.2.1 Software

Di seguito saranno introdotti vari strumenti software che hanno trovato ampio utilizzo per la realizzazione di questo progetto.

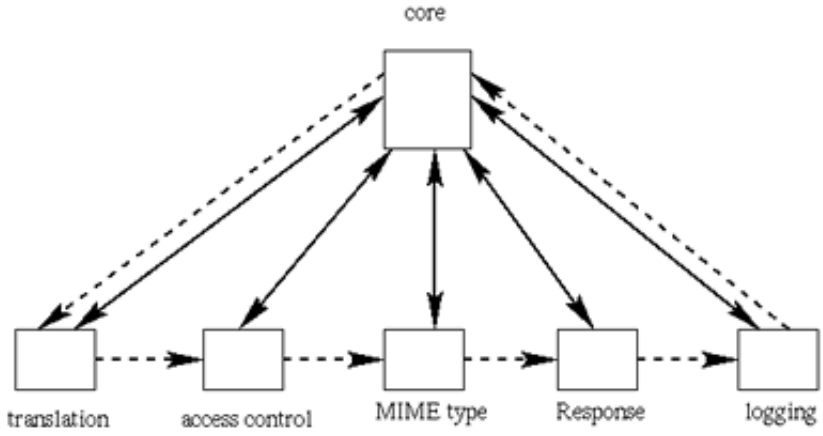
2.2.1.1 Web Server

Un server web (o web server) è un'applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser. La comunicazione tra server e client avviene tramite il protocollo **HTTP**, che utilizza la porta TCP 80 (o 8080), o eventualmente la versione sicura **HTTPS**, che utilizza invece la 443 [10].

In questo progetto è stato utilizzato **Apache HTTP Server**. Il Web Server Apache presenta un'architettura modulare, quindi ad ogni richiesta del client vengono svolte funzioni specifiche da ogni modulo di cui è composto, come unità indipendenti. Ciascun modulo si occupa di una funzionalità, ed il controllo è gestito dal core. In linea continua il flusso dei dati reale Tratteggiato il flusso dei dati astratto che forma la pipeline Al di sopra del ciclo del core un demone esegue un ciclo di polling, attraverso il quale vengono interrogate continuamente le linee logiche da cui possono pervenire messaggi di richiesta. Il core passa poi la richiesta ai vari moduli in modo sequenziale, usando i parametri di uscita di un modulo come parametri di accesso per il successivo, creando così l'illusione di una comunicazione orizzontale fra i moduli (Pipeline software) [5]. Le principali fasi di cui è composto il ciclo sono:

- **Translation**: traduce la richiesta del client **Access Control**: controlla le richieste in base ai criteri di autorizzazione.
- **MIME Type**: identifica il tipo di contenuto e decide quali moduli possono contribuire a servire la richiesta.
- **Response**: invia la risposta al client e attiva eventuali procedure.
- **Logging**: tiene traccia di tutto ciò che è stato fatto.

Figura 2.1: Architettura di Apache : In linea continua il flusso dei dati reale, tratteggiato il flusso dei dati astratto che forma la pipeline [5].



2.2.1.2 Database Management System

Un Database Management System, abbreviato in **DBMS** o Sistema di gestione di basi di dati, è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database e ospitato su architettura hardware dedicata oppure su semplice computer [7].

In questo progetto viene utilizzato **MySQL** che è un Relational database management system (**RDBMS**) composto da un client a riga di comando e un server. Entrambi i software sono disponibili sia per sistemi Unix e Unix-like sia per Windows.
MySQL è un software libero rilasciato a doppia licenza, compresa la GNU General Public License, ed è sviluppato per essere il più possibile conforme agli standard ANSI SQL e ODBC SQL [18].
I sistemi e i linguaggi di programmazione che supportano MySQL sono molto numerosi: ODBC, Java, Mono, .NET, PHP, Python e molti altri.
In MySQL una tabella può essere di diversi tipi (o **storage engine**). Ogni tipo di tabella presenta proprietà e caratteristiche differenti (transazionale o meno, migliori prestazioni, diverse strategie di locking, funzioni particolari, ecc). Degna di nota è sicuramente il tipo di tabella **InnoDB** [9], che mette a disposizione le seguenti funzionalità:

- transazioni SQL con savepoint e transazioni XA;
- lock a livello di record;
- chiavi esterne (**Foreign Key**);
- buffer per gli indici, le chiavi, modifiche, cancellazioni;
- compressione delle tabelle;
- Il lock in InnoDB per quanto riguarda i comandi SELECT è di tipo non-locking. InnoDB offre delle ottime performance in termini di prestazioni e utilizzo della CPU.

Per quanto riguarda i limiti di questo tipo di tabella, si ha:

- Impossibilità creare più di 1000 colonne per tabella;
- Su alcuni sistemi operativi le dimensioni del tablespace non possono superare i 2 GB;
- La grandezza di tutti i file di log di InnoDB deve essere inferiore ai 4 GB;
- La grandezza minima di un tablespace è di 10 MB;

- Gli indici FULLTEXT sono stati introdotti nella versione 5.6 di MySQL e nella versione 10.0 di MariaDB, ma in alcune circostanze sono meno efficienti rispetto a quelli di **MyISAM** (storage engine predefinito in MySQL dalla sua introduzione (versione 3.23) fino alla versione 5.5).

2.2.1.3 PhpMyAdmin

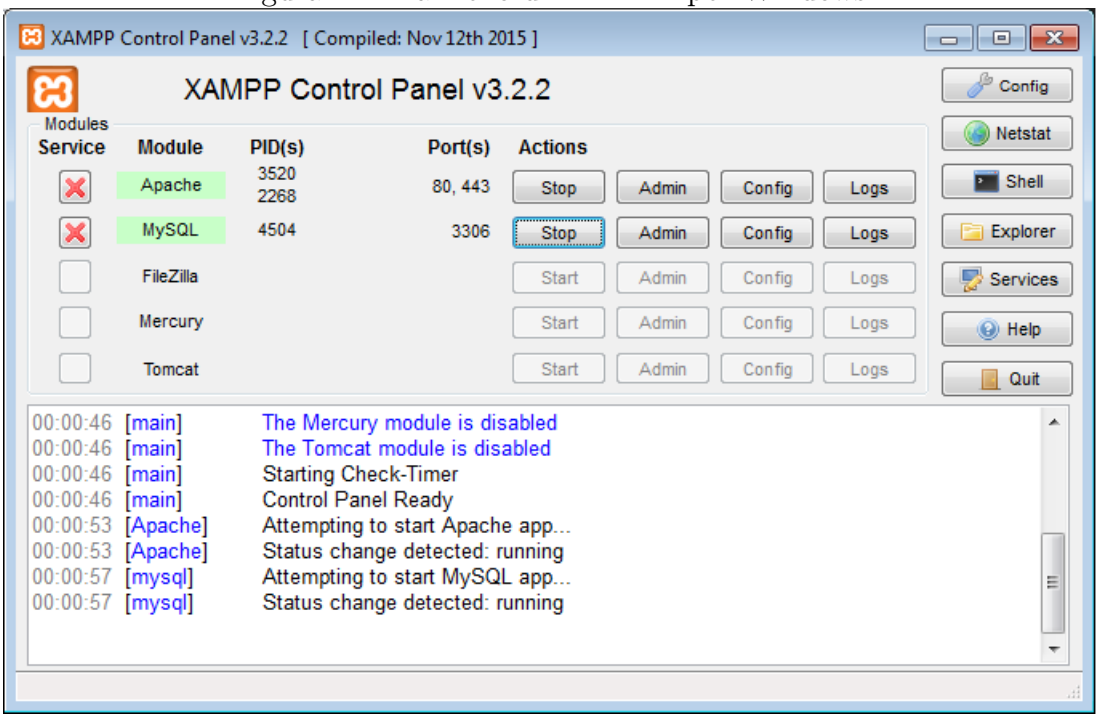
PhpMyAdmin è un'applicazione web scritta in PHP, distribuita con licenza GPL, che consente di amministrare un database MySQL o MariaDB tramite un qualsiasi browser.
PhpMyAdmin permette di creare un database da zero, creare le tabelle ed eseguire operazioni di ottimizzazione sulle stesse. Sono previste delle funzionalità per l'inserimento dei dati (popolazione del database), per le query, per il backup dei dati, e molto altro [19].

2.2.1.4 XAMPP

XAMPP è una piattaforma software multi-piattaforma e libera costituita da Apache HTTP Server, il gestore di database MariaDB (prima della versione 5.5.30 il gestore di database era MySQL) e tutti gli strumenti necessari per utilizzare i linguaggi di programmazione PHP e Perl. Il nome è un acronimo dei software sopra citati (la X sta per x-platform, l'abbreviazione di cross-platform in lingua inglese ovvero multipiattaforma) [17]. I componenti inclusi in questo pacchetto e utilizzati sono:

- il web server: **Apache** HTTP Server;
- il database management system : **MariaDB** e SQLite;
- come linguaggio di programmazione: **PHP**;
- come modulo **PhpMyadmin**.

Figura 2.2: Pannello di XAMPP per Windows



2.2.2 Linguaggi di Programmazione

Verranno ora introdotti i linguaggi di programmazione utilizzati nella parte di **BE**.

2.2.2.1 PHP

Il PHP (acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti; originariamente acronimo di "Personal Home Page") è un linguaggio di scripting interpretato, originariamente concepito per la programmazione di pagine web dinamiche. L'interprete PHP è un software libero distribuito sotto la PHP License.
PHP è in grado di interfacciarsi a innumerevoli DBMS tra cui MySQL [13].

2.2.2.2 SQL

L' SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- creare e modificare schemi di database (DDL - Data Definition Language);
- inserire, modificare e gestire dati memorizzati (DML - Data Manipulation Language);

- interrogare i dati memorizzati (DQL - Data Query Language);
- creare e gestire strumenti di controllo ed accesso ai dati (DCL - Data Control Language).

SQL è un linguaggio per interrogare e gestire basi di dati mediante l'utilizzo di costrutti di programmazione denominati query. Con SQL si leggono, modificano, cancellano dati e si esercitano funzioni gestionali ed amministrative sul sistema dei database [24]. Esso si divide in:

- Data Definition Language (DDL) - permette di creare e cancellare database o di modificarne la struttura;
- Data Manipulation Language (DML) - permette di inserire, cancellare, modificare i dati;
- Data Control Language (DCL) - permette di gestire gli utenti e i permessi;
- Query language (QL) - permette di interrogare il database, cioè di leggere i dati;
- Device Media Control Language (DMCL) - permette di controllare i supporti (memorie di massa) dove vengono memorizzati i dati.

2.3 Front End

2.3.1 Software

2.3.1.1 Web Browser

Il web browser (o più semplicemente browser) è un'applicazione per il recupero, la presentazione e la navigazione di risorse sul web. Tali risorse (come pagine web, immagini o video) sono messe a disposizione sul World Wide Web (la rete globale che si appoggia su Internet), su una rete locale o sullo stesso computer dove il browser è in esecuzione. Nell'architettura di rete **client-server** di Internet il browser rappresenta dunque il client che fa richieste di risorse web ai vari web server e application server ospitanti rispettivamente siti web e applicazioni web. Esso rappresenta dunque il sistema software di interfacciamento dell'utente con la rete che rende la navigazione dell'utente tipicamente user-friendly, sebbene ai primordi della rete siano esistiti anche browser testuali da riga di comando su shell. I browser vengono principalmente utilizzati su personal computer, ma anche su altri dispositivi che consentono la navigazione in Internet, come i palmari e gli smartphone. Quelli più noti e diffusi sono Internet Explorer, Mozilla Firefox, Google Chrome, Safari, Opera e Internet Explorer [6].

In questo progetto viene utilizzato **Mozilla Firefox** che è un web browser libero e multipiattaforma, mantenuto da Mozilla Foundation per la navigazione da desktop e il web browser **Safari** sviluppato dalla Apple Inc per i sistemi operativi iOS e macOS per la navigazione da mobile.

2.3.1.2 Editor

Viene utilizzato **Visual Studio Code** che è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. Esso include supporto per debugging, un controllo per Git integrato, Syntax highlighting, IntelliSense, Snippet e refactoring del codice. È anche personalizzabile: gli utenti possono cambiare il tema dell'editor, le scorciatoie da tastiera, e le preferenze. È un software libero, anche se la versione ufficiale è sotto una licenza proprietaria [2].
Questo editor supporta la colorazione della sintassi per i linguaggi utilizzati nel progetto (**Syntax highlighting**) come è possibile vedere nella seguente immagine.

Figura 2.3: script php (config.php) per la connessione al database

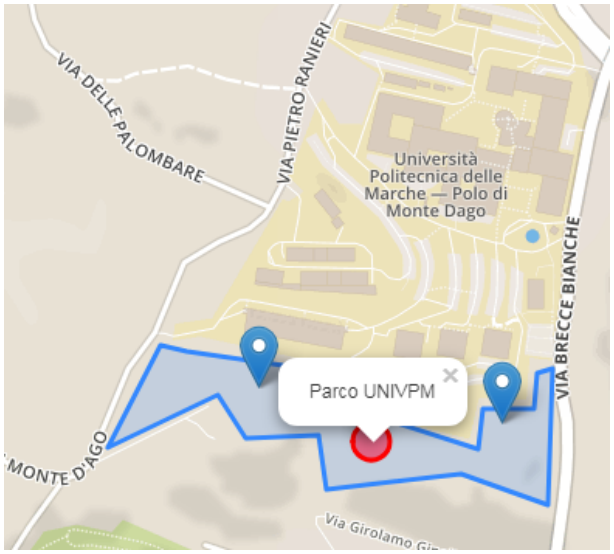
Config.php

```php
<?php

    $mySQL["server"] = 'localhost';
    $mySQL["username"] = 'root';
    $mySQL["password"] = '';
        $mySQL["database"] = 'orto_botanico';


    try
    {
        $dbhandle = mysqli_connect($mySQL["server"], $mySQL["username"], $mySQL["password"])  or die("Unable to connect to MySQL");
        $selected = mysqli_select_db($dbhandle, $mySQL["database"]);
    }
    catch (Exception $e)
    {
        echo "<script> alert('".$e->getMessage()."')</script>";
    }

?>
```

2.3.1.3 Leaflet

Leaflet è una libreria JavaScript per sviluppare mappe geografiche interattive (WebGIS). Sviluppato dal 2010, supporta la maggior parte dei browser e degli standard HTML5 e CSS3. Permette di mostrare punti di interesse, linee o aree, o strutture dati come file GeoJSON, o livelli interattivi, su una mappa a tasselli [12].
Punti di interesse (markers) e livelli (layers) possono essere aggiunti successivamente, in questo progetto viene fatta ampio uso di questa metodologia, aggiungendo e rimuovendo dinamicamente punti di interesse.

Figura 2.4: immagine raffigurante dei punti di interesse: è possibile vedere il **polygon** che raffigura l'orto botanico *Parco UNIVPM* con al centro un **circle** che cliccato apre il tooltip e i due **markers** che raffigurano i beacon (piante) nel parco.



2.3.1.4 W3C Geolocation API

La W3C Geolocation API è una libreria del World Wide Web Consortium (W3C) per standardizzare un'interfaccia per recuperare le informazioni sulla posizione geografica di un dispositivo lato client. Le fonti più comuni di informazione sulla posizione dell'utente vengono stimate tramite :

- indirizzo IP;
- Wi-Fi;
- indirizzo MAC Bluetooth;
- identificazione radio-frequenza (RFID);
- posizione della connessione Wi-Fi;
- Global Positioning System (GPS);
- ID cella GSM / CDMA.

La posizione dell'utente verrà restituita con una precisione determinata in base alla migliore fonte di informazione sulla posizione disponibile [23].

2.3.1.5 WebRTC API

WebRTC (Web Real-Time Communications) è una tecnologia che consente alle applicazioni Web e ai siti di acquisire e, a scelta, trasmettere contenuti audio e / o video, nonché di scambiare dati arbitrari tra browser senza richiedere un intermediario. L'insieme di standard che comprende WebRTC consente di condividere dati ed eseguire teleconferenze peer-to-peer, senza richiedere che l'utente installi plug-in o qualsiasi altro software di terze parti [3] [15].

2.3.2 Linguaggi Utilizzati

2.3.2.1 HTML

L'HyperText Markup Language è un **linguaggio di markup** che descrive le modalità di impaginazione o visualizzazione grafica (layout) del contenuto, testuale e non, di una pagina web attraverso tag di formattazione. Sebbene l'HTML supporti l'inserimento di script e oggetti esterni quali immagini o filmati, non è un linguaggio di programmazione: non prevedendo alcuna definizione di variabili, strutture dati, funzioni o strutture di controllo che possano realizzare programmi, il suo codice è in grado soltanto di strutturare e decorare dati testuali. Quando un documento ipertestuale scritto in HTML è memorizzato in un file la sua estensione è tipicamente **.html** o **.htm**. I documenti HTML vengono immagazzinati sui dischi rigidi di macchine elaboratrici (computer-server) costantemente collegate e connesse alla rete Internet. Su queste macchine è installato un software specifico (web server) che si occupa di produrre e inviare i documenti ai browser degli utenti che ne fanno richiesta usando il protocollo HTTP per il trasferimento dati [20].

2.3.2.2 CSS

Il CSS (acronimo di Cascading Style Sheets, in italiano fogli di stile a cascata), è un linguaggio usato per definire la **formattazione** di documenti HTML, XHTML e XML. Le regole per comporre il CSS sono contenute in un insieme di direttive (Recommendations) emanate a partire dal 1996 dal W3C. L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione [22]. Nessun browser attuale offre il supporto completo e corretto delle specifiche CSS. Tuttavia esistono browser che si avvicinano molto a questo risultato ed altri che invece ne sono molto lontani. La lista che segue è di motori di rendering perché a loro è assegnato il compito di formattare la pagina secondo le istruzioni CSS.

2.3.2.3 CSS Framework CSS

Un framework CSS è un software predisposto per consentire un web design più semplice e conforme agli standard utilizzando il linguaggio CSS. La maggior parte di questi framework contiene almeno una griglia. I framework più funzionali includono anche più funzionalità e funzioni aggiuntive basate su JavaScript e JQuery, ma sono principalmente orientati al design e non invadenti.
Due esempi ampiamente utilizzati sono Bootstrap e **Materialize** (utilizzato in questo progetto) [4].

2.3.2.4 Javascript

JavaScript è un linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici interattivi tramite funzioni di script invocate da eventi innescati a loro volta in vari modi dall'utente sulla pagina web in uso (mouse, tastiera, caricamento della pagina ecc...).
Tali funzioni di script, utilizzati dunque nella logica di presentazione, possono essere opportunamente inserite in file HTML, in pagine JSP o in appositi file separati con estensione .js poi richiamati nella logica di business.
Un uso principale del JavaScript in ambito Web è la scrittura di piccole funzioni integrate nelle pagine HTML che interagiscono con il DOM del browser per compiere determinate azioni non possibili con il solo HTML statico: controllare i valori nei campi di input, nascondere o visualizzare determinati elementi, ecc.
Sfortunatamente, gli standard DOM imposti dal W3C non sempre vengono rispettati dai vari browser: browser diversi (anche a seconda del loro motore di rendering) espongono diversi oggetti o metodi allo script (Internet Explorer è solito aderire agli standard con piccole modifiche, e tratta ad esempio l'oggetto event come globale), ed è quindi spesso necessario implementare controlli aggiuntivi ad una funzione JavaScript, per garantirne la compatibilità con ciascun browser [21].

2.3.2.5 JQuery

JQuery è una libreria JavaScript per applicazioni web. Nasce con l'obiettivo di semplificare la selezione, la manipolazione, la gestione degli eventi e l'animazione di elementi DOM in pagine HTML, nonché implementare funzionalità **AJAX** [11].

