

# UNIVERSITÀ DI SIENA

1240

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE E  
SCIENZE MATEMATICHE

---

Corso di Laurea in  
INGEGNERIA INFORMATICA E DELL'INFORMAZIONE

Elaborazione e visualizzazione di dati  
geo-spatiali per una mappatura  
dell'ambiente marino.

**Relatore:**  
**Prof. Alessandro Pozzebon**

**Candidato:**  
**Aldi Piroli**

Anno Accademico 2016-2017

*A mia madre, mio padre e ad Alessio,  
le persone più importanti nella mia vita.*

# Indice

<b>Introduzione</b>	<b>1</b>
<b>1 Analisi Delle Tecnologie Software Utilizzate</b>	<b>3</b>
1.1 QGIS . . . . .	3
1.2 R . . . . .	3
1.3 Google Maps JavaScript API . . . . .	4
1.4 Database e Hosting utilizzati . . . . .	4
1.5 Linguaggi di formattazione e programmazione web . . . . .	4
1.5.1 HTML XML e CSS . . . . .	4
1.5.2 PHP . . . . .	4
1.5.3 JavaScript . . . . .	5
1.5.4 AJAX . . . . .	5
1.5.5 SQL . . . . .	5
<b>2 Algoritmi di creazione, visualizzazione e simulazione</b>	<b>6</b>
2.1 Creazione delle mappe . . . . .	6
2.1.1 Creazione del ShapeFile . . . . .	6
2.1.2 Elaborazione dei dati e creazione della mappa . . . . .	10
2.1.3 Salvataggio della misurazione nel database . . . . .	14
2.1.4 Grafici 3D . . . . .	15
2.1.5 Librerie usate . . . . .	16
2.2 Visualizzazione delle mappe . . . . .	17
2.2.1 Descrizione del portale . . . . .	17
2.2.2 Descrizione degli script principali . . . . .	18
2.3 Simulazione utilizzando dati auto-generati . . . . .	22
2.3.1 Random points inside a polygon . . . . .	22

2.3.2	Regular points inside a polygon . . . . .	22
<b>3</b>	<b>Applicazioni e sviluppi futuri</b>	<b>23</b>
3.1	Drone marino . . . . .	23
3.2	Ampliamento alla costa Toscana . . . . .	25
<b>Conclusione</b>		<b>26</b>

# Introduzione

In questo elaborato viene analizzato il lavoro svolto nel tirocinio presso il Laboratorio di Telecomunicazioni e Telematica del Dipartimento di Ingegneria Informatica e dell'Informazione, durante il quale è stato realizzato un sistema di algoritmi in grado di elaborare ed analizzare dati geo-referenziati sulla qualità dell'acqua e la mappatura di fondali marini ai fini di produrre delle mappe di calore e grafici 3D. Inoltre è stato realizzato anche un portale accessibile pubblicamente dove è possibile consultare i risultati di questa ricerca.

L'obiettivo finale è quello di creare le fondamenta per lo sviluppo di un'applicazione automatizzata che permetta ad ogni istituzione, organismo o dipartimento di elaborare i dati raccolti e renderli di dominio pubblico.

I risultati fino ad ora ottenuti forniscono gli strumenti necessari ad una prima analisi e la possibilità di integrare tecnologie più avanzate.

Per quanto riguarda la parte di visualizzazione il risultato è un'intuitiva interfaccia che dà la possibilità di una rapida e accessibile consultazione della banca dati.

Il primo capitolo di questo elaborato parla delle tecnologie software di base che sono state integrate nel progetto.

Il secondo capitolo parlerà in dettaglio degli algoritmi di creazione delle mappe, quelli di visualizzazione e simulazione.

Il terzo capitolo oltre a parlare degli sviluppi futuri mostrerà come questo software sia in grado di interagire elaborando i dati rilevati da strumentazioni hardware, come per esempio un drone marino.

# Capitolo 1

## Analisi Delle Tecnologie Software Utilizzate

### 1.1 QGIS

QGIS (noto anche come Quantum GIS) è un'applicazione desktop GIS open source.

Con GIS (Geographic Information System) si intende un sistema informativo computerizzato che permette l'acquisizione, la registrazione, l'analisi, la visualizzazione e la restituzione di informazioni derivanti da dati geografici (geo-riferiti).