2.3.2.6 AJAX

AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive (Rich Internet Application). Lo sviluppo di applicazioni HTML con AJAX si basa su uno scambio di dati in background fra web browser e server, che consente l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente.
La tecnica AJAX utilizza una combinazione di:

- HTML (o XHTML) e CSS per il markup e lo stile;
- DOM (Document Object Model) manipolato attraverso un linguaggio ECMAScript come JavaScript o JScript per mostrare le informazioni ed interagirvi;

- l'oggetto XMLHttpRequest per l'interscambio asincrono dei dati tra il browser dell'utente e il web server. In alcuni framework AJAX e in certe situazioni, può essere usato un oggetto Iframe invece di XMLHttpRequest per scambiare i dati con il server e, in altre implementazioni, tag <script> aggiunti dinamicamente (JSON);

- in genere viene usato XML come formato di scambio dei dati, anche se di fatto qualunque formato può essere utilizzato, incluso testo semplice, HTML preformattato, JSON e perfino EBML. Questi file sono solitamente generati dinamicamente da script lato server.

AJAX non è una tecnologia individuale, quanto piuttosto un gruppo di tecnologie utilizzate insieme.
Le applicazioni web che usano AJAX richiedono browser che supportano le tecnologie necessarie (quelle dell'elenco sopra).
Questi browser includono: Firefox, Opera, Konqueror, Safari, Internet Explorer e Chrome. Tuttavia, per specifica, *Opera non supporta la formattazione degli oggetti XSL* [1].

Capitolo 3

Progettazione e Sviluppo

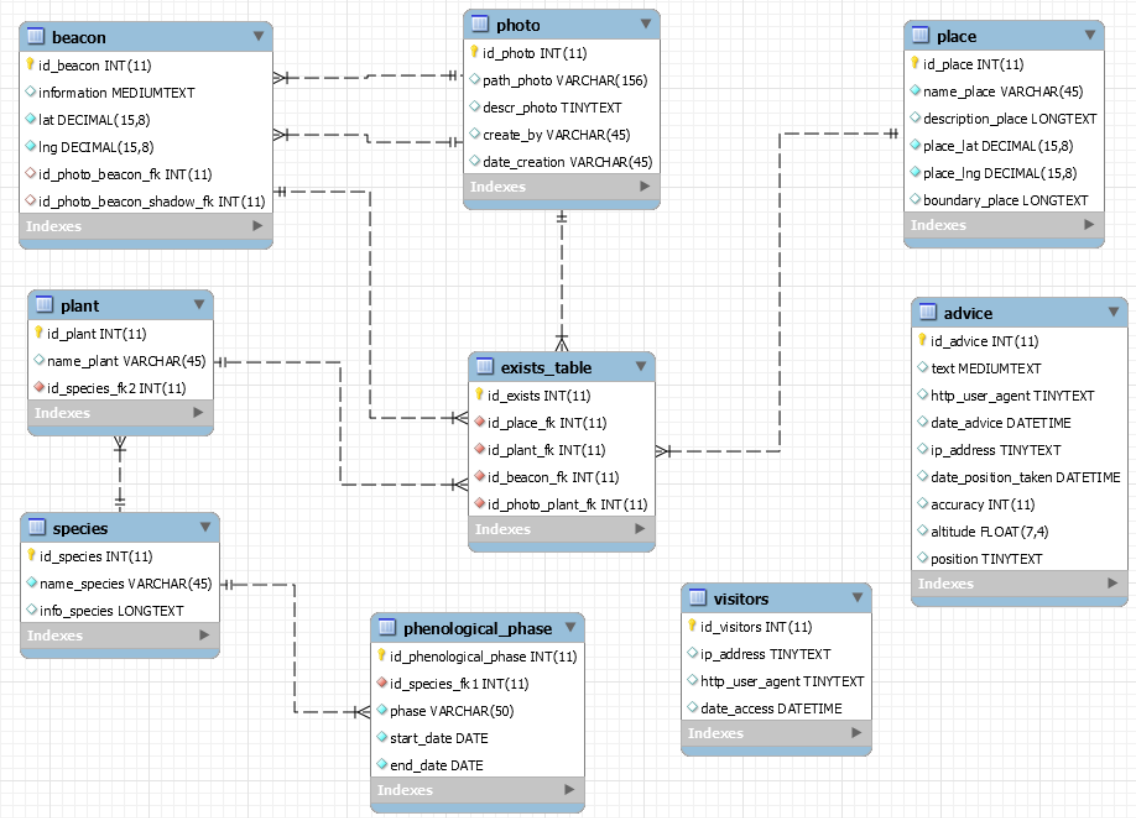
In questo capitolo verrà trattato l'approccio utilizzato per sviluppare dalle fondamenta l'applicazione web. Verrà quindi spiegato passo passo come è stata realizzata l'applicazione web e verranno forniti i codici utilizzati per una maggiore chiarezza.
Si parte analizzando il livello di back-end più alto, quello del database.

3.1 Database Logic

3.1.0.1 Contesto e Esigenze del Database

Il database deve rappresentare astrattamente la situazione che si sta studiando. Essendo un orto botanico composto essenzialmente dalle piante, si parte da quest'entità a sviluppare il modello della base di dati. La logica che c'è a questo livello prevede che una pianta sia di una data specie e che idealmente abbia un dispositivo elettronico per permettere l'interazione telematica (beacon). Queste piante faranno parte di un dato sito botanico ed è infatti qua che nasce una rappresentazione astratta di tipo n:m tra pianta e sito.
Un'esigenza del database riguarda l'engine utilizzato per le tabelle, che è del tipo **InnoDB**, dato che si fa ampio uso delle chiavi esterne (*foreign keys*). Tali Foreign Keys permettono al meglio di realizzare una base di dati astratta conforme alla realtà di nostro interesse.

3.1.1 Modello E-R



3.1.2 Tabelle

Di seguito verranno analizzate e spiegate, una ad una tutte le tabelle utilizzate nella base di dati.

3.1.2.1 Photo

Questa tabella rappresenta i contenuti multimediali che vengono caricati dinamicamente nel web server.
È composta da:

- **id_photo** : chiave primaria univoca intera, not null e auto incrementante per distinguere univocamente ogni oggetto registrato nel database;
- **path_photo** : il percorso dell'oggetto multimediale nel nostro server, questa colonna è unique, dato che non possono esserci due oggetti con lo stesso percorso;
- **descr_photo** : la descrizione dell'oggetto caricato;

- **create_by** : campo che indica da chi è stato inserito l'oggetto nel server;
- **date_creation** : la data di inserimento, nel formato YYYY-MM-DD hh:mm:ss.

```
CREATE TABLE `photo` (
`id_photo` int(11) NOT NULL AUTO_INCREMENT,
`path_photo` varchar(156),
`descr_photo` tinytext,
`create_by` varchar(45),
`date_creation` varchar(45),
PRIMARY KEY (`id_photo`),
UNIQUE (`path_photo`)
) ENGINE=InnoDB;

```

Listing 3.1: Codice SQL della Tabella Photos

3.1.2.2 Beacon

Questa Tabella rappresenta i beacon nella nostra mappa. Ci sono due chiavi esterne che fanno referenza alla tabella photo.
Troviamo le seguenti colonne:

- **id_beacon** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni beacon registrato nel database;
- **information** : informazioni sul beacon;
- **lat** : la latitudine, non può essere null;
- **lng** : la longitudine, non può essere null;
- **id_photo_beacon_fk**: chiave esterna che può anche essere null. In tal caso il beacon avrà l'immagine di default, diversamente si avrà l'immagine specificata;
- **id_photo_beacon_shadow_fk** : chiave esterna che può anche essere null. In tal caso il beacon avrà l'immagine ombra di default, diversamente si avrà l'immagine specificata.

```

CREATE TABLE `beacon` (
`id_beacon` int(11) NOT NULL AUTO_INCREMENT,
`information` mediumtext,
`lat` decimal(15,8) NOT NULL,
`lng` decimal(15,8) NOT NULL,
`id_photo_beacon_fk` int(11) DEFAULT NULL,
```

```
8  `id_photo_beacon_shadow_fk` int(11) DEFAULT NULL,
9  PRIMARY KEY (`id_beacon`),
10 CONSTRAINT `id_photo_beacon_fk` FOREIGN KEY (`id_photo_beacon_fk`)
       REFERENCES `photo` (`id_photo`) ON UPDATE CASCADE,
11 CONSTRAINT `id_photo_beacon_shadow_fk` FOREIGN KEY (`
       id_photo_beacon_shadow_fk`) REFERENCES `photo` (`id_photo`) ON
       UPDATE CASCADE
12 ) ENGINE=InnoDB;
```

Listing 3.2: Codice SQL della Tabella Beacon

3.1.2.3 Species

Questa Tabella indica le specie degli alberi dei nostri orti di interesse.
È formata dalle colonne:

- **id_species** : chiave primaria univoca intera, not null e auto incrementante per distinguere univocamente ogni specie registrata nel database;
- **name_species** : il nome della specie;
- **info_species** : informazioni degne di nota sulla specie;

```
1 CREATE TABLE `species` (
2 `id_species` int(11) NOT NULL AUTO_INCREMENT,
3 `name_species` varchar(45) NOT NULL,
4 `info_species` longtext,
5 PRIMARY KEY (`id_species`)
6 ) ENGINE=InnoDB;
```

Listing 3.3: Codice SQL della Tabella Species

3.1.2.4 Phenological Phase

Questa tabella rappresenta le possibili fasi fenologiche delle specie registrata nel database. Ha una chiave esterna che la collega con la tabella **species**.
È formata dalle colonne:

- **id_phenological_phase** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni fase fenologica registrato nel database;
- **id_species_fk1** : chiave esterna che fa riferimento a id_species e server per capire quale specie ha una data fase fenologica;
- **phase** : la fase fenologica;
- **start_date** l'inizio della fase fenologica in formato YYYY-MM-DD;

- **end_date**.

Tutte le colonne sono not null. È importante far notare che se si vuole registrare una fase fenologica che scavalchi l'anno, non è possibile farlo con un solo record (es: inizio 1 Dicembre, fine 1 Febbraio) ma si inseriscono due record spezzandoli alla fine dell'anno come segue :

- inizio 1 Dicembre, fine 31 Dicembre;
- inizio 1 Gennaio , fine 1 Febbraio.

```
CREATE TABLE `phenological_phase` (
`id_phenological_phase` int(11) NOT NULL AUTO_INCREMENT,
`id_species_fk1` int(11) NOT NULL,
`phase` varchar(50) NOT NULL,
`start_date` date NOT NULL,
`end_date` date NOT NULL,
PRIMARY KEY (`id_phenological_phase`),
CONSTRAINT `id_species_fk1` FOREIGN KEY (`id_species_fk1`)
    REFERENCES `species` (`id_species`) ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Listing 3.4: Codice SQL della Tabella Phenological Phase

3.1.2.5 Place

Questa tabella rappresenta gli orti botanici.
È formata dalle colonne:

- **id_place** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni place registrato nel database;
- **name_place** : nome del place. Non possono esserci duplicati del nome dato che ha attrbuto unique;
- **description_place** : descrizione del place;
- **place_lat** : latitudine del place;
- **place_lng** : longitudine del place;
- **boundary_place**: insieme cordinate che permettono a leaflet di disegnare l'elemento sulla mappa, questa colonna è di tipo longtext.

```

CREATE TABLE `place` (
`id_place` int(11) NOT NULL AUTO_INCREMENT,
`name_place` varchar(45) NOT NULL,
`description_place` longtext,
```

```
`place_lat` decimal(15,8) NOT NULL,
`place_lng` decimal(15,8) NOT NULL,
`boundary_place` longtext,
PRIMARY KEY (`id_place`),
UNIQUE (`name_place`)
) ENGINE=InnoDB;
```

Listing 3.5: Codice SQL della Place Photos

3.1.2.6 Plant

Questa tabella rappresenta una pianta. È collegata tramite chiave esterna alla tabella species.
È formata dalle colonne:

- **id_plant** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni pianta registrata nel database;
- **name_plant** : nome fittizio della pianta;
- **id_species_fk2** : chiave esterna che indica la specie della pianta.

Da notare che la colonna *id_species* della tabella *species* viene referenziata due volte, per evitare di incorrere nell'errore **Duplicate key on write or update** si distinguono rispettivamente **id_species_fk1** e **id_species_fk2**.

```
CREATE TABLE `plant` (
`id_plant` int(11) NOT NULL AUTO_INCREMENT,
`name_plant` varchar(45) DEFAULT NULL,
`id_species_fk2` int(11) NOT NULL,
PRIMARY KEY (`id_plant`),
CONSTRAINT `id_species_fk2` FOREIGN KEY (`id_species_fk2`)
    REFERENCES `species` (`id_species`) ON UPDATE CASCADE
) ENGINE=InnoDB;
```

Listing 3.6: Codice SQL della Tabella Plant

3.1.2.7 exists_table

Un record di questa tabella indica effettivamente che esiste una pianta, in un dato orto dove è stato posizionato un beacon.
Per fare ciò, questa tabella ha come chiave esterna la chiave primaria della tabella **beacon** e dato che funge da appoggio alla relazione n:m tra le tabelle **plant** e **place** conterrà le loro chiavi primarie come chiavi esterne.

È quindi formata dalle seguenti colonne:

- **id_exists** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni esistenza di un elemento di interesse completo, localizzabile nella mappa, registrato nel database;
- **id_place_fk** : chiave esterna che fa riferimento a un dato place;
- **id_plant_fk** : chiave esterna che fa riferimento a una data pianta;
- **id_beacon_fk** : chiave esterna che fa riferimento a un dato beacon;
- **id_photo_plant_fk** : chiave esterna che fa riferimento alla foto della pianta.

Da notare che non è possibile chiamare la tabella solamente **exists**, dato che in SQL è una parola chiave.

```
1
2 CREATE TABLE `exists_table` (
3 `id_exists` int(11) NOT NULL AUTO_INCREMENT,
4 `id_place_fk` int(11) NOT NULL,
5 `id_plant_fk` int(11) NOT NULL,
6 `id_beacon_fk` int(11) NOT NULL,
7 `id_photo_plant_fk` int(11) NOT NULL,
8 PRIMARY KEY (`id_exists`),
9 CONSTRAINT `id_beacon_fk` FOREIGN KEY (`id_beacon_fk`) REFERENCES `
    beacon` (`id_beacon`) ON UPDATE CASCADE,
10 CONSTRAINT `id_plant_fk` FOREIGN KEY (`id_plant_fk`) REFERENCES `
    plant` (`id_plant`) ON UPDATE CASCADE,
11 CONSTRAINT `id_place_fk` FOREIGN KEY (`id_place_fk`) REFERENCES `
    place` (`id_place`) ON UPDATE CASCADE,
12 CONSTRAINT `id_photo_plant_fk` FOREIGN KEY (`id_photo_plant_fk`)
    REFERENCES `photo` (`id_photo`) ON UPDATE CASCADE
13 ) ENGINE=InnoDB;
```

Listing 3.7: Codice SQL della Tabella exists_table

3.1.2.8 Visitors

Questa tabella serve a tener conto di tutti i visitatori del sito. Non presenta collegamenti con le altre tabelle.
È formata dalle colonne:

- **id_visitors** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni visita registrato nel database;
- **ip_address** : indirizzo IP del dispositivo che ha visitato l'applicazione web.
  Questa colonna risulta particolarmente utile per più motivi, in primis è possibile bloccare la visualizzazione del sito a tutti gli IP che provano ad

effettuare continue connessioni al fine di rallentare il sito (**DDos attack**). Inoltre per fini statistici è possibile sapere in quali zone viene consultato maggiormente l'applicazione web tramite servizi quali **IP Locator Finder**;

- **http_user_agent** : stringa che racchiude informazioni come il nome dell'applicazione client, la versione, il sistema operativo e la lingua.
  L'user Agent di un dispositivo desktop con Windows 7 e Mozilla Firefox è del tipo : **Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:62.0) Gecko/20100101 Firefox/62.0**. Anche questa colonna desta un particolare interesse: è possibile fare un'indagine statistica per vedere quali sono i browser e dispositivi più utilizzati per sviluppi futuri mirati;

- **date_access** : Data in formato YYYY-MM-DD hh:mm:ss che viene registrata appena si accede al sito.

```
1 CREATE TABLE `visitors` (
2 `id_visitors` int(11) NOT NULL AUTO_INCREMENT,
3 `ip_address` tinytext DEFAULT NULL,
4 `http_user_agent` tinytext DEFAULT NULL,
5 `date_access` datetime DEFAULT NULL,
6 PRIMARY KEY (`id_visitors`)
7 ) ENGINE=InnoDB;
```

Listing 3.8: Codice SQL della Tabella Visitors

3.1.2.9 Advice

Questa tabella rappresenta un consiglio o un report delle funzionalità del sito, espressi dagli utenti. Non è collegata ad altre tabelle, ed è composta come segue:

- **id_advice** : chiave primaria univoca intera, not null e auto incrementante per distinguare univocamente ogni report registrato nel database;

- **text** : testo del report inserito dall'utente;

- **http_user_agent** : stringa user agent di chi ha effettuato il report;

- **date_advice** : la data del report nel formato YYYY-MM-DD hh:mm:ss;

- **ip_address** : indirizzo IP di chi ha fatto il report;

- **date_position_taken** : data in formato YYYY-MM-DD hh:mm:ss di quando è stata presa la posizione dell'utente;

- **accuracy** : accuratezza della posizione dell'utente;

- **altitude** : altitudine dell'utente;

- **position** : posizione dell'utente nel formato LatLng(XX.XXXXXX, YY.YYYYYY).

Se Durante la navigazione sul sito, l'utente non ha acconsentito alla richiesta della Geolocation API di trovare la sua posizione, le colonne **date_position_taken**, **accuracy**, **altitude** e **position** saranno **null**.
Dato che per ogni visita al sito web viene registrato l'indirizzo IP e la data di accesso nella tabella **visitors**, se si vorrebbe conoscere la tempistica dell'utente a trovare il malfunzionamento e segnalarlo basta confrontare le rispettive colonne e trovare la differenza di tempo. (Se è un lasso di tempo breve è ovvio che è un falso positivo)

```
CREATE TABLE `advice` (
`id_advice` int(11) NOT NULL AUTO_INCREMENT,
`text` mediumtext,
`http_user_agent` tinytext  DEFAULT NULL,
`date_advice` datetime DEFAULT NULL,
`ip_address` tinytext DEFAULT NULL ,
`date_position_taken` datetime DEFAULT NULL,
`accuracy` int(11) DEFAULT NULL,
`altitude` FLOAT(7,4) DEFAULT NULL,
`position` tinytext DEFAULT NULL,
PRIMARY KEY (`id_advice`)
) ENGINE=InnoDB;
```

Listing 3.9: Codice SQL della Tabella Advice

3.1.3 Schema Logico

- **photo** (**id_photo**, path_photo, descr_photo, create_by, date_creation);
- **beacon** (**id_beacon**, information, lat, lng, *id_photo_beacon_fk*, *id_photo_beacon_shadow_fk*);
- **species** (**id_species**, name_species, info_species);
- **phenological_phase** (**id_phenological_phase**, *id_species_fk1*, phase, start_date, end_date);
- **place** (**id_place**, name_place, description_place, place_lat, place_lng, boundary_place);
- **plant** (**id_plant**, name_plant, *id_species_fk2*);
- **exists_table** (**id_exists**, *id_place_fk*, *id_plant_fk*, *id_beacon_fk*, *id_photo_plant_fk*);
- **visitors** (**id_visitors**, ip_address, http_user_agent, date_access);
- **advice** (**id_advice**, text, http_user_agent, date_advice, ip_address, date_position_taken, accuracy, altitude, position).

3.1.4 Query

Verranno ora illustrate le query più importanti utilizzate per elaborare i dati e consentire un corretto funzionamento dell'applicazione web.

3.1.4.1 Possibili valori di una Foreign Key

Durante lo sviluppo del progetto, è tornato utile un approccio dinamico per ricavare tutti i possibili valori che potesse assumere una chiave esterna. Data una chiave esterna, per ottenere la tabella e colonna referenziata, si è utilizzato il seguente codice SQL:

```
SELECT k.REFERENCED_TABLE_NAME, k.REFERENCED_COLUMN_NAME
FROM information_schema.TABLE_CONSTRAINTS i
LEFT JOIN information_schema.KEY_COLUMN_USAGE k
ON i.CONSTRAINT_NAME = k.CONSTRAINT_NAME
WHERE i.CONSTRAINT_TYPE = 'FOREIGN KEY'
AND i.CONSTRAINT_NAME ='nome_della_colonna';
```

Dopodichè con le colonne **REFERENCED_TABLE_NAME** e **REFERENCED_COLUMN_NAME** si costruisce la query finale:

```
SELECT REFERENCED_COLUMN_NAME FROM REFERENCED_TABLE_NAME;
```

Il nome della colonna sarà generato dinamicamente a seconda del caso.

3.1.4.2 Piante di un dato Place con una data Fase Fenologica e una data Specie

```
SELECT id_beacon,information,lat,lng,id_photo_beacon_fk,
    id_photo_beacon_shadow_fk
FROM beacon,exists_table,plant,phenological_phase,species,place
WHERE species.id_species = 'specie_selezionata'
AND beacon.id_beacon = exists_table.id_beacon_fk
AND exists_table.id_plant_fk = plant.id_plant
AND plant.id_species_fk2 = phenological_phase.id_species_fk1
AND phenological_phase.phase = 'fase_fenologica_selezionata'
AND (MONTH(curdate()) BETWEEN MONTH(start_date) AND MONTH(end_date))
AND (DAY(curdate()) BETWEEN DAY(start_date) AND DAY(end_date))
AND exists_table.id_place_fk = place.id_place
AND place.id_place ='place_selezionato';
```

Il place, fase fenologia e specie vengono selezionate dall'utente e la query viene generata dinamicamente.

3.2 Back-End

In questa sezione verranno analizzati gli script php che compongono una parte di back-end dell'applicazione web.
Il requisito essenziale affinchè questa parte funzioni è il database: quasi tutti gli script php faranno ampio uso, tramite le funzioni del linguaggio stesso, della base di dati.

Come precedentemente introdotto, la parte di back-end è la parte interna di un sistema dove l'utente interagisce indirettamente, in generale attraverso l'utilizzo di una applicazione front-end (che verrà analizzato nella sezione successiva).
La parte back-end di scripting language PHP impiegata si trova nelle cartelle *php*, una situata nella directory principale del progetto e altre due, situate rispettivamente nelle due varianti del progetto.
È importante dire che, anche se non è stato introdotto, in questo progetto vengono utilizzati due front-end. Entrambi condividono i file php nella directory principale e utilizzano rispettivamente i rimanenti script php nella loro directory.
All'interno della directory generale sono presenti vari file php, ora saranno analizzati e spiegati e laddove fossero di notevole importanza saranno elencati i codici.

3.2.1 Config.php

Questo script contiene le informazioni di autenticazione per stabilire la connessione con il database. Tali informazioni sono : server, username password, database.
Viene incluso tramite comando **include** ogni volta che è necessario.
Il codice di questo file php è mostrato interamente nella figura dell'editor.

3.2.2 add.php

Questo script si occupa di inserire un record in una data tabella del database.
Riceve tramite metodo GET la tabella e i valori da inserire, verrà costruita la stringa come segue:

```
1 $sql = "INSERT INTO ".$table_name." VALUES(".$values.")";
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.3 delete.php

Questo script si occupa di eliminare un record da una data tabella con un dato id dal database.
Riceve tramite metodo GET la tabella da dove si vuole eliminare il record, il

nome della colonna della primary key e il valore della primary key.
La query viene costruita come segue:

```
$sql = "DELETE FROM ". $table_name .
       "WHERE ".$table_name."."$pk_name." = '".$pk_value."'";
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.4 upload.php

Questo script si occupa di aggiornare il valore di una data colonna di un record, di una data tabella.
Riceve tramite metodo GET la tabella dove si vuole eseguire l'operazione, il nome della colonna da aggiornare, il nuovo valore della colonna, il nome della colonna della primary key e il valore della primary key.
La query viene costruita come segue:

```
$sql = "UPDATE ".$table_name." SET ". $column_name." = '".$new_value
    ."' WHERE ".$table_name."."$pk_name." = ".$pk_value;
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.5 rename_t.php

Questo script si occupa di rinominare una data tabella con un nuovo valore, entrambi forniti come parametri GET.
La query viene costruita come segue:

```
$sql = "RENAME TABLE ". $table_name ." TO ".$new_value;
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.6 rename_c.php

Questo script si occupa di rinominare una data colonna di una tabella con un nuovo valore. Tutte le informazioni sono fornite come parametri GET.
Dato che per rinominare una colonnna va specificato anche il tipo, per semplificare l'operazione all'utente, è il sistema che recupera (tramite passaggio intermedio) il tipo della colonna interessata. La query viene costruita come segue:

```
$sql = "DESCRIBE ". $table_name ." ".$column_name;

$query_data_type = mysqli_query($dbhandle,$sql);

$type = mysqli_fetch_array($query_data_type,MYSQLI_ASSOC);

```

```
7 $sql = "ALTER TABLE ".$table_name." CHANGE ". $column_name." ".
    $new_value." ".$type["Type"];
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.7 type.php

Questo script si occupa di cambiare il tipo di una data colonna, di una data tabella. Il nuovo tipo, il nome della colonna e il nome della tabella vengono forniti come parametri GET.
La query viene costruita come segue:

```
1 $sql = "ALTER TABLE ". $table_name ." CHANGE ".$column_name." ".
    $column_name." ".$new_value;
```

Eseguita la query, in caso di errore, si effettua un redirect allegando al link un parametro contenente tale errore.

3.2.8 function.php

Questo script definisce delle funzioni che vengono richiamate spesso dagli altri script php.
Ora verranno illustrate alcune delle funzioni più utilizzate nel progetto:

3.2.8.1 Ottenere l'IP Address

Per ottenere l'indirizzo IP del client vanno fatto alcune considerazioni analizzando le risorse che PHP mette a disposizione.

1. **$_SERVER['REMOTE_ADDR']** : Contiene l'indirizzo IP reale del client, ed è il valore più affidabile che si può trovare.
2. **$_SERVER['REMOTE_HOST']** : Contiene il nome dell'host da cui l'utente sta visualizzando la pagina corrente. Ma per far funzionare questo script, è necessario configurare il modulo Hostname Lookups all'interno del file di configurazione di apache *httpd.conf*.
3. **$_SERVER['HTTP_CLIENT_IP']** : Contiene l'indirizzo IP del client quando l'utente proviene dallo Shared Internet Services.
4. **$_SERVER['HTTP_X_FORWARDED_FOR']** : Contiene l'indirizzo IP del client quando sta utilizzando un server proxy.

Combinando queste informazioni è possibile realizzare un script PHP per ottenere l'indirizzo IP reale dagli utenti che stanno visualizzando l'applicazioni web da posizioni diverse. [14] Il codice utilizzato è il seguente:

```php
function get_client_ip() {
$ipaddress = '';
if (isset($_SERVER['HTTP_CLIENT_IP']))
$ipaddress = $_SERVER['HTTP_CLIENT_IP'];
else if(isset($_SERVER['HTTP_X_FORWARDED_FOR']))
$ipaddress = $_SERVER['HTTP_X_FORWARDED_FOR'];
else if(isset($_SERVER['HTTP_X_FORWARDED']))
$ipaddress = $_SERVER['HTTP_X_FORWARDED'];
else if(isset($_SERVER['HTTP_FORWARDED_FOR']))
$ipaddress = $_SERVER['HTTP_FORWARDED_FOR'];
else if(isset($_SERVER['HTTP_FORWARDED']))
$ipaddress = $_SERVER['HTTP_FORWARDED'];
else if(isset($_SERVER['REMOTE_ADDR']))
$ipaddress = $_SERVER['REMOTE_ADDR'];
else
$ipaddress = 'UNKNOWN';
return $ipaddress;
}
```

È da far notare che è possibile aggirare questo sistema con l'estensione **Requestly**, disponibile sia per Mozilla Firefox che per Chrome, settando **Client-ip header** con **111.111.111.111**.
Essendo questo progetto uno scopo accademico, non si è dato peso a tale problematica.
Tuttavia qualora si volesse rendere affidabile l'identificazione del client, dato che tramite tricks o hacks è possibile settare l'HTTP header information (es. $_SERVER['HTTP_...) come si vuole, è molto più affidabile usare **$_SERVER ['REMOTE_ADDR']**, in quanto questo non può essere impostato dall'utente.
Un'alternativa più sicura potrebbe essere la seguente:

```php
if (!empty($_SERVER['HTTP_CLIENT_IP'])) {
  $ip = $_SERVER['HTTP_CLIENT_IP'];
} elseif (!empty($_SERVER['HTTP_X_FORWARDED_FOR'])) {
  $ip = $_SERVER['HTTP_X_FORWARDED_FOR'];
} else {
  $ip = $_SERVER['REMOTE_ADDR'];
}
```

3.2.8.2 Ottenere la data e ora corrente

Per ottenere la data corrente, nel formato YYYY-MM-DD hh:mm:ss si usa il seguente codice.

```php
function get_time()
{
  date_default_timezone_set('Europe/Rome');
  return date("Y-m-d H:i:s");
}
```

3.2.8.3 Controllo accesso diretto a pagine php

Per garantire un corretto utilizzo dell'applicazione web, è importante impedire l'accesso diretto ad alcune pagine php che solo in determinati vengono eseguite.
Per effettuare tale controllo si verifica se è stata registrata **$_SESSION['logon']**.
Il controllo viene effettuato con il seguente codice:

```
if (!session_id())
  session_start();
if (!$_SESSION['logon'])
redirect_basic_or_full_error("Accesso Negato. Non puoi accedere
   direttamente a questa pagina.");
```

3.2.8.4 Smart Redirect Front-End

La precedente funzione in caso di Forbidden-Access esegue un redirect nella home page, mostrando il relativo errore.
Con la funzione **redirect_basic_or_full** si verifica quale è il front end attualmente in uso e si effettua il redirect in quest'ultimo.
La funzione **redirect_basic_or_full_error** è la medesima funzione, con l'aggiunta che si accoda al link del redirect un parametro che in questo caso rappresenta l'errore.

3.2.9 login.php

Questo script consente di eseguire l'accesso all'area riservata all'amministrazione dell'applicazione web.
Riceve come parametri con metodo GET l'**username** e la **password**, esegue un confronto e in caso affermativo registra la sessione **$_SESSION['logon']** e fa un redirect sulla pagina dell'amministrazione.

3.2.10 logout.php

Questo script consente di cancellare la sessione registrata, e quindi l'autenticazione per poter accedere all'area riservata.
Per fare ciò si distruggono tutte le variabili di sessione, si cancella il session cookie e infine si distrugge la sessione.
Il codice è il seguente:

```
<?php
session_start();
$_SESSION = array();
if (ini_get("session.use_cookies")) {
$params = session_get_cookie_params();
setcookie(session_name(), '', time() - 42000,
$params["path"], $params["domain"],
$params["secure"], $params["httponly"]
```

```
9  );
10 }
11 session_destroy();
12 header("Location: ../index.php");
13 ?>
```

3.2.11 report.php

Questo script php consente di inserire nella tabella **advice** una segnalazione da parte di un'utente.
Si riceve come parametro GET la **segnalazione**, e qualora l'utente avesse acconsentito alla richiesta di geolocalizzazione anche i parametri riguardanti la posizione dell'utente, l'accuratezza e l'altitudine. Qualora l'utente non avesse acconsentito alla localizzazione, questi parametri saranno settati a 0.
La query viene costruita come segue:

```
1 $sql = "INSERT INTO advice value(NULL,'".mysqli_escape_string(
    $dbhandle,$text)."','".mysqli_escape_string($dbhandle,user_agent
    ()).".','".get_time().".','".get_client_ip().".','".$user_pos_time."
    ','".$user_pos_accuracy."','".$user_pos_altitude."','".$user_pos.
    "');";
```

3.2.12 upload.php

Questo script consente di caricare un file nella directory *images*. Può essere richiamato solo tramite il pannello dell'amministrazione, quindi bisogna prima essere autenticati.
Vengono effettuati vari controlli, prima di procedere al caricamento effettivo:

- Si verifica che il file sia effettivamente un'immagine utilizzando un controllo sul valore di ritorno della funzione **getimagesize**;
- Si verifica che non ci sia un duplicato del file utilizzando la funzione **file_exists**;
- Si verifica se l'immagine sia superiore a una dimensione prefissata. Si è scelto di mettere come limite 5MB;
- Con la funzione **strtolower(pathinfo($target_file,PATHINFO_EXTENSION))** si ottiene l'estensione dell'immagine. Si verifica quindi se l'immagine ha una tra le seguente estensioni: *jpg*, *png*, *jpeg* o *gif*.

Dopo aver caricato con successo il file, che verrà spostato dalla cartella *temp* del web server alla cartella *images* con la funzione **move_uploaded_file** , si inseriranno nel database le informazioni necessarie per tenere traccia del file e poterlo

usare successivamente.
Le informazioni vengono passate con il metodo GET e sono la descrizione dell'immagine e il nome del file. Infine chiamando la funzione **get_time()** si ottiene la data corrente.
La query viene costruita come segue:

```
1 $sql = "INSERT INTO photo VALUES(NULL,'$file_name','$descr','CPanel
    ', '$time')";
```

3.2.13 upload_canvas.php

Questo script come il precedente effettua il caricamento di un canvas nella cartella *images*. È utilizzato dal front end basic. La query viene costruita come segue:

```
1 $sql = "INSERT INTO photo VALUES(NULL,'$file_name','$descr','user
    basic', '$time')";
```

3.2.14 upload.php full front-end

Questo script, localizzato nella directory *full/php/* è analogo ai precedenti: effettua il caricamento di un file multimediale nella cartella *images*. La query viene costruita come segue:

```
1 $sql = "INSERT INTO photo VALUES(NULL,'$file_name','$descr','user
    full', '$time')";
```

3.2.15 images.php

Questo script producendo dinamicamente codice html, consente di visualizzare tutte i contenuti multimediali situati nella cartella *images*.
Il funzionamento è il seguente:

- Si iterano tutti i file presenti nella cartella tramite il construtto **foreach (new DirectoryIterator("../images") as $file)**
- Si verifica se effettivamente è un file con la funzione **$file->isFile()**;
- Si verifica se il file ha un'estensione di un'immagine valida con l'istruzione **(strpos($file->getFilename(), '.estensione')**, vengono fatti controlli su più estensioni: *jpg*, *png*, *jpeg* o *gif*;
- Se è un'immagine valida si ricava il nome del file con l'istruzione **$file->getFilename()**;

- tramite una semplice select si estrapola dal database il record dell'immagine. Dato che la colonna path_photo è unique avremo un solo record di ritorno. Con tale record si otterranno informazioni quali la descrizione, la data di creazione e da chi è stata creata.
  Nel caso in cui non fosse registrata alcuna referenza con il database (quindi l'immagine è stata semplicemente copiata nella directory) ciò verrà fatto presente nella pagina web.

3.2.16 init.php

Questo script inizializza il front end aggiungendo i dati sulla relativa mappa. I dati che aggiunge sono essenzialmente i beacon e i place, che vengono aggiunti alla mappa Leaflet stampando dinamicamente codice javascript attraverso la funzione **echo**.
Viene eseguito ad ongi avvio o refresh dell'index dell'applicazione web.
Questo script può essere suddiviso logicamente in due sotto scripts per facilitarne la comprensione. Una parte per l'inizializzazione dei beacon e una parte per l'inizializzazione dei place.

3.2.16.1 Beacon Init

I passaggi che vengono svolti per inizializzare la mappa inserendo tutti i beacon sono i seguenti:

- Si estrapolano dal database tutti i beacon effettuando una **select ***, dopodiché si effettua la query utilizzando la funzione **mysqli_query** e infine si contano tutti i beacon ottenuti tramite la funzione **mysqli_num_rows**. Per ogni beacon si entra in un ciclo volto a ricavare dati supplementari;

- Si ricavano tutte le informazioni del beacon, quali la latitudine, la longitudine, e la descrizione (colonna *information*);

- Si ricava il nome della pianta associata al beacon (se è associato) e la relativa foto della pianta.
  Per fare ciò vengono utilizzate le seguenti query:

```
1 $sql_plant_name = "select name_plant from plant,exists_table
    WHERE plant.id_plant = exists_table.id_plant_fk AND
    exists_table.id_beacon_fk = '".$id_beacon."' ;";
2
3 $sql_beacon_plant_image = "select id_photo_plant_fk from
    exists_table where id_beacon_fk = '".$id_beacon."';";
4
5 $sql_photo_plant_beacon = "select path_photo from photo where
    id_photo = '".$result["id_photo_plant_fk"]."';";
```

La variabile **id_beacon** è dinamica.
La query **sql_beacon_plant_image** restituisce l'id univoco dell'immagine della pianta dove è stato posizionato il beacon e con la query **sql_photo_plant_beacon** si ottiene infine il path di tale immagine.

- Si verifica a questo punto se il beacon ha un'immagine custom controllando il valore della colonna **id_photo_beacon_fk**. In caso affermativo, si recupera l'id dell'immagine e successivamente il path.
  Si definirà allora in PHP una variabile **myIcon** come segue:

```
1 echo "var myIcon = L.icon({iconUrl: '../images/$path',iconSize
    : [45, 95]";
```

- Dopodichè si adotta lo stesso procedimento per l'immagine ombra custom del beacon controllando il valore della colonna **id_photo_beacon_shadow_fk** e se settata, si recupera con lo stesso metodo precedentemente introdotto.
  Verrà allora accodata alla variabile myIcon l'url della shadow custom:

```
1 echo ", shadowUrl: '$path_shadow',
2     shadowSize: [68, 95],
3     shadowAnchor: [22, 94]";
4
```

- Infine il beacon verrà aggiunto dinamicamente come segue:

```
1 echo "}); ";
2 echo "var marker".$i." = L.marker([$lat,$lng], {icon: myIcon}).
    addTo(map);".
```

Listing 3.10: Aggiunta dinamica di un beacon con la customizzazione del parametro icon.

```
1 echo "var marker".$i." = L.marker([$lat,$lng]).addTo(map); ";
```

Listing 3.11: Aggiunta dinamica di un beacon con immagini di default. (Blu)

- Al beacon appena aggiunto, si associa un evento al **click**. (Verrà analizzato nella parte di front-end)

```
1 echo "  marker".$i.".on('click',
2 function (event) {
3     create_popup('$plant_name',
4             '$beacon_info','$beacon_plant_image',
5             '$lat','$lng')
6         }
7  ); ";
```

- Infine il Beacon viene aggiunto a un layer, per essere gestito successivamente.

```
1 echo "markers.addLayer(marker".$i.");";
```

3.2.16.2 Place Init

I passaggi che vengono svolti per inizializzare la mappa inserendo tutti i place sono simili a quelli sopra descritti pe r inserire i beacon. Nel dettaglio sono i seguenti :

- Si estrapolano dal database tutti i place effettuando una **select ***, dopodiché si effettua la query utilizzando la funzione **mysqli_query** e infine si contano tutti i place ottenuti tramite la funzione **mysqli_num_rows**. Per ogni place si entra in un ciclo volto a ricavare dati supplementari;

- Si ricavano le informazioni del place, quali il nome, l'id, la latitudine, la longitudine, la descrizione e l'insieme di coordinate che permettono di disegnarlo (*boundary*);

- Si aggiunge il place alla mappa con il seguente codice:

```
echo "var ggeojson".$i." = ".$boundary."; ";
echo "var pol".$i." = L.geoJSON(ggeojson".$i.", {onEachFeature:
    onEachFeature}).addTo(map).bindPopup('$place_descr'); ";
```

- Il place viene aggiunto a un layer per essere gestito successivamente:

```
echo "polygon.addLayer(pol".$i."); ";
```

- Si aggiunge il punto noto del place sulla mappa, ossia il punto ottenuto dalle colonne latitudine e longitutine:

```
echo "var circle".$i." = L.circle([".$lat.",".$lng."], {
color: 'red',
fillColor: '#f03',
fillOpacity: 0.5,
radius: 20
}).addTo(map).bindPopup('$name_place');";
```

- Infine anche il punto noto del place viene aggiunto al layer per essere gestito successivamente.

```
echo "polygon.addLayer(circle".$i."); ";
```

Si specifica che il place (figura geometrica rappresentante l'orto botanico sulla mappa) e il punto noto del place (circle rosso che solitamente è situato al centro del place) essendo stati aggiunti allo stesso layer sono gestiti come un'unica entità.

3.2.17 search.php

Questo script effettua la ricerca dei contenuti memorizzati nel database secondo dei filtri predefiniti.
Se la ricerca ha esito positivo, la mappa viene inizializzata con i contenuti corrispondenti alla ricerca. I filtri di ricerca sono 3 :

- Filtro per un dato Orto Botanico (*place*) : viene mostrata sulla mappa la geometria del place e tutti i beacon (piante) appartenenti ad esso.
- Filtro per una data Specie (*species*) : vengono mostrati tutti i beacon (piante) di una data specie tenendo conto di tutta la mappa. (*filtro limitrofo*)
- Filtro per una data Fase Fenlogica (*phenological_phase*) : vengono mostrati tutti i beacon (piante) che sono in una data fase fenologica confrontando la data di tale fase fenologica con la data odierna.

Vi è la possibilità di selezionare più filtri insieme e in tal caso verrà applicata l'operatore **AND** a tali filtri.
Lo script segue la seguente logica:

1. Si controlla se è stato selezionato il filtro **place**. In caso negativo si passa la punto 2.
   In caso positivo si stampa la geometria del place (fare riferimento a **place init**). Dopo aver stampato il place verifico se sono stati immessi altri filtri di ricerca:
   - Solo **Place** : stampo tutti i beacon (piante) appartenenti a tale orto botanico (place). (fare riferimento a **beacon init**)
   - **Place** + **Species** : mostro i beacon del dato place con una certa specie.
   - **Place** + **Phase** : mostro i beacon del dato place con una certa fase fenologica.
   - **Place** + **Phase** + **Species** : mostro i beacon del dato place con una data fase fenologica e una data specie.
2. Se non ho messo come filtro di ricerca place, verifico se è stato messo come filtro di ricerca la fase fenologica e non la specie.
   In caso negativo si passa al punto 3.
   In caso positivo mostro tutte le piante (beacon) dove la fase è quella selezionata e $start\_day < current\_day < end\_day$ *AND*
   $start\_month < current\_month < end\_month$
3. Verifico se è stato messo come filtro di ricerca la specie e non la fase fenologica. In caso negativo si passa al punto 4.
   In caso positivo mostro tutte le piante (beacon) di una data specie.

4. L'ultimo caso prevede che siano stati selezionati il filtro specie e il filtro fase fenologica. Quindi si applica l'operatore AND al punto 2 e 3, stampando tutte le piante con una data specie e una data fase fenologica scelta dove la data odierna sia compresa tra l'inizio e la fine di tale fase.

3.2.18 fill.php

Questo script riempe dinamicamente, a seconda dei dati nel database gli elementi che ci permettono di selezionare i filtri di ricerca.
Vi sono due file *fill.php* uno nella directory *full/php* e un altro nella directory *basic/php*. L'esigenza di fare due file per riempire tali elementi, nasce dal fatto che gli elementi presi in considerazione sono di tipo diverso e di conseguenza richiedono costrutti diversi per essere manipolati dinamicamente.
L'idea è quella di fillare dinamicamente gli elementi contenitori dei filtri per place,specie e fase fenologica. Gli elementi che conterranno i filtri saranno così logicamente strutturati:

- L'elemento del filtro place conterrà tutti i nome degli orti botanici (si ricorda che la colonna name_place è **unique**)
- L'elemento del filtro per specie conterrà tutte le specie registrate.
- L'elemento del filtro per fase fenologica conterrà tutte le fasi fenologiche ottenute con una **select distinct *** per evitare duplicati.

Ad ogni elemento che contiene un dato filtro, oltre alla label del filtro selezionato, avrà il suo id per una più facile implementazione della ricerca.

3.2.18.1 basic fill.php

Nel front end basic, il contenitore scelto per ospitare i filtri di ricerca è il tag html **select**. Nel caso in cui non si abbiano filtri di ricerca disponibili (in assenza di dati nel database), il contenitore sarà nascosto. Per procedere al filling dinamico di tale contenitore si impiega il seguente codice PHP:

```
1  $sql_places = "select * from place";
2  $query_get_places = mysqli_query($dbhandle,$sql_places);
3  $count_places = mysqli_num_rows($query_get_places);
4
5  if($count_places> 0)
6  {
7    for($i=0;$i<$count_places;$i++)
8    {
9     $row_place = mysqli_fetch_array($query_get_places, MYSQLI_ASSOC);
10    $name_place = $row_place["name_place"];
11    $id_place = $row_place["id_place"];
12    echo "var place_select =
```

```
13      document.getElementById('place-select');";
14     echo "place_select.options[place_select.options.length] =
15       new Option('$name_place', '$id_place'); ";
16
17   }
18 }
19 else
20   echo "document.getElementById('div_places').style.display =
21     \"none\"; ";
```

Listing 3.12: Filling Dinamico per i place - select element

Il meccanismo utilizzato si applica in modo uguale anche per il filling dinamico delle specie e delle fasi fenologiche.

3.2.18.2 full fill.php

In questo front end il contenitore scelto per ospitare i filtri è di tipo **li** (*list-item*). L'approccio utilizzato è identico, di seguito il codice per il filling dinamico di tali contenitori:

```
1 echo "var place_select = document.getElementById('place_select');";
2 echo "var li = document.createElement('li');";
3 echo "li.innerHTML = \"<a href='#' id = \'$id_place\'>
4     $name_place</a>\";";
5 echo "place_select.appendChild(li);";
```

Listing 3.13: Filling Dinamico per i place - li element

Anche qui viene utilizzato lo stesso meccanismo per il filling dinamico delle specie e delle fasi fenologiche.

3.2.19 manage.php

Come descritto nella prefazione, questo script php si occupa di gestire il database logic senza l'ausilio di software esterni o pannelli di controllo del web server, quali **PhpMyAdmin**, **MySQLWorkbench**, **Navicat** ecc..
Genera un documento html ausiliato da da codice javascript per gestire direttamente dall'applicazione i dati sul database.
Questa pagina può essere suddivisa in più sotto sezione per una migliore comprensione. Dato che genera un front-end verrà analizzata nelle prossime sezioni.

3.2.20 fill_manage_map.php

Questo script si occupa di inizializzare la mappa che è generata dinamicamente dallo script manage.php. Il funzionamento è analogo a quello precedentemente introdotto con lo *init.php* a differenza che l'evento associato al click dei beacon (piante) richiamava una funzione (*create_popup*, che verra analizzata successivamente) mentre ora si chiama la funzione **alert**.

```
echo "  marker".$i.".on('click', function (event) { alert('
    Informazioni beacon :$beacon_info\nLatitudine: $lat\nLongitudine:
    $lng')}); ";
```

3.2.21 main index.php

Questo script è il primo ad essere eseguito appena si digita l'indirizzo del web server, questo perchè come suggerisce il nome, è la prima pagina e la più importante, è da qui si accede a tutte le altre pagine del sito. Perchè è il primo script ad essere eseguito? Perchè quando un web server riceve una richiesta HTTP riguardo ad una determinata directory cerca di rispondere alla richiesta inviando la pagina sua pagina index, che in questo caso è **main index.php**.
Questa pagina permette all'utente di scegliere dinamicamente il front end desiderato tramite due ancore, inoltre registra i dati dell'utente come segue:

```
$sql = "INSERT INTO visitors value(NULL,'".get_client_ip()."',
    '".mysqli_escape_string($dbhandle,user_agent()."',
    '".get_time()."');";
```

Listing 3.14: Query per l'inserimento dei dati dell'utente nella tabella visitors facendo ausilio delle funzioni definite nello script function.php

3.2.22 front end index.php

All'iterno delle directory di ciascun front end vi è un index.php, questo script si occupa di generare dinamicamente la pagina html e verifica se gli sono stati passati dei parametri con metodo GET. La logica back end di questi due script è la stessa ed è la seguente:

1. si inizializza la sessione e si fa una set della variabile di sessione per il framework corrente e una unset della variabile di sessione dell'altro framework. Riassumendo si setta una variabile per riconoscere quale front end dovrà essere utilizzato per futuri redirect.
   Il codice utilizzato è il seguente:

   ```
   <?php
           if (!session_id())
           session_start();
           $_SESSION['full'] = true;
           unset($_SESSION['basic']);
   ?>
   ```

   Listing 3.15: Inizializzazione della sessione avendo scelto il front end full

2. il list item contenente un'ancora per accedere alla pagina *manage.php*, si verifica tramite codice php se è registrata la variabile di sessione *logon*, in caso negativo l'ancora rimanderà l'utente alla form di login mentre in caso

positivo si effettuerà un redirect alla pagina *manage.php*;
Il codice utilizzato è il seguente:

```
if (!session_id()) session_start();
  if (isset($_SESSION["logon"]))
    echo "<li><a id='login_db' href=\"../php/manage.php\">
    <i class=\"material-icons\">build</i>
    Gestisci il Database</a></li>";
  else
    echo "<li><a id='login_db' href=\"#login\"  class=\"modal-
    trigger\">
    <i class=\"material-icons\">build</i>
    Gestisci il Database</a></li>";
```

3. si verifica se sono stati passati dei filtri di ricerca alla pagina. In caso negativo si richiama lo script *init.php* per inizializzare la mappa normalmente. In caso positivo vengono registrati in una sessione tali filtri e si richiama lo script *search.php*; Il codice utilizzato è il seguente:

```
if( isset($_GET["place"]) ||
    isset($_GET["species"]) ||
    isset($_GET["phase"]))
{
  if(isset($_GET["place"]))
    $_SESSION["place"] = $_GET["place"];
  if(isset($_GET["species"]))
    $_SESSION["species"] = $_GET["species"];
  if(isset($_GET["phase"]))
    $_SESSION["phase"] = $_GET["phase"];

  include '../php/search.php';
}
else
  include '../php/init.php';
```

4. si verificano se si sono verificati degli errori prima di aprire questa pagina, in caso positivo viene stampato a video tramite la funzione **alert**. L'errore viene passato come parametro con metodo GET; Il codice utilizzato è il seguente:

```
if(isset($_GET["error"]))
{
  $error = $_GET["error"];
  echo "<script> alert(\"$error\"); </script>";
}
```

5. infine si esegue lo script *fill.php*;

3.3 Front-End

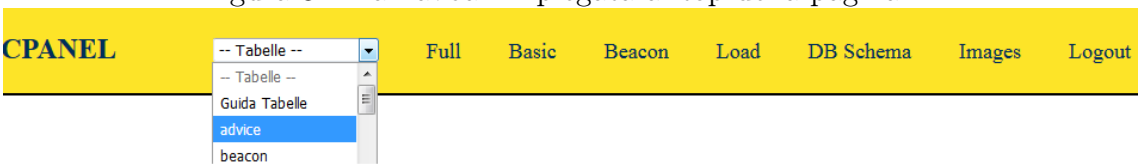
Ora verranno analizzati i file *.js* e il file *manage.php* nonché tutti i construtti utilizzati degni di nota realizzati tramite linguaggi di front-end. Si specifica che dato l'utilizzo a fini amministrativi di questa pagina, è predisposta solo la navigazione tramite desktop. (la navigazione da mobile è comunque consentita ma con stili non perfettamenti idonei a tale risoluzione)

3.3.1 manage.php

Per comprendere meglio come viene generata dinamicamente la pagina da questo script, si suddivide in più parti.

3.3.1.1 Navbar

Figura 3.1: la navbar impiegata al top della pagina



Per garantire una migliore esperienza di navigazione, soprattutto nella parte di amministrazione del sito; è stata inserita al top della pagina e resa fissa anche allo scroll (proprietà css **position: fixed**) una barra di navigazione (navbar). Come è possibile vedere dalla figura 3.1 la navbar è composta da vari elementi:

- **menù a tendina** contenente le tabelle, che è riempito dinamicamente a seconda del database selezionato;
- **Full** : àncora che fa riferimento all'applicazione web con front-end Full (**href='full/indexp.php'**);
- **Basic** : ancora che fa riferimento all'applicazione web con front-end Basic (**href='basic/indexp.php'**);
- **Logout** : àncora che fa riferimento allo script **logout.php** che disconnette l'utente cancellandone la sessione;
- **Images** : àncora che fa riferimento allo script **images.php** che consente di monitorare le immagini presenti nel nostro web server alla cartella images;
- **Beacon**, **Load**, **DB Schema** : al loro click si effettua uno scroll per mostrare a schermo la sezione del documento desiderata;

3.3.1.2 Scrolling delle sezioni

Cliccando su un'opzione o sulle ancore Beacon, Load o DB Schema viene eseguita una funzione che ci permette di scrollare la pagina a tale sezione.
Per fare ciò a ogni sezione, a un elemento scelto (solitamente il primo ben in vista) gli viene assegnato al suo attributo **id** un identificativo univoco preceduto dal simbolo #, che è lo stesso identificativo assegnato alla ancora corrispondente, in modo da essere referenziato in caso di click. Quindi a un evento click verrà eseguita la funzione **scroll_function(this.value)**.

```
function scroll_function(string)
{
  var id = "#"+string;
  var element = document.getElementById(id);
  element.scrollIntoView();
  window.scrollBy(0,-100)
}
```

È da notare che alla funzione **window.scroll** come secondo parametro viene passato **-100**. Questo perchè la navbar ha come proprietà css **margin-top: 100px** e questo margine top viene aggiunto in modo da prevenire il **content overlay**.

3.3.1.3 Sezione inserimento beacon

Cliccando sulla ancora **Beacon** della navbar (o in alternativa scorrendo manualmente la pagina fino a tale sezione) si visualizzerà questa sezione dedicata all'inserimento del beacon sulla mappa.
Il funzionamento è facile ed intuitivo, ed è schematizzabile seguendo i seguenti passagi:

1. è presente la mappa di leaflet e grazie allo script **fill_manage_map.php** anche in questa mappa sono visibili le geometrie dei place e i beacon;

2. dopo aver scelto la posizione del nuovo beacon sulla mappa è sufficiente eseguire un **double click** per aggiungerlo;

3. verrà eseguito un evento che tramite dei **confirm** chiederà in input da parte dell'utente le altre informazioni inerenti al beacon, quali la descrizione (colonna *information*), l'**id** dell'immagine custom e dell'immagine ombra custom. L'id delle due immagini possono anche essere **null**, così facendo verranno utilizzate le immagini di default;

4. se le informazioni sono state immesse correttamente e non si è scelto alcun cancel ai vari prompt mostrati, si procede all'esecuzione della query;

5. infine se la query ha dato esito negativo verrà mostrato l'errore, altrimenti si effettuerà uno scroll in un'altra sezione della pagina mostrando il beacon appena aggiunto, in una tabella;

Di seguito il codice utilizzato per realizzare quanto detto sopra.

```
map.on('dblclick', function(e){
var lat = e.latlng.lat;
var lng = e.latlng.lng;
var information = prompt('Inserisci le informazioni di questo beacon.');
var icon = prompt('Inserisci l\'ID dell\'icona.');
var shadow_icon = prompt('Inserisci l\'ID della shadow icon.');

var values = 'NULL,';
if(information==null || information ==='')
  values += 'NULL,';
else
  values += '\''+information +'\',';
  values += '\'' + lat + '\',\'' + lng + '\'';
if(icon ==null || icon ==='')
  values += ',NULL';
else
  values += ',\'' + icon  + '\'';
if(shadow_icon ==null || shadow_icon ==='')
  values += ',NULL';
else
  values += ',\'' + shadow_icon  + '\'';
document.location =   'add.php?table=beacon&values='+values;
```

3.3.1.4 Sezione Caricamento Immagini

Cliccando sull'ancora **Load** della navbar si visualizzerà la sezione della pagina dedicata al caricamento di contenuti multimediali nella cartella **images**.

Per caricare un'immagine si eeguono i semplici passaggi:

1. si clicca sul bottone **Browse** per selezionare il file dal dispositivo. Questo elemento è un input tag con attributo **type=file** definito dall'HTML5;

2. si lascia una descrizione dell'immagine che si vuole caricare nella casella di testo corrispondente;

3. infine per procedere alla richiesta caricamento effettivo basta cliccare il botton **Carica File**;

4. al click del bottone si verifica che gli input file e text siano stati riempiti correttamente. In caso negativo viene mostrato un messaggio d'errore, altrimenti si esegue lo script *upload.php* che effettua tutte le verifiche del caso per poi procedere al caricamento;

3.3.1.5 Database data management

È possibile accedere a questa sezione o scrollando la pagina fino al suo inizio o selezionando una tabella del database dal menù a tendina della navbar. Per ogni tabella del database si effettuano i medesimi passaggi. La logica utilizzata è la seguente:

1. si ricava la lista delle tabelle del database selezionato (il database viene settato nel file *config.php*) utilizzando l'istruzione SQL **show tables**;

2. effettuata la query, tramite la funzione **mysqli_num_rows** si entra in un ciclo che itera quante volte quante tabelle si hanno;

3. si aggiunge dinamicamente il nome della tabella al menù a tendina;

4. si genera un tag **h1** contenente il nome di tale tabella. Oltre alla label associata, si associa al click l'evento **rename_table** e inoltre gli viene assegnato come id, la sua label preceduta dal # per essere referenziata per lo scrolling;

5. viene generata un tag **table** che conterrà i dati di tale tabella;

6. tramite il comando SQL **show columns from nome_tabella** si ricavano tutte le colonne della tabella iterata;

7. si crea la prima riga della **table** aprendo il tag **tr**;

8. effettuata la query, tramite la funzione **mysqli_num_rows** si entra in un altro ciclo che itera quante volte quante colonne si hanno;

9. per ogni colonna si ottengono le seguenti informazioni dall'array associativo:

   - **Field** : Il nome della colonna;
   - **Key** : Se è un tipo di chiave particolare (primary,fk,unique,ecc..);
   - **Type** : Il tipo della colonna;
   - **Null** : Se la colonna può essere null;

10. Nell'iterazione del ciclo verranno create tante colonne (**th**) alla **table**, quante colonne ha la tabella. Quindi si apre il tag **th** e il tag **a**;

11. Ogni colonna (tag **th**) ha al suo interno un'ancora (tag **a**). Vengono associati due eventi al click, uno per ciascun tag:

    - **th** : Si chiama la funzione **change_column**;
    - **a** : Si chiama la funzione **rename_column**;

    È importante dire che al trigger dell'evento click dell'elmento più interno successivamente verrà eseguito il trigger del click anche al contenitore (**bubbling**). Per prevenire ciò al click dell'elemento più interno al contenitore (in questo caso l'ancora) viene eseguita la funzione **event.stopPropagation()** per prevenire il bubbling.

12. Si verifica se null ha valore 'no'. In caso segativo si passa la punto successivo, altrimenti si apre il tag **u** (underline);

13. Si verifica se la chiave è un tipo particolare. In caso positivo si apre il tag **font** specificando una data classe definita nei fogli di stile. Dopodiché si stampa il nome della colonna e si chiude il tag. Possono verificarsi i seguenti casi:

    - Foreign Key : il tag font avrà class='fk';

    ```
    color : rgb(76, 160, 199);
    ```

    - Primary Key : il tag font avrà class='pk';

    ```
    color : red;
    ```

    - Unique Key : il tag font avrà class='uq';

```
color : rgb(145, 120, 204);
```

- La chiave non è un tipo particolare: quindi non si utilizzerà nessun font tag.

14. Si verifica nuovamente se null ha valore 'no', in modo da chiudere il tag se è stato precedentemente aperto;

15. Vengono chiusi ordinatamente prima il tag dell'ancora e poi il tag dell'header della colonna;

16. Si costruisce la query **select * from nome_tabella_iterata** per ottenere tutti i record presenti in tale tabela;

17. Viene eseguita la query e con la funzione **mysqli_num_rows** si contano i record della tabella;

18. Si entra in un ciclo che itera quante volte quanti record ci sono;

19. Per ogni record si ottiene sia il nome che il valore della chiave primaria. Per i valori delle chiavi primarie si utilizza la query:

```
$primary_key_sql = "SHOW KEYS FROM ".$nome_tabella." WHERE
    Key_name = 'PRIMARY'";
```

20. Per ogni record si entra in un altro ciclo che itera quante volte quante colonne ha la tabella del record;

21. Per ogni colonna del record si apre il tag **td**, si associa al click l'evento **update**, si apre il tag **a** si stampa il valore del record della data colonna e si chiudono i tag aperti;

22. Se la tabella si chiama **visitors** alla cella della tabella non viene associato alcun evento al click;

23. Dopo aver stampato tutti i valori delle colonne del record viene inserita un'ulteriore colonna volta a eliminare il record. Dentro questa cella si trova un'immagine la quale è associato l'evento **delete_row**;

24. Finiti i record da inserire nella table si costruisce un'ultima riga per permettere l'inserimento manuale da parte dell'utente. (Se il nome della tabella è visitors si passa al punto successivo; viene eseguita l'istruzione **continue**) Per fare ciò si fanno delle verifiche sulle colonne:

    - se è una chiave primaria **auto increment** non sarà presente alcuna textbox in quanto è al livello sql che viene assegnata la chiave primaria;

- se la colonna è di tipo date, nella cella verrà stampato un input di tipo **date**.
- se la colonna è di tipo enumerazione, nella cella verrà stampato un menù a tendina riempito con le possibili alternative (**select**);
- se è una chiave esterna verrà creato un menù a tendina riempito con i possibili valori. Per fare ciò viene utilizzata la query introdotta nella sezione 3.4.1.1;
- si verifica infine se la colonna può essere `null` o meno. In caso positivo positivo viene stampata una semplice textbox (che può anche essere lasciata vuota) mentre in caso negativo gli viene associato un id che permetterà di essere gestito da degli eventi;

Finite le colonne, se ne aggiunge una ulteriore contenente un'immagine che ha associato l'evento **add**, volto ad aggiungere il nuovo record.

Figura 3.2: Tabella exists_table generata dinamicamente dalla logica appena introdotta.

exists_table

id_exists	id_place_fk	id_plant_fk	id_beacon_fk	id_photo_plant_fk	
1	1	1	1	14	
2	2	2	2	15	
3	1	5	3	12	
4	3	6	4	13	
5	2	3	5	16	
6	3	4	6	17	
7	1	7	7	18	
8	3	8	8	19	
9	2	9	9	18	
AI	1	1	1	2	

3.3.2 Funzioni javascript impiegate nella sezione 3.3.1.5

3.3.2.1 add

Questa funzione permette di aggiungere un nuovo record al database. Gli vengono passati due parametri:

- il nome della tabella;
- la riga della tabella dove sono stati inseriti i valori (**tr**);

Come è possibile vedere dal codice sopra, questa riga viene passata tramite l'istruzione **this.parentElement.parentElement** eseguita nel tag **img**; il parentElement di img è td e parentElement di td è tr.
La funzione estrapola i valori dal tag tr e li concatena con la seguente sintassi : *valore1,valore2,valore3..*, nel caso la chiave primaria fosse auto incrementante la sintassi sarà la seguente: *NULL,valore2,valore3...* Ogni valore viene controllato per prevenire eventuali sequenze di escape volte a un **sql injection**.

```
1 if(element.cells[i].children[0].value.indexOf("'") > -1 || element.
    cells[i].children[0].value.indexOf("\"") > -1)
2 values = values.concat("'"+element.cells[i].children[0].value.
    replace(/[\\"']/g, '\\$&')+"',");
```

Listing 3.16: Codice di escape

Concatenati correttamente tutti i valori, dopo aver chiesto conferma all'utente, se confermato si procede al redirect come segue, invocando lo script **add.php**.

```
1 document.location =   "add.php?table="+table+"&values="+values;
```

3.3.2.2 delete_row

Questa funzione consente di eliminare il record selezionato nella tabella, dal database. Vengono forniti come parametri di ingresso:

- il nome della tabella dal quale si vuole cancellare il record;
- l'ancora che ha invocato questa funzione;
- il nome della chiave primaria;
- il valore della chiave primaria.

Dopo aver chiesto conferma dell'operazione, se confermata si procede con la seguente istruzione che permetterà il redirect alla pagina **delete.php**.

```
1 element.href =   "delete.php?table="+table+"&pk="+pk+"&pk_value="+
    pk_value;
```

3.3.2.3 update

Questa funzione consente, cliccando sulla cella che si vuole modificare, di aggiornare tale valore nel database. Riceve i seguenti parametri di ingresso:

- il nome della tabella dove si trova il record da aggiornare;
- il nome della colonna che si vorebbe aggiornare;
- il valore che si vuole modificare;
- il nome della chiave primaria;
- il valore della chiave primaria.

La funzione procede come segue:

1. attraverso la funzione **prompt** viene chiesto il nuovo valore;
2. si verifica che il valore sia diverso da null e che ci sia stato inserito almeno un carattere, in caso negativo si esce dallo script;
3. si verifica che il valore inserito non coincida con il valore già presente nel database, in tal caso si esce dallo script mostrando la motivazione dell'errore;
4. si chiede conferma della modifica;
5. si effettue il redirect invocando lo script; **update.php** con la seguente istruzione:

```
1 document.location =   "update.php?table="+table+"&new_value="+
    update+"&column="+column+"&pk="+pk+"&pk_value="+pk_value ;
```

3.3.2.4 rename_column

Questa funzione permette di rinominare una colonna di una tabella del database. Riceve i seguenti parametri di ingresso:

- il nome della tabella dove si trova la colonna da rinominare;
- l'ancora che ha invocato questa funzione;
- il nome attuale della colonna.

La funzione procede in modo analogo come la precedente. Infine effettua il redirect invocando lo script **rename_c.php** come segue:

```
1 element.href =   "rename_c.php?column="+text+"&new_value="+update+"&
    table="+table ;
```

3.3.2.5 rename_table

Questa funzione permette di rinominare una tabella. Prende come parametro in ingresso solamente il nome attuale della tabella, che sarà confrontato con il valore inserito. Dopo aver verificato anche che il valore inserito non sia null e abbia almeno un carattere, si effettua il redirect invocando lo script **rename_table.php** come segue:

```
1 document.location =   "rename_t.php?table="+table+"&new_value="+
    update;
```

3.3.2.6 change_column

Questa funzione permette di cambiare il tipo di una colonna. Gli vengono passati il nome della tabella dove si trova la colonna da modificare e il nome della colonna. Dopo aver inserito il nuovo tipo, aver controllato che sia diverso da null e con un formato accettabile si chiede conferma di tale modifica. Dopo aver confermato si effettua il redirect invocando lo script **type.php** come segue:

```
1 document.location =   "type.php?column="+text+"&new_value="+update+"
    &table="+table ;
```

3.3.3 Le differenze sostanziali tra i due front-end

Le differenze sostanziali che intercorrono tra i due front sono schematizzabili come segue:

- Il FE basic è stato sviluppato in maniera *old style* ossia senza seguire lo stile moderno delle applicazioni web di ultima generazione, mentre il FE full adotta stili e design di ultima generazione;
- Il FE basic è stato stilizzato interamente con il **CSS**, senza alcun ausilio di librerie grafiche. Il FE full utilizza come framework **Materialize** che rispecchia il *Material Design*;
- Il FE basic utilizza Javasript puro, mentre il FE full include la libreria di JQuery e ne fa ampio uso. (Scelta vincolante in quanto materialize inizializza i suoi componenti esclusivamente con JQuery);
- Nel FE basic le form modali, i menù a tendina, le navbar e molti altri elementi vengono tutti gestiti manualmente tramite css e javascript. Nel FE full non ci si preoccupa di tale gestione dato che è Materialize che effettua questa parte;

3.3.4 basic front-end

Il *basic* front-end è il front end più basilare che utilizza puro codice css per essere stilizzato e puro javascript per i vari eventi. Verranno ora analizzate alcuni aspetti legati al javascript.

3.3.4.1 Menù humburgher

Per garantire un'esperienza piacevole e intuitiva, viene utilizzato un menù richiamabile appunto dalla figura humburgher. Per realizzare questo effetto al click

Figura 3.3: Funzionamento grafico del menù

dell'humburgher non si utilizza codice javascript, ma puro CSS. Il meccanismo si basa sul fatto che l'immagine raffigurante l'humburgher in realtà è una checkbox e la sidebar ha la proprietà transform: **translateX(-250px)**. Si gioca sulla proprietà **:checked** che al suo click permette di effettuare il translate per rendere visibile la sidebar. Per un effetto più accattivante viene settata una transizione di 250ms ease-in-out.

Il menù è composto dalle seguenti voci:

- Effettua una ricerca : apre la finestra modale della ricerca;
- Gestisci il database : apre la finestra modale del login;
- Scatta una foto : apre la finestra modale per scattare foto;
- Nascondi Beacons : nasconde i beacon dalla mappa. Al click questa voce prende la label 'Mostra beacons' che consente di effettuare l'operazione opposta;
- Nascondi Poligoni : nasconde i poligoni dalla mappa. Al click questa voce prende la label 'Mostra poligoni' che consente di effettuare l'operazione;
- Segnala un problema : apre la finestra modale per inviare i report;
- Aggiorna la pagina : effettua un refresh della pagina;
- Full website : effettua un redirect all'altro front-end. (full front-end).

3.3.4.2 Principio di funzionamento delle finestre modali

In questo format dell'applicazione web, si fanno ampio uso delle finestre modali, che vengono richiamate tramite il menù humburgher. Il principio di funzionamento è semplice: ogni form viene nascosta o mostrata tramite seguente istruzione javascript che modifica una proprietà css.

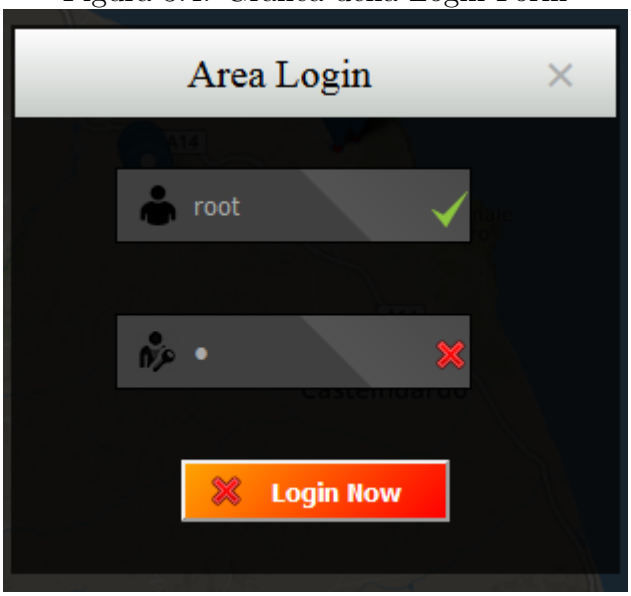
```
var modal = document.getElementById('myModal'); /*recupero la form
    dall'id*/
modal.style.display = "none"; /*nasconde la form*/
modal.style.display = "block"; /*mostra la form*/
```

Quando viene mostrata la form, viene aggiunto un overlay css con colore rgba(0,0,0,0.5). Per chiudere la finestra è sufficiente premere l'immagine di chiusura (una x in alto a destra) o cliccare ovunque fuori la form.

3.3.4.3 Finestra modale - Login

Questa finestra modale permette di effettuare il login e accedere alla pagina *manage.php*. Come è possibile vedere dall'immagine, troviamo la x per chiudere

Figura 3.4: Grafica della Login Form



tale form, due campi per inserire le credenziali e il bottono submit. Ecco alcune caratteristiche degne di nota delle caselle di testo (**input** tags):

- ottengono tale effetto grafico tramite un gradiente lineare;

```
 linear-gradient(45deg, #404040 0%, #404040 60%, #535353 60%,
#494949);
```

- Presentano un padding tale da inserire un'icona inerente al campo. Quest'effetto si raggiunge utilizzando la proprietà **background**;

```
background: url('../../images/icon.png') 10px center no-repeat,
    -webkit-linear-gradient(45deg, #404040 0%, #404040 60%,
    #535353 60%, #494949);
```

- All'evento keyup viene associata una funzione che controlla se il testo inserito supera dei determinati controlli. (In questo caso che la lunghezza minima sia di 4 caratteri). A seconda dell'esito dei controlli viene mostrata l'immagine corrispondente. Per inserire le immagini rispettivamente con padding sia a destra che a sinistra si utilizza sempre la proprietà background che viene formattata come segue;

```
background : url('../../images/icon_left.png') 10px center no-
    repeat, url('../../images/icon_right.png')  right 10px no-
    repeat , linear-gradient(45deg, #404040 0%, #404040 60%,
    #535353 60%, #494949);
```

- Ad ogni evento keyup si controlla se entrambi i campi sono stati fillati correttamente. In caso positivo si abilita il bottone submit che effettua il redirect allo script *login.php* procedere all'autenticazione.

3.3.4.4 Finestra modale - Segnala

Questa finestra modale permette di inviare una segnalazione e registrarla nel database. È composta da una **textarea** e un **submit**. Inoltre nella form vengono salvate le variabili che tengono traccia della posizione dell'utente, delll'accuratezza e dell'altitudine. (queste variabili vengono inizializzate dalla geolocation API se si è acconsentito alla geolocalizzazione, altirmenti sono null)
All'evento **onsubmit** si controlla che la segnalazione non sia vuota e si procede ad esegire lo script *report.php*.

3.3.4.5 Finestra modale - Scatta Foto

Questa finestra modale permette di scattare delle foto, registrarle nel database e caricarle nella cartella *images*. Si procede come segue:

1. si scatta la foto premendo il bottone *Scatta*;

2. il canvas viene riempito con l'immagine acquisita usando la funzione **canvas.getContext('2d').drawImage(...)**;

3. premendo il bottone *Carica* si verifica se il canvas non è vuoto (creandone uno nuovo e vuoto dinamicamente e confrontandolo);

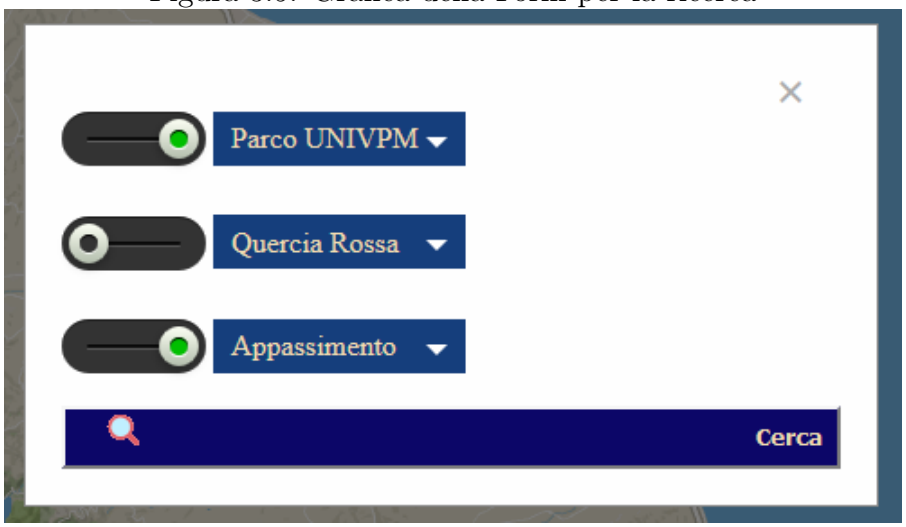
4. attraverso la funzione **prompt** viene chiesto all'utente di inserire la descrizione della foto appena scattata, si controlla che non sia null e che abbia almeno un carattere;

5. si inoltra una richiesta **Xml Http Request** per il caricamento (analizzata successivamente).

3.3.4.6 Finestra modale - Ricerca

Questa finestra modale permette di effettuare una ricerca tra elementi di interesse che abbiamo nella nostra mappa. I filtri desiderati si abilitano con appositi

Figura 3.5: Grafica della Form per la ricerca



checkbox stilizzati in stile slide e si selezionano tramite menù a tendina che come detto in precedenza vengono fillati dinamicamente dal proprio script del front end *fill.php*.
Cliccando il bottone *Cerca* si esegue la funzione **search()** che formatta l'url del redirect in base ai filtri abilitati e scelti. Si procede nel seguente modo:

1. si ottengono gli elementi delle checkbox;

2. si ottengono gli elementi dei menù a tendina;

3. si verifica se i menù a tendina non sono inizializzati, ciò vuoldire che non ci sono dati sufficienti per effettuare la ricerca e lo script si interrompe mostrando il relativo messaggio d'errore;

4. si verifica se almeno un filtro di ricerca è stato abilitato, altrimenti lo script si interrompe mostrando il relativo messaggio d'errore;

5. per ogni filtro abilitato e scelto si procede alla creazione dinamica dell'url utilizzando concatenazioni di stringhe attraverso l'operatore ternario;

Di seguito il codice utilizzato per realizzare i passaggi sopra descritti.

```
function search()
{
/*filtri abilitati*/
var place_check_var = document.getElementById('slide_place').checked
    ;
var species_check_var = document.getElementById('slide_species').
    checked;
var phase_check_var = document.getElementById('slide_phase').checked
    ;

/*filtri selezionati*/
var place_var = document.getElementById("place-select");
var species_var = document.getElementById("species-select");
var phase_var = document.getElementById("phase-select");


var name_place_selected_var = "";
var name_species_selected_var = "";
var name_phase_selected_var = "";

/*nel caso si forzasse l'apertura della modale utilizzando dei
    tricks*/
if((place_var.selectedIndex ==-1) && (species_var.selectedIndex
    ==-1) && (phase_var.selectedIndex==-1))
{
alert("Non ci sono dati nella mappa per effettuare una ricerca.");
return false;
}

/*se non sono stati selezionati filtri*/
if((place_check_var===false) && (species_check_var===false) && (
    phase_check_var===false))
{
alert("Seleziona almeno un filtro di ricerca.");
return false;
}

if((place_var.selectedIndex !=-1) && (place_check_var===true))
name_place_selected_var = place_var.options[place_var.selectedIndex
    ].value;

if((species_var.selectedIndex !=-1) && (species_check_var===true))
name_species_selected_var = species_var.options[species_var.
    selectedIndex].value;


if((phase_var.selectedIndex!=-1) && (phase_check_var===true))
```

```
40 name_phase_selected_var = phase_var.options[phase_var.selectedIndex
       ].value;
41
42 var url = "homepage.php?";
43 url += name_place_selected_var==""? "": "place="+
       name_place_selected_var;
44 url += (name_species_selected_var ==""  ?  "" : (
       name_place_selected_var==""? "species="+name_species_selected_var
        : "&species="+name_species_selected_var));
45 url += (name_phase_selected_var ==""? "" : ((name_place_selected_var
       ==""  && name_species_selected_var ==""))? "phase="+
       name_phase_selected_var : "&phase="+name_phase_selected_var));
46
47 console.log(url); /*si rgistra nella console l'url di ricerca*/
48 window.location.href = url;
49 }
```

Esempio di url generato in caso di ricerca secondo i filtri scelti e abilitati dell'immagine 3.5 :

homepage.php?place=3&phase=Appassimento

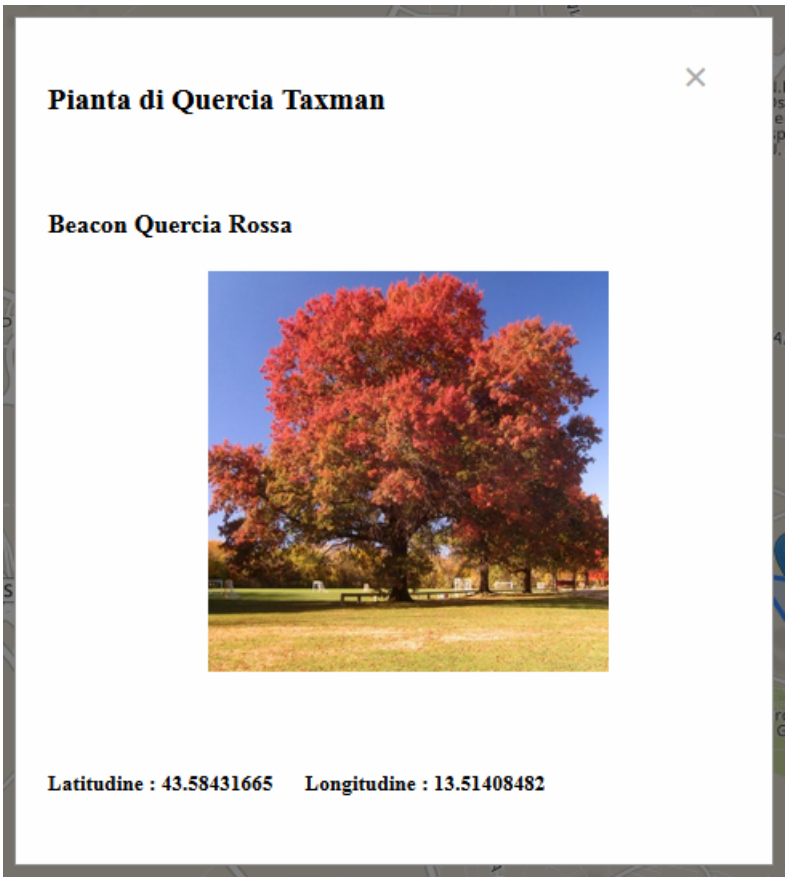
Come precedentemente introdotto, dopo aver effettuato il redirect, nella pagina homepage si verificano se sono stati passati dei parametri. Essendo stati passati i parametri place e phase si eseguirà lo script *search.php*.

3.3.4.7 Finestra Modale dei beacon - Funzione create_popup(..)

Al click di ogni beacon (pianta) presente sulla mappa viene associato l'evento **create_popup**. Questo evento è volto a creare dinamicamente una form modale contenente tutte le informazioni del beacon cliccato e mostrarla a schermo. Di seguito un esempio della form:
Come è possibile vedere dalla foto, la form è formata dalle seguenti parti, partendo dall'alto:

- il tag **h3** che conterrà il nome della pianta dove è stato posizionato il beacon;
- il tag **h4** che conterrà le informazioni del beacon;
- il tag **img** rappresentante l'immagine della pianta;
- due tag **h5** rappresentanti rispettivamente uno la latitudine e l'altro la longitudine del beacon.

Per far si che ogni beacon (pianta) sulla mappa abbia una propria form contenente i propri dati si è scelto di utilizzare una singola form dove ad ogni click vengono modificati dinamicamente tutti i tag introdotti sopra. dalla funzione associata al beacon. La modifica viene fatta dalla funzione che si sta analizzando. Per capire



il funzionamento andremo ad analizzare il codice di quest'ultima.
Dalla form risulta facile capire quali sono i parametri che gli vengono passati a questa funzione:

- il nome della pianta. (colonna *name_plant* della tabella *plant*);
- le informazioni del beacon. (colonna *information* della tabella *beacon*);
- il path della foto della pianta. (*id_photo_plant_fk* della tabella *exists_table*);
- la latitudine e la longitudine del beacon. (colonne *lat* e *lng* della tabella *beacon*).

Si ricorda che questi parametri vengono passati dinamicamente dallo script che inizializza la mappa. (*init.php*)
Si effettuano i seguenti controlli sulle variabili passate:

- si verifica se il path dell'immagine è null: in tal caso non si ha nessuna immagine per la pianta e viene messa un'immagine di default;

- si verifica se si hanno informazioni sul beacon (la colonna information può anche essere null). In caso negativo si stampa un'etichetta che esprime tale mancanza;

- si verifica che il nome della pianta sia diverso da null : se fosse null il beacon non è associato dal alcuna pianta. In questo caso viene mostrata la label dedicata (beacon non associato) e viene messa un'immagine di default.

Quanto detto sopra è riassumibile nel codice utilizzato.

```
function create_popup(plant_name,information,path,lat,lng)
{

if(path == '' || path == null || path =='null' || path === 'null')
document.getElementById('plant_photo').src = "../images/no-image.jpg
    ";

else
document.getElementById('plant_photo').src = "../images/"+path;

if(information == 0 || information == null || information === null
    || information =='null' || information ==='null')
document.getElementById('beacon_info').innerHTML = "Nessuna
    informazione per questo beacon";
else
document.getElementById('beacon_info').innerHTML = information;

if(plant_name == 0 || plant_name == null || plant_name === null ||
    plant_name =='null' || plant_name ==='null')
document.getElementById('plant_name').innerHTML = "Beacon non
    associato";
else
document.getElementById('plant_name').innerHTML =plant_name;


document.getElementById('beacon_lat').innerHTML = "Latitudine : "+
    lat;
document.getElementById('beacon_lng').innerHTML = "Longitudine : "+
    lng;

beacon_modal();
}
```

3.3.4.8 Caricamento del Canvas con Xml HTTP Request

Al click del bottone *Carica* della form per scattare una foto, si chiama la funzione **load_canvas**. Questa funzione invia una richiesta **XHR** per caricare asincronamente il contenuto multimediale. Il funzionamento può essere riassunto dicendo semplicemente che viene creata una **Form Data** contenente i dati che si vogliono

mandare al server in modo asincrono e con la funzione **send** si inviano tali dati. Il codice utilizzato per la XHR è il seguente:

```
var dataURL = canvas.toDataURL("image/png");
document.getElementById('hidden_data').value = dataURL;
var fd = new FormData(document.forms["form1"]);

var xhr = new XMLHttpRequest();
xhr.open('POST', '../php/upload_canvas.php', true);

xhr.upload.onloadend = function(e) {
  alert('Upload eseguito con successo.');
};
xhr.send(fd);
```

3.3.5 Full Front-End

Il *full* front-end è il front-end con funzioni più eleganti, sia dal punto di vista grafico che dal punto di vista del codice.
Come già detto, tutti gli eventi e stili sui vari elementi utilizzati (form modali, menù a tendina ecc..) vengono gestiti dal framework. Verranno analizzati alcune funzionalità grafiche e funzionali del JQuery mentre non verranno nuovamente affrontate construtti o form che risultino strutturalmente identiche a quelle dell'altro front-end.

3.3.5.1 Side Navbar

Come per l'altro front-end, anche in questo viene inserita una sidebar sulla sinistra. Quasi tutte le funzioni e le ancore sono le stesse, ad eccezione di come si effettua la ricerca che verrà analizzata successivamente.
In questa barra troviamo una particolare funzionalità del framework grafico, la **Collapsible** che è una sorta di menù a tendina collassabile che si espande al click.
La voce Segnala un problema non è presente nella sidebar ma è stata posizionata sulla navbar, esattamente in alto a destra.
La sidebar è composta dalle seguenti voci:

- **Effettua la ricerca** : Ancora che permette di effettuare la ricerca invocando lo *script.php*;
- **Place** : è un Collapsible "fillato" dinamicamente con tutti gli orti botanici memorizzati nel sistema;
- **Species** : è un Collapsible fillato dinamicamente con tutte le specie memorizzate nel sistema;
- **Phenological** Phase : è un Collapsible fillato dinamicamente con tutte le fasi fenologiche memorizzate nel sistema;
- **Filtri** : è un Collapsible dove all'interno è possibile visualizzare i filtri scelti e resettarli;
- **Scatta una Foto** : ancora che permette di aprire la finestra modale relativa al caricamento di contenuti multimediali;
- **Nascondi Beacons** : ancora che permette di nascondere tutti i beacon dalla mappa. Al click quest'ancora prende la label 'Mostra Beacons' che consente di effettuare l'operazione opposta;
- **Nascondi Poligoni** :ancora che permette di nascondere tutti gli orti botanici dalla mappa. Al click quest'ancora prende la label 'Mostra Poligoni' che consente di effettuare l'operazione opposta;

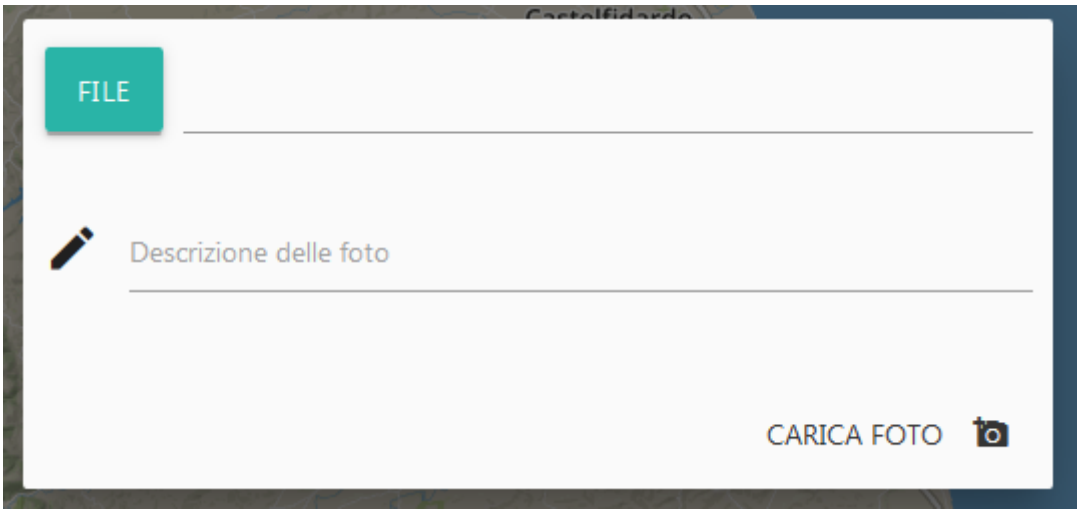
- **Gestisci il Database** : ancora che permette di aprire la finestra modale relativa all'inserimento delle credenziali per effettuare il login ed accedere a *manage.php*;
- **Aggiorna la Pagina** : ancora che effettua il refresh della pagina;
- **Basic Website** : ancora che effettua un redirect all'altro front-end (basic front-end).

3.3.5.2 Form Caricamento Immagini

La parte di WebRTC non viene utilizzata, per gestire la fotocamera si usa il tag input dell'**HTML5** di tipo file. Questo tag si comporta in modo differente a seconda del dispositivo:

- In una piattaforma desktop si aprirà la **file browser dialog** che permetterà di caricare un file esistente;
- In una piattaforma mobile (smartphone) si aprirà la fotocamera con l'applicazione di sistema : dopo aver scattato la foto il dispositivo chiederà di utilizzarla o meno, assegnandola all'elemento html.

Ecco come si presenta la form :

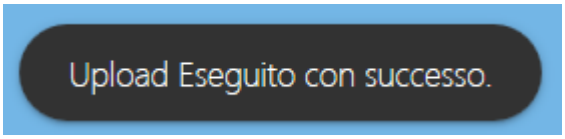


Effettuando il click su *Carica Foto*, dopo aver superato i controlli di validità sui dati, si esegue come per il caricamento del canvas, un richiesta asincrona. Anche qui si crea una **Form Data** dove si appendono i dati da inviare in modo asincrono al server. La richiesta viene effettuata con il JQuery:

```
var file_data = $('#fileToUpload').prop('files')[0];
var descr_data = $('#descr_text').val();
var form_data = new FormData();
```

```
form_data.append('file', file_data);
form_data.append('descr_text',descr_data);
$.ajax({
  url: 'php/upload.php',
  dataType: 'text',
  cache: false,
  contentType: false,
  processData: false,
  data: form_data,
  type: 'post',
  success: function(){
    var upload_success = "<span>Upload Eseguito con successo.</span>";
    M.toast({html: upload_success,classes: 'rounded'});
  },
  error: function(xhr){
    var error = '<span> Request Status: ' + xhr.status + ' Status Text: ' + xhr.statusText + ' ' + xhr.responseText +'</span>'
    M.toast({html: error});
  }
});
```

Appena la richiesta asincrona è conclusa, viene mostrato un messaggio attraverso la funzionalità **Toast** di Materialize. Ecco un esempio in caso di caricamento eseguito con successo:



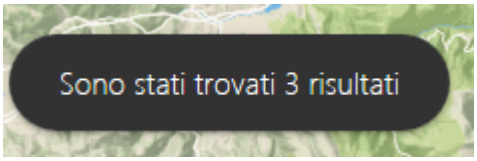
3.3.5.3 Effettuare una Ricerca

In questo front-end la ricerca ha come obiettivo di essere espletata con meno interazioni possibili da parte dell'utente. Non viene quindi utilizzata una form ma i **Collapsible** della sidebar.
Per effettuare una ricerca si procede come segue:

1. si effettua un click sul Collapsible del filtro desiderato;
2. si sceglie il filtro desiderato dal Collapsible che si è espanso;
3. al click del filtro scelto verrà eseguito un Toast con la scelta effettuata;
4. si procede alla scelta o meno degli altri filtri;
5. per visualizzare i filtri immessi o resettarli basta navigare tra gli elementi del Collapsible **Filtri**;

6. infine basta cliccare su **Effettua La Ricerca**.

L'url creato dinamicamente con i filtri selezionati viene creato in modo analogo a quello dell'altro front-end. Appena viene effettuata la ricerca, si esegue un **Toast** con il suo esito.



3.3.5.4 Finestra Modale dei beacon - Funzione create_popup(..)

Anche in questo front-end le finestre modali vengono create con la stessa identica logica analizzata nella sezione **3.3.4.7**. A differenza dell'altro front-end in questo grazie a Materialize, oltre a un design più accattivante viene utilizzata **Material Box** che è un'implementazione del del plugin material design **Lightbox**. Al click dell'immagine Material Box la centra e la allarga con un dolce effetto di transizione. Per uscire dall'immagine ingrandita è sufficiente premere ESC, cliccare sull'immagine o scrollare la pagina.

3.3.6 Javascript vs JQuery

3.3.6.1 Approccio nel Coding

Si può affermare che usando

```
1 $('#myElement').click(function(){ ... }); /*JQuery*/
```

sia meglio dal punto di vista dell'**event registration model** dato che JQuery usa internamente **addEventListener** and **attachEvent** per gestire gli eventi.
Registrando un evento utilizzando questa scrittura moderna è un modo di gestire gli eventi con **Unobtrusive JavaScript**.
L' Unobtrusive JavaScript è un tipo di approccio che prevede che:

- separazione delle funzionalità (*behavior layer*) della struttura/contenuto e presentazione della pagina web;
- migliori pratiche per evitare i problemi della programmazione JavaScript tradizionale (come le incoerenze del browser e la mancanza di scalabilità);
- miglioramento progressivo per supportare i programmi utente che potrebbero non supportare la funzionalità JavaScript avanzata.

Inoltre per registrare più di un event liestener per un target si può chiamare addEventListener() per lo stesso target come segue:

```
var myEl = document.getElementById('myelement');

myEl.addEventListener('click', function() {
alert('Ciao Mondo!');
}, false);

myEl.addEventListener('click', function() {
alert('Ciao Mondo di nuovo!');
}, false);
```

Altri metodi come l'**HTML attributes**:

```
<button onclick="alert('Ciao Mondo!')">
```

oppure come il **DOM element properties**:

```
myEl.onclick = function(event){alert('Ciao Mondo');};
```

sono vecchi paradigmi e possono essere sovrascritti facilmente, questo perchè :

- l'attributo HTML dovrebbe essere evitato in quanto rende il markup più grande e meno leggibile. Struttura / contenuto e comportamento non sono ben separate, rendendo più difficile trovare un bug;
- il problema con il metodo delle proprietà dell'elemento DOM (Document Object Model) è che solo un gestore di eventi può essere associato a un elemento per evento.

3.3.6.2 Performance

Per una migliore performance si usa il JavaScript nativo mentre per uno sviluppo più veloce si usa jQuery.

Effettuando un test con Firefox 16.0 32-bit su Windows Server 2008 R2 / 7 64-bit sulla performance si ottengono i seguenti dati:

```
$('myElement'); /*6,604 operations per second - JQuery*/
document.getElementsByTagName('myElement'); /*10,331,708 operations/
    sec - Native Javascript*/
```

Capitolo 4

Set up dell'applicazione web

In questo capitolo vengono introdotti i software che permetteranno di configurare l'applicazione web realizzata affinché sia online.

4.1 Hosting

L'hosting è un servizio che permetta di caricare dei contenuti su un server web affinché sia accessibile dalla rete internet. Si utilizza un Hosting di Amazon (a pagamento) che è una macchina virtuale dove vi è installato **Bitnami** (web server), e vi tutti gli strumenti necessari per procedere al setup.

4.2 Database

Si procede alla creazione del database sul nostro host, è necessario quindi connettersi al modulo PhpMyAdmin dell'hosting.

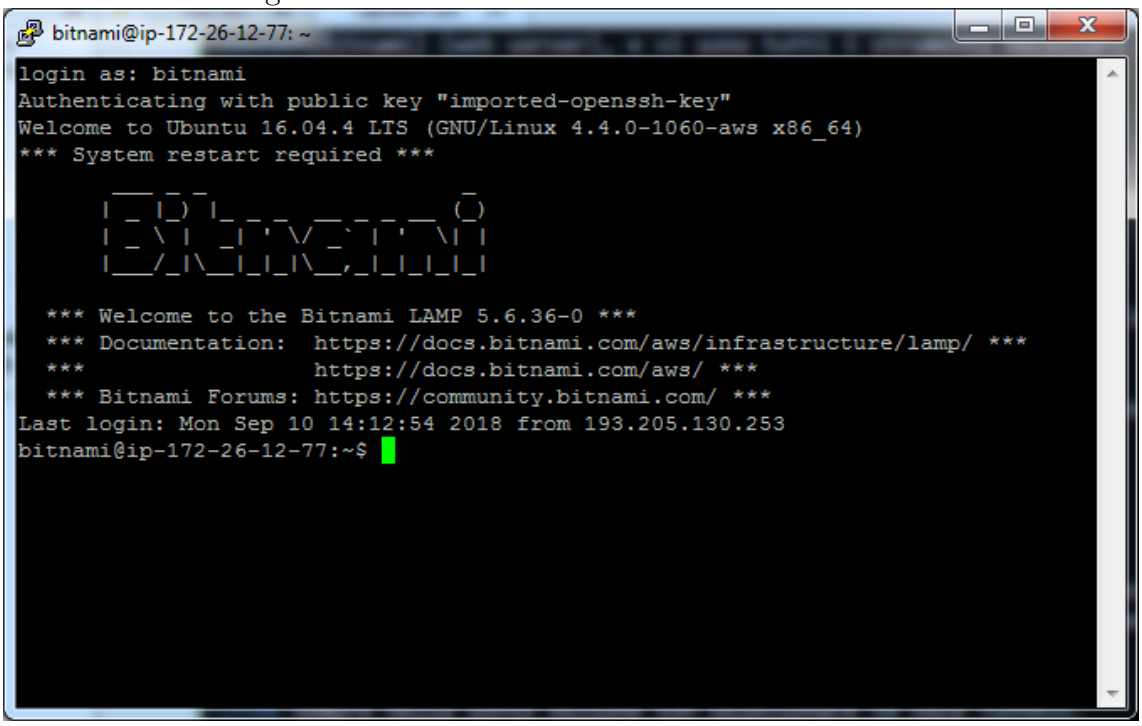
4.2.1 Putty Setup

Per effettuare il setup dell'applicativo Putty si devono effettuare i seguenti passaggi:

1. installare il Client SSH per la gestione in remoto di sistemi informatici **Putty**;
2. aprire Putty;
3. nella sezione **Session**, nel campo **Host Name** inserire l'indirizzo IP del nostro hosting;
4. selezionare la porta 80;
5. entrare sezione **Connection**;

6. entrare nella sotto sezione **SSH**;
7. selezionare la voce **Auth**;
8. nella sezione **Authentication Parameters** trovare la voce **Private key File for Authentication**, cliccare Sfoglia/Browse e selezionare il file di autenticazione fornitoci (*LightEC2private.ppk*);
9. sempre nella sotto sezione SSH selezionare la voce **Tunnels**;
10. settare la **Source Port** con **8888**;
11. settare **Destination**;
12. cliccare Aggiungi/Add;
13. assicurarsi che la porta aggiunta sia tra le **Forwarded Ports**;
14. cliccare **Apri/Open** per aprire la connessione SSH con il server. La sessione SSH includerà ora un tunnel SSH sicuro tra le due porte specificate;
15. inserire come username **bitnami**;

Figura 4.1: Connessione stabilita con successo



Lasciando il terminale aperto si prosegue con la connessione a phpmyadmin.

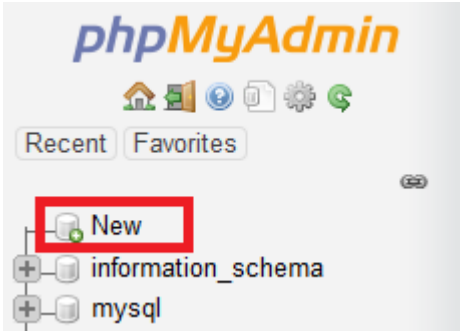
4.2.1.1 Connessione a PhpMyAdmin

Per connettersi a questo modulo si procede come segue:

1. si apre il web browser;
2. si naviga all'indirizzo **http://127.0.0.1:8888/phpmyadmin/**;
3. si apre il pannello di login di phpmyadmin come in figura:

4. si immettono le credenziali - Username : **root** -Password;
5. si clicca **Go** per entrare nel pannello di PhpMyAdmin;

Entrati nel pannello si procede alla creazione del database, cliccando sulla voce new dalla **tree-view** sulla sinistra. Fare riferimento alla seguente immagine.



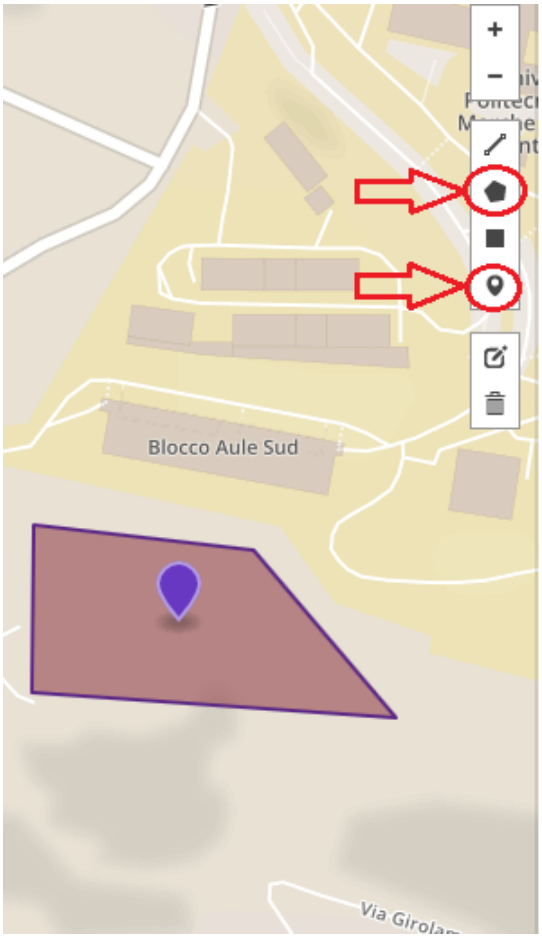
Ora si procede come segue:

1. Inserire il nome del database (**orto_botanico**).
2. Selezionare il set di caratteri **utf8_general_ci**.
3. Premere il bottone **Create**.
4. Selezionare dalla tree-view il database **orto_botanico**.
5. Cliccare la Tab **SQL**.
6. Creare le tabelle definite una per una nella sezione **3.1.3**. Le tabelle devono rispettare il seguente ordine a causa delle dipendenze da chiavi esterne:
   - **photo**;
   - **beacon**;
   - **species**;
   - **phenological_phase**;
   - **place**;
   - **plant**;
   - **exists_table**;
   - **visitors**;
   - **advice**;

A questo punto è possibile iniziare a inserire i dati nel database.

4.2.2 geojson.io

Per comodità per la **latitudine** e **longitudine** di un elemento e per il **boundary** di un orto botanico si utilizza il sito web **http://geojson.io**. Ecco un esempio di utilizzo:




```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "stroke": "#5a2585",
        "stroke-width": 2,
        "stroke-opacity": 1,
        "fill": "#842634",
        "fill-opacity": 0.5
      },
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              13.512325286865234,
              43.584914020494175
            ],
            [
              13.512314558029175,
              43.58419126953671
```

Come è possibile vedere dall'immagine, sulla destra troviamo il codice che poi sarà inserito nella colonna **boundary**. Sul riquadro della mappa invece, sulla destra si trova una sidebar volta a creare vari elementi. Quelli cerchiati sono quelli di interesse, utilizzati nel progetto.
Inoltre cliccando sull'elemento creato nella mappa è possibile specificare ulteriori parametri come suggeriscono le seguenti immagini.

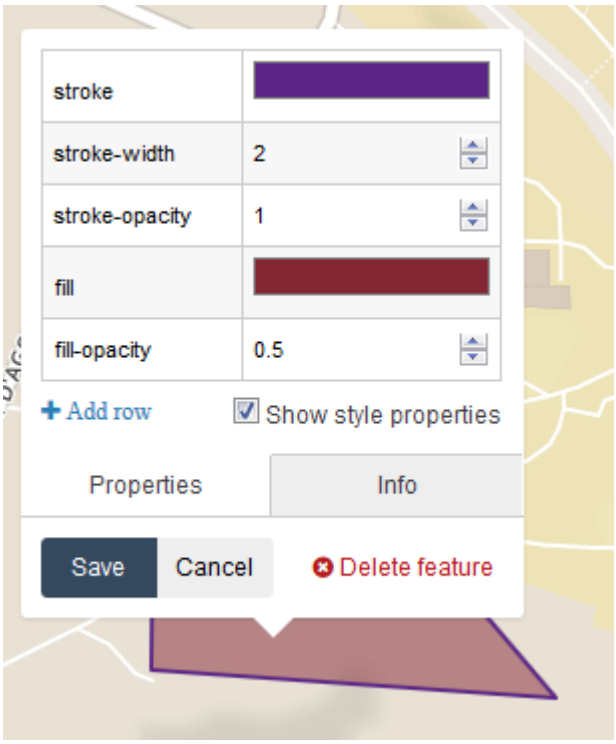


Figura 4.2: Click sul poligono : possibilità di specificare attributi grafici come il colore, la linea di contorno e l'opacità.

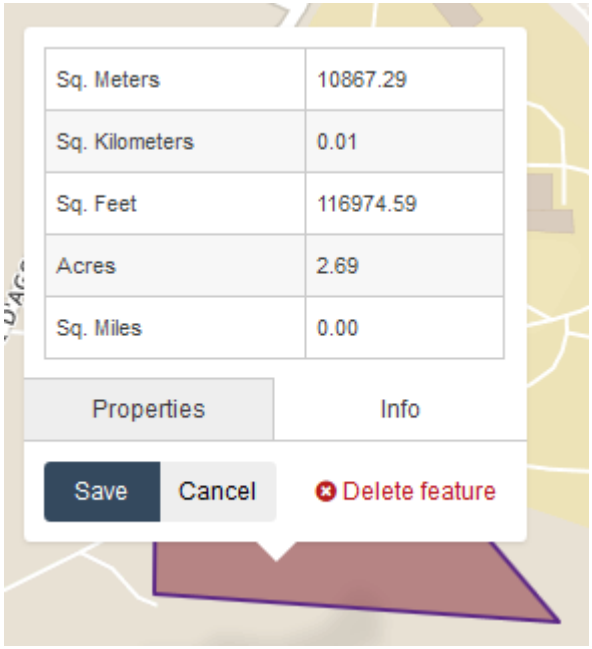


Figura 4.3: Click sul poligono : informazioni utili.

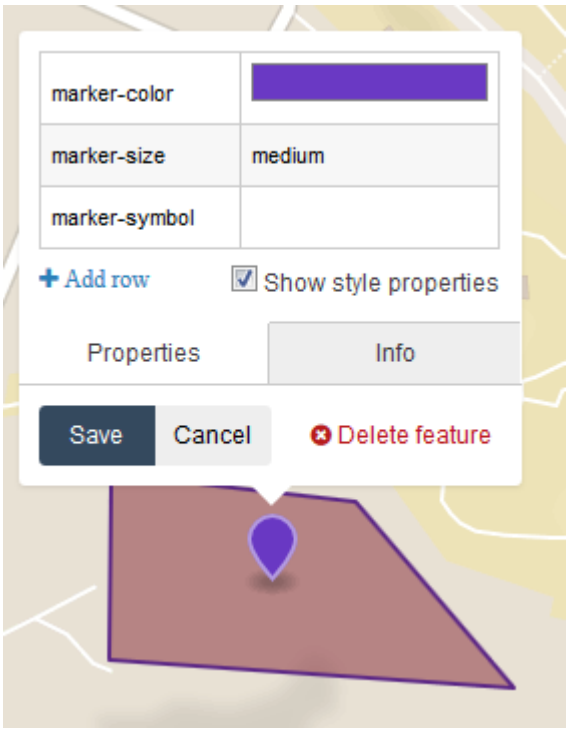


Figura 4.4: Click sul beacon : possibilità di specificare attributi grafici come il colore, il simbolo e la dimensione.

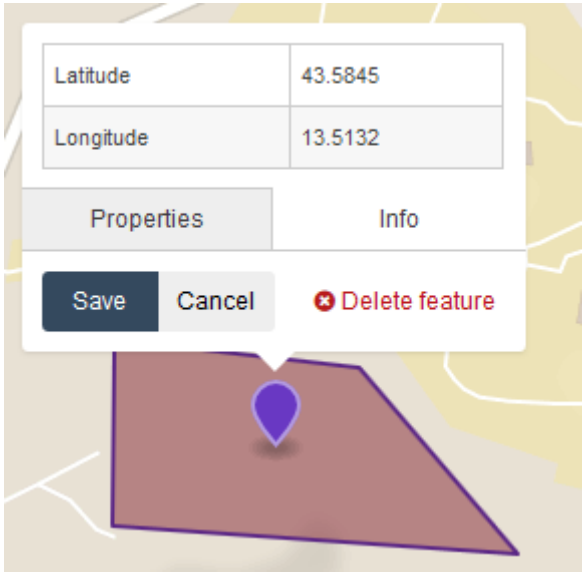


Figura 4.5: Click sul beacon : latitudine e longitudine.

4.3 Caricamento Files con WinSCP

WinSCP, acronimo di Windows Secure Copy che è un client SFTP e FTP di tipo grafico che ha come funzione principale quella di trasferimento dati tra un computer locale ed uno remoto. In più WinSCP consente semplici funzionalità di file manager. Per trasferimenti sicuri usa Secure Shell (SSH) e supporta il protocollo SCP in aggiunta al SFTP [16].
Per configurare la connessione e effettuare il caricamento dei file si procede come segue:

1. si Installa WinSCP;
2. si avvia il client WinSCRP;
3. si imposta l'indirizzo IP;
4. si imposta come username **bitnami**;
5. si clicca su **Advanced**, a questo punto si apre una finestra;
6. cliccare la sezione **SSH**;
7. aprire la voce **Authentication**;
8. immettere la chiave *LightEC2private.ppk* sul campo **Private Key File**;
9. sfogliare le directory del server fino a raggiungere **/opt/bitnami/apache2/htdocs**;
10. trascinare i files desiderati ricordando di applicare le modifiche al file config.php che sviluppando in locale ha parametri differenti;

4.4 Disabilitare il modulo Pagespeed

Per evitare eventuali problemi di visualizzazione delle icone dei beacon sulla mappa, è necessario disabilitare il modulo pagespeed. Per fare ciò bisogna localizzare il file **pagespeed.conf** che è situato nella directory **/opt/bitnami/apache2/conf**, aprire il file e modificare la linea **ModPagespeed on** in **ModPagespeed off**. Infine è necessario riavviare il servizio di apache con il comando **service httpd restart**.

4.5 Permessi per uploads

Per far funzionare correttamente gli script php di upload e non incorrerre errori particolari è necessario dare tutti i permessi alle seguenti cartelle:

- /opt/bitnami/php/tmp
- /opt/bitnami/apache2/htdocs/images

Questa operazione può essere effettuata tramite terminale con i seguenti passaggi:

1. Ci si assicura che la connessione SSH sia aperta.
2. Si ottengono i permessi di amministratore utilizzando il comando **sudo su**.
3. Si esegue l'istruzione chmod 755 -R /opt/bitnami/php/tmp per i permessi della cartella temp.
4. Si esegue l'istruzione chmod 755 -R /opt/bitnami/apache2/htdocs/images per i permessi della cartella images.

In alternativa è possibile utilizzare **WinSCP** : cliccando su proprietà è possibile impostare i permessi come in figura.

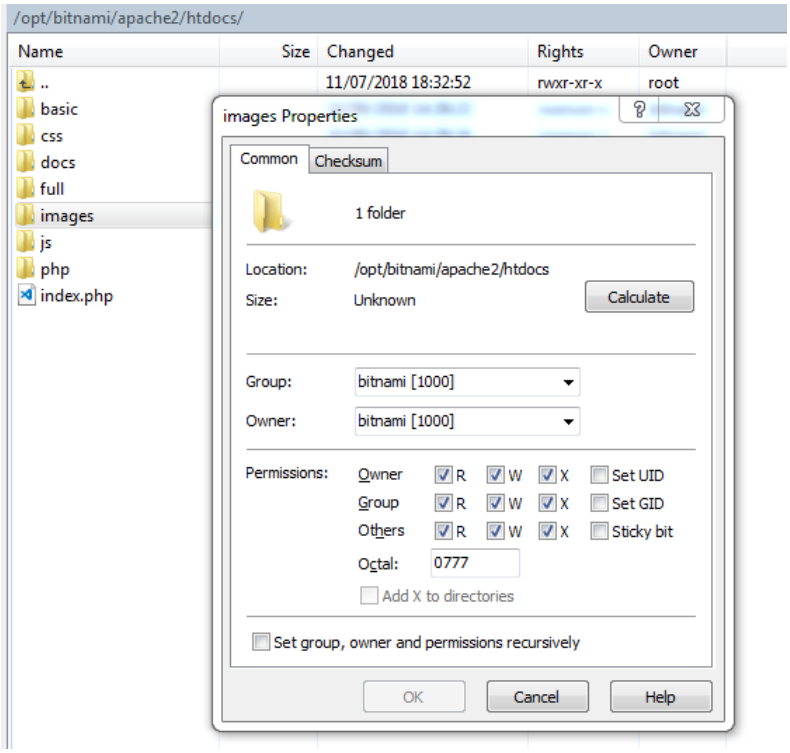


Figura 4.6: Proprietà : Assicurarsi che Write sia spuntato

4.6 Accesso all'applicazione Web con il protocollo HTTPS

Per accedere da desktop all'applicazione web utilizzando il protocollo sicuro è necessario aggiungere un'eccezione al certificato che è **self-signed**.

Figura 4.7: Aggiungendo l'eccezione è possibile navigare nel sito utilizzando a pieno le API

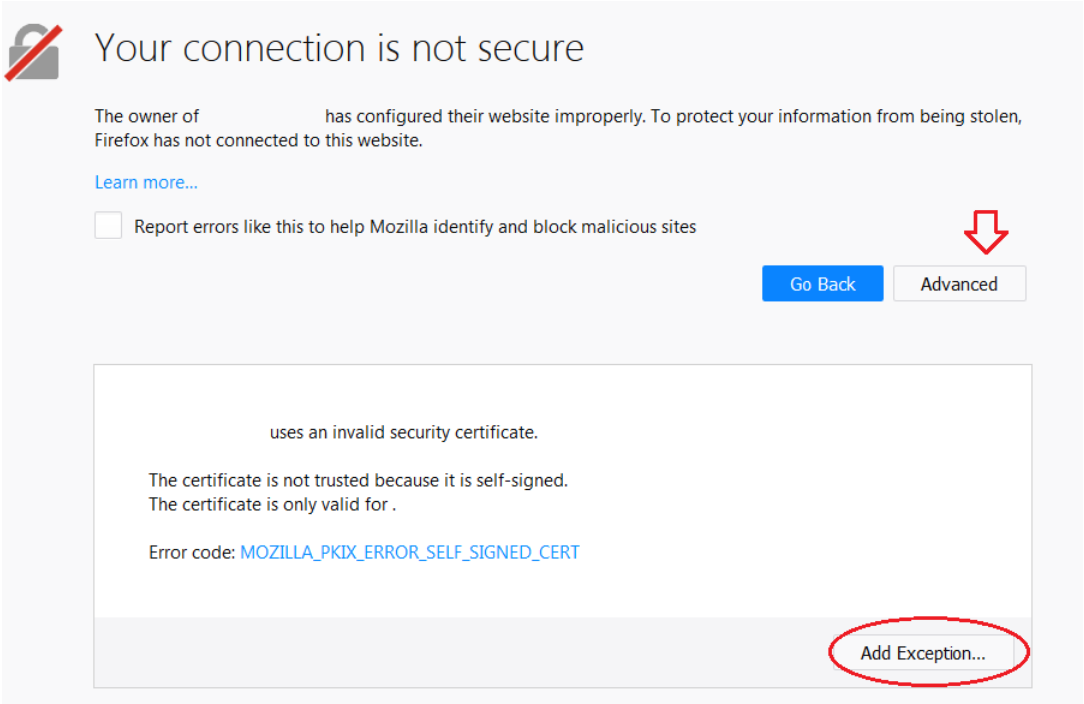


Figura 4.8: Confermando l'eccezione sarà possibile la navigazione

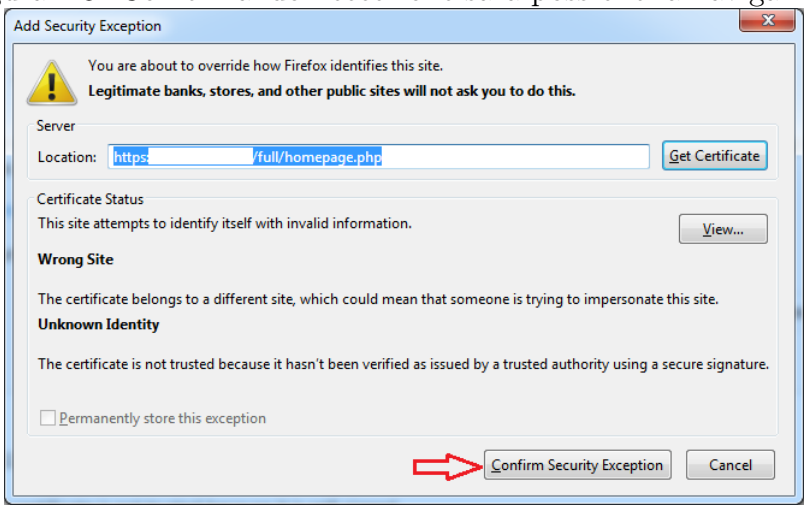
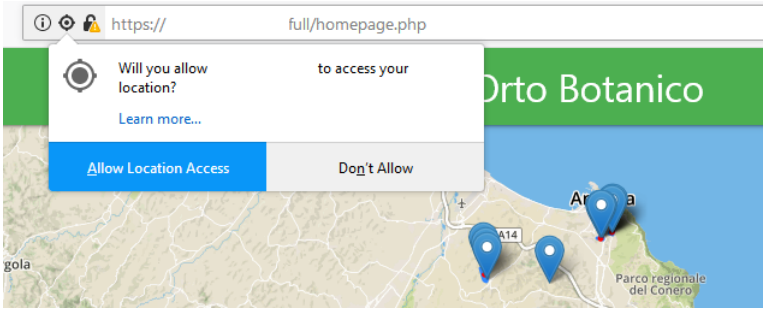


Figura 4.9: Richiesta di Localizzazione da Desktop



4.7 Gestione del Database tramite script manage.php

Dopo aver eseguito tutte le configurazioni precedentemente introdotte, tramite la pagina manage.php non si ha più necessità di utilizzare il modulo PhpMyAdmin per manipolare i dati nel database.
Ora verranno introdotte le funzioni restanti messe a disposizione da questo script che non sono state introdotte in precedenza.

- modifica del nome di una tabella, cliccando semplicemente sul nome;
- notifica tooltip del tipo della colonna, lasciando il puntatore (evento **onhover**) sopra il nome della colonna;
- modifica del tipo della colonna, cliccando sopra il nome della colonna;
- modifica del nome della colonna, cliccando sulla cella, sulla parte esterna del nome;

Si ricordano inoltre le funzioni già introdotte, quali:

- modifica dei valori di un record;
- inserimento di un record facilitato tramite utilizzo di menù a tendina già formattati con i dati idonei (es: possibili valori di una enumerazione o di una chiave esterna);
- eliminazione di un record;

Capitolo 5

Panoramica dell'applicazione web da dispositivo mobile

In questo capitolo verrà effettuata una panoramica di come si presenta l'applicazione web da un dispositivo mobile.

5.1 Dispositivo

Il dispositivo utilizzato per la navigazione sulla web app realizzata è un dispositivo Apple, precisamente un iPhone che ha come sistema operativo iOS.
Il browser utilizzato è Safari che risponde perfettamente a tutte le funzionalità dell'applicazione web.

5.1.1 Set Up del Dispositivo

Per far si che l'applicazione funzioni correttamente sono necessarie determinate impostazioni che sono elencate di seguito.

- Aprendo le impostazioni, andando sulla sezione privacy e aprendo la voce **Localizzazione** è possibile attivare il servizio di localizzazione del dispositivo;

Figura 5.1: Impostazione che permette tramite spunta di abilitare l'opzione di localizzazione

- Sempre nella stessa sezione si danno i permessi all'applicazione Safari di utilizzare tale servizio;

Figura 5.2: Impostazione che permette tramite spunta di selezionare la voce Mentre si usa l'app

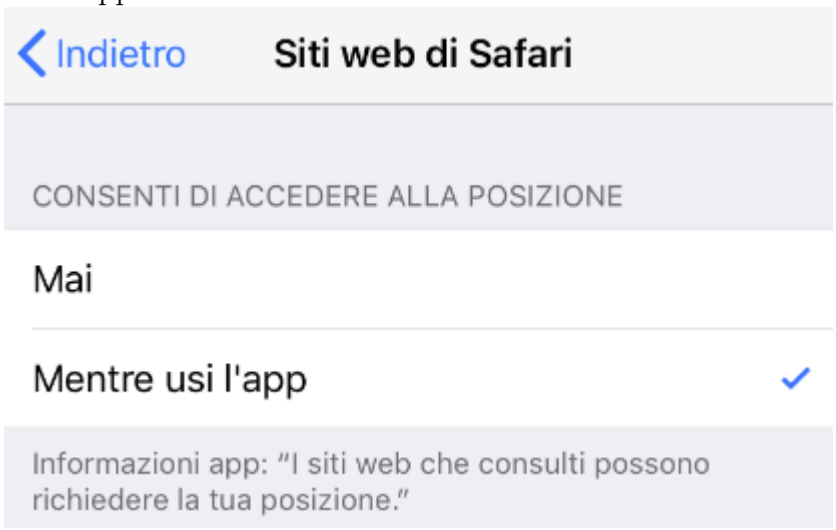
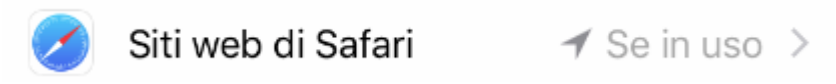
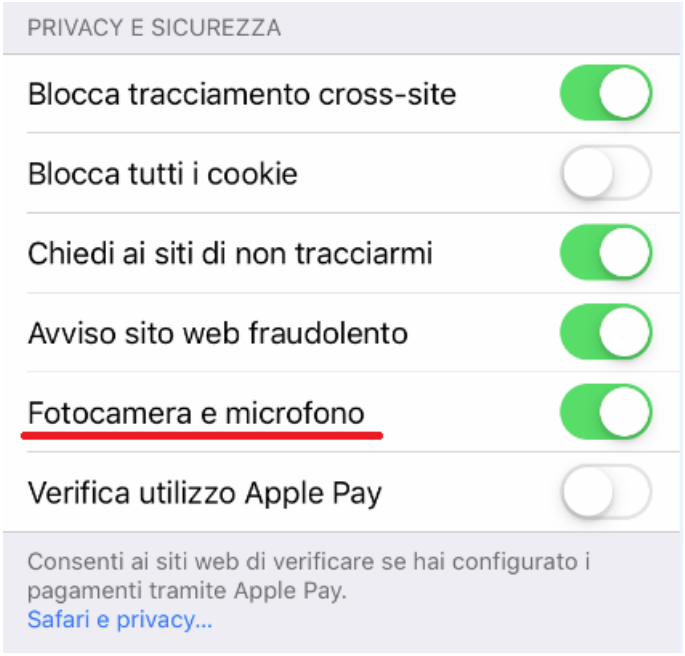


Figura 5.3: Dimostrazione della corretta procedura, Safari ha ora ottenuto i permessi



- Sempre dalle impostazioni del dispositivo, selezionando safari è possibile abilitare l'utilizzo della fotocamera spuntando la relativa voce. (Questo passaggio è essenziale affinché nel front-end basic funzioni la parte per scattare una foto);

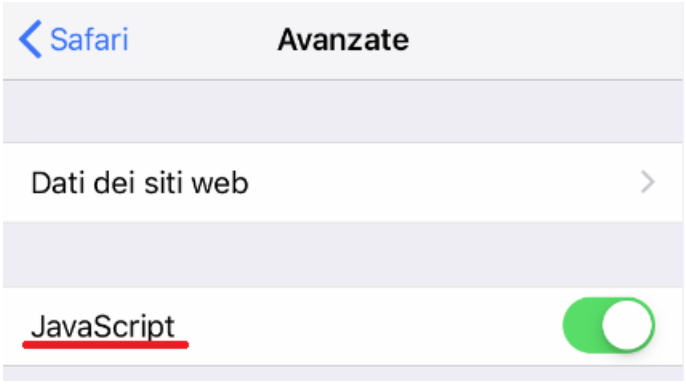
Figura 5.4: Impostazioni che permettono tramite spunta delle voci 'Microfono e Fotocamera', di abilitare tali servizi



- Aprire l'ultima voce della sezione di Safari (Avanzate) e abilitare la voce **JavaScript**;

- A questo punto il dispositivo è settato correttamente e per accedere al sito si deve utilizzare il protocollo **HTTPS** e non il protocollo HTTP (altrimenti le API non funzionerebbero).

Figura 5.5: Impostazione che permette tramite spunta di abilitare le funzionalità 'JavaScript'



5.2 Navigazione dell'applicazione Web

5.2.1 Full Front-End

5.2.1.1 Richiesta di Localizzazione

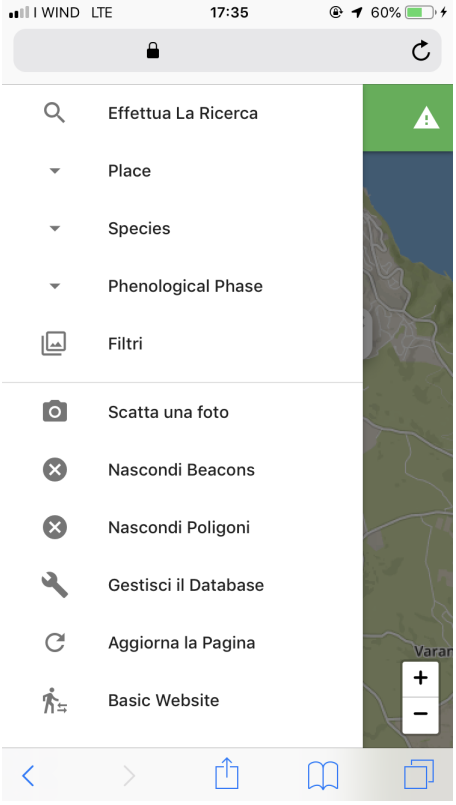
Figura 5.6: Richiesta di Localizzazione da parte dell'API, accettando si acconsente a condividere la proprio posizione affinché l'applicazione web funzioni correttamente.

5.2.1.2 Homepage

Figura 5.7: Come si presenta l'homepage dell'applicazione Web con il front end full

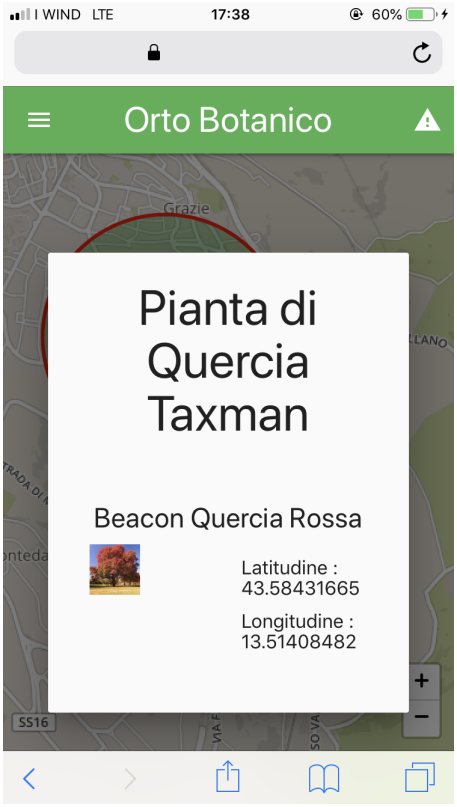
5.2.1.3 Sidebar

Figura 5.8: Come si presenta la sidebar di questo front end



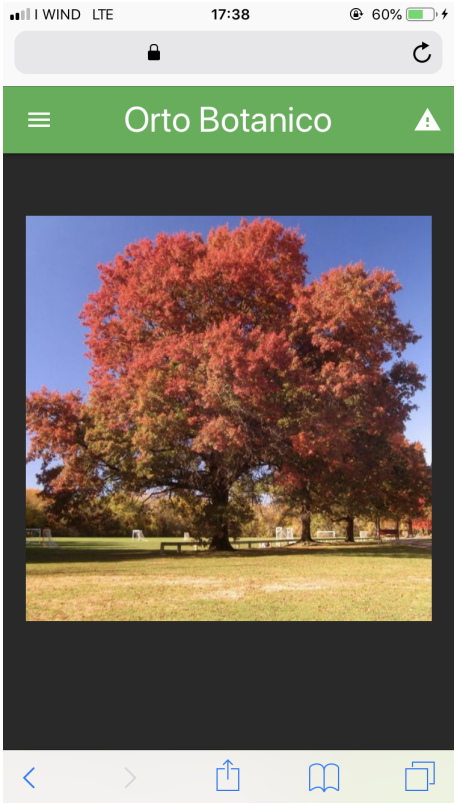
5.2.1.4 Form Modale al click del Beacon

Figura 5.9: Ecco come si presenta la form che viene mostrata al click del beacon sulla mappa



5.2.1.5 Material Box della Form Modale del Beacon

Figura 5.10: Ecco come si presenta la Material Box (Light Box) al click dell'immagine della finestra modale del beacon



5.2.2 Basic Front-End

5.2.2.1 Homepage

Figura 5.11: Come si presenta l'homepage dell'applicazione Web con il front end basic

5.2.2.2 Sidebar

Figura 5.12: Come si presenta la sidebar di questo front end

5.2.2.3 Finestra Modale per scattare Foto

Figura 5.13: Come si presenta la form modale per scattare le foto.

Capitolo 6

Conclusioni e Sviluppi Futuri

6.1 Conclusioni

Lo sviluppo dell'applicazione web, secondo gli obiettivi fissati, è stato eseguito con successo. Per realizzare questo progetto sono stati utilizzati vari linguaggi e tecniche di programmazione che utilizzati insieme hanno portato a questo risultato. Tuttavia si sarebbero potuti utilizzare differenti approcci e differenti ambienti di sviluppo integrato per realizzare questo applicativo. Tali ambienti sarebbero potuti essere Xamarin, Mono, Android Studio o altri ancora.
L'applicazione web presenta, per come è stata fatta, un grande margine di miglioramento. Possono essere fatte molte implementazioni di nuove funzioni; ora andremo a vedere quali potrebbero essere.

6.2 Sviluppi futuri

6.2.1 Implementazione AJAX

Utilizzando l'applicazione web, notiamo subito che essa non è interattiva, cioè non aggiorna automaticamente i dati se non eseguendo un refresh. Un miglioramento consistente sarebbe implementare una logica di aggiornamento instantaneo della mappa utilizzando l'AJAX. In questo modo qulunque modifica fatta dal pannello di amministrazione (*manage.php*) viene mostrata istantaneamente sulla mappa senza eseguire un refresh della pagina.
Ciò serve per avere una corretta visualizzazione in tempo reale dei dati, in modo da tener sempre aggiornati gli utenti per eventuali modifiche e aggiornamenti.

6.2.2 Implementazione logica admin-staff-user

Un altro accorgimento che si fa utilizzando l'applicazione web è l'impossibilità di avere un proprio account. Questo tipo di struttura per l'autenticazione mira ad

implementare un sistema di accesso basato su 3 livelli: **user**,**staff** e **admin**. Si dovrà poi inserire un pannello per la registrazione di account user e un pannello di registrazione per account staff nella pagina *manage.php*. Inoltre si implementa un sistema di notifiche.
Dopo l'autenticazione ogni tipo di account avrà a disposizione un'area riservata come segue:

- **user** : la possibilità di salvare orti o piante di interesse, in modo da ottenere notifiche per le fioriture;
- **staff** : questo tipo di account può accedere ad alcune funzioni del pannello di controllo (*manage.php*) quali l'inserimento di beacon sulla mappa, il caricamento di immagini, lettura dei report e l'utilizzo di una nuova funzione (non implementata al momento) che permetta di associare un beacon ad un pianta;
- **admin** : controllo generale del pannello *manage.php* e l'implementazione di una nuova sezione volta a creare utenti nel database;

6.2.3 Più funzioni in manage.php per manipolare i dati nel database

Un altro aspetto futuro l'implementazione di una sezione nella pagina *manage.php* in grado di effettuare ulteriori manipolazioni al database, quali:

- possibilità di eseguire codice SQL tramite un editor creato ad hoc. (Per esempio per la creazione delle tabelle);
- possibilità di specificare i parametri di connessione e non renderli statici dal file *config.php* (in modo da connettersi anche ad altri database);
- possibilità di esportare un backup del database;
- possibilità di importare un backup del database per effettuare un ripristino dei dati;
- possibilità di creare in maniera guidata delle viste;
- possibilità di creare in maniera guidata delle function;
- possibilità di creare in maniera guidata delle stored procedures;
- possibilità di creare in maniera guidata degli indici.

6.3 Correzione dei BUG riscontrati

Durante lo sviluppo, si sono riscontrati svariati bug con l'utilizzo dell'API per gestire la fotocamera dei dispositivi, verranno elencati i più rilevanti:

- non funziona con il browser Google Chrome da mobile (versione rilasciata in data 06/2018);
- per cambiare fotocamera da quella anteriore a quella posteriore si deve utilizzare il menù a tendina svariate volte prima che abbia effetto;
- la funzione *drawImage*, che riceve in input l'immagine in streaming al momento dello pressione del bottone *Scatta* dall'API, permette di disegnare l'immagine sul canvas, presenta un grave problema di resizing come è possibile vedere dalla figura **5.1.3**.

6.4 Ottimizzazione del Codice

Per quanto riguarda il codice, si potrebbero usare diversi approcci per avere più pulizia e efficienza. Dato che non era questo il concept del progetto, verranno ora elencati alcuni accorgimenti possibili per ottimizzare l'efficienza dell'applicazione web. Si può approccio per la gestione degli eventi come definito nella sezione 3.3.6.1 separando le funzionalità delle struttura/contenuto e presentazione della pagine web.
Dunque eliminando tutti gli *onclick* e utilizzare *addEventListener* nella pagina *manage.php*.

6.5 Altri accorgimenti futuri

Un possibile accorgimento futuro per rendere l'applicazione web più professionale e desiderabile online è quella di acquistare un dominio, in modo che per raggiungere l'applicazione web si utilizza un'etichetta al posto dell'indirizzo ip.
Un altro accorgimento è quello di acquistare un certificato in modo che ci si possa connettere all'applicazione web da browser utilizzando una connessione sicura (Https).

Bibliografia

[1] http://api.jquery.com/jquery.ajax/.

[2] https://code.visualstudio.com/.

[3] https://developer.mozilla.org/en-us/docs/web/api/webrtc_api.

[4] https://en.wikipedia.org/wiki/css_framework.

[5] https://httpd.apache.org/.

[6] https://it.wikipedia.org/wiki/browser.

[7] https://it.wikipedia.org/wiki/database_management_system.

[8] https://it.wikipedia.org/wiki/front-end_e_back-end.

[9] https://it.wikipedia.org/wiki/innodb.

[10] https://it.wikipedia.org/wiki/server_web.

[11] https://jquery.com/.

[12] https://leafletjs.com/.

[13] https://php.net/.

[14] https://stackoverflow.com/questions/3003145/how-to-get-the-client-ip-address-in-php.

[15] https://webrtc.github.io/samples/.

[16] https://winscp.net/eng/docs/lang:it.

[17] https://www.apachefriends.org/it/index.html.

[18] https://www.mysql.com/about/legal/licensing/oem/.

[19] https://www.phpmyadmin.net/.

[20] https://www.w3.org/html/.

[21] https://www.w3.org/standards/webdesign/script.

[22] https://www.w3.org/style/css/overview.en.html.

[23] https://www.w3.org/tr/geolocation-api/.

[24] https://www.w3schools.com/sql/.

Elenco delle figure