Il GIS è composto da una serie di strumenti software per acquisire, memorizzare, estrarre, trasformare e visualizzare dati spaziali dal mondo reale [1]. La versione utilizzata in questo progetto è "QGIS 2.18 'Las Palmas'" .

### 1.2 R

R è un linguaggio di programmazione e un ambiente di sviluppo specifico per l'analisi statistica dei dati.

R fornisce un'ampia varietà di strumenti statistici (modelli lineari e non, classificazione, clustering,...) e grafici.

È un software libero in quanto viene distribuito con la licenza GNU GPL [2].

## 1.3 Google Maps JavaScript API

Le Google Maps JavaScript API forniscono strumenti per creare applicazioni web in grado di unire i propri algoritmi con una solida e conosciuta piattaforma come Google Maps [3].

## 1.4 Database e Hosting utilizzati

Come database è stato utilizzato MySQL, open source database manager. Come hosting invece è stato usato il dominio dell'università di Siena: "*teamcoste.diism.unisi.it*".

## 1.5 Linguaggi di formattazione e programmazione web

### 1.5.1 HTML XML e CSS

L'HyperText Markup Language (HTML) è un linguaggio di markup utilizzato principalmente per il disaccoppiamento della struttura logica di una pagina web (definita appunto dal markup) e la sua rappresentazione, gestita tramite gli stili CSS.

Il CSS (Cascading Style Sheets, in italiano fogli di stile a cascata), in informatica, è un linguaggio usato per definire la formattazione di documenti HTML e XML.

L'XML (eXtensible Markup Language) è un metalinguaggio per la definizione di linguaggi di markup, ovvero un linguaggio marcatore basato su un meccanismo sintattico che consente di definire e controllare il significato degli elementi contenuti in un documento o in un testo [4] [5] [6].

### 1.5.2 PHP

Il PHP è un linguaggio di scripting server-side, utilizzato per applicazioni web ma anche per programmazione generale [7].

### 1.5.3 JavaScript

Il 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 [8].

### 1.5.4 AJAX

In informatica AJAX, acronimo di Asynchronous JavaScript and XML, è una tecnica di sviluppo software per la realizzazione di applicazioni web interattive. 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 [9].

### 1.5.5 SQL

L'SQL (Structured Query Language) è un linguaggio standardizzato che permette di interagire creare e modificare database [10].

# Capitolo 2

## Algoritmi di creazione, visualizzazione e simulazione

### 2.1 Creazione delle mappe

La creazione delle mappe si divide in tre parti, la creazione di uno ”ShapeFile”, l’elaborazione dei dati e il successivo caricamento dei risultati su una struttura dati.

#### 2.1.1 Creazione del ShapeFile

Uno ShapeFile è un formato vettoriale di registrazione di identità geometriche e delle loro informazioni associate.

Esso viene generato utilizzando QGIS; i passaggi che si va ad effettuare sono i seguenti:

**Creazione del file dati:** Quando uno strumento hardware raccoglie dei dati, esso deve salvarli seguendo un semplice standard così fatto:

---

```
1 lat,long,id,data1
2 10.2953219108,42.7564984134,0,0.5755136427
3 10.296876179,42.7560490387,1,0.3930092551
4 10.2962652569,42.7562753101,2,0.6320004668
5 10.2956698713,42.7559709335,3,0.7739448631
6 10.294642703,42.7559279058,4,0.1647003424
```

---



Figura 2.1: Punti geo-localizzati con layer mappa satellite.

- *lat, long*: rappresentano la latitudine e longitudine dei dati raccolti.
- *id*: è un indice univoco per ogni singola misurazione.
- *data1*: può essere rinominato a seconda della tipologia del rilevamento, rappresenta il valore numerico della misurazione del sensore.

Il file deve rispettare l'ordine dei dati e deve essere in formato *".csv"*, separato da *,* (comma).

**Import file dati su QGIS:** Il file dati deve essere poi importato su QGIS nel seguente modo: **Layer->AddLayer->AddDelimitedTextLayer**.

Una volta importato avremmo una serie di punti geo-localizzati, come si vede in figura: 2.1.

**Delimitazione area di contorno:** Il passo successivo è quello di delimitare l'area che questi punti occupano.

Utilizziamo il comando **Vector->ConcaveHull**, come si vede in figura: 2.2.

**Determinazione dell'area di ogni singolo punto:** L'idea ora è quella di delimitare ogni singolo punto inscrivendolo dentro a un poligono, la cui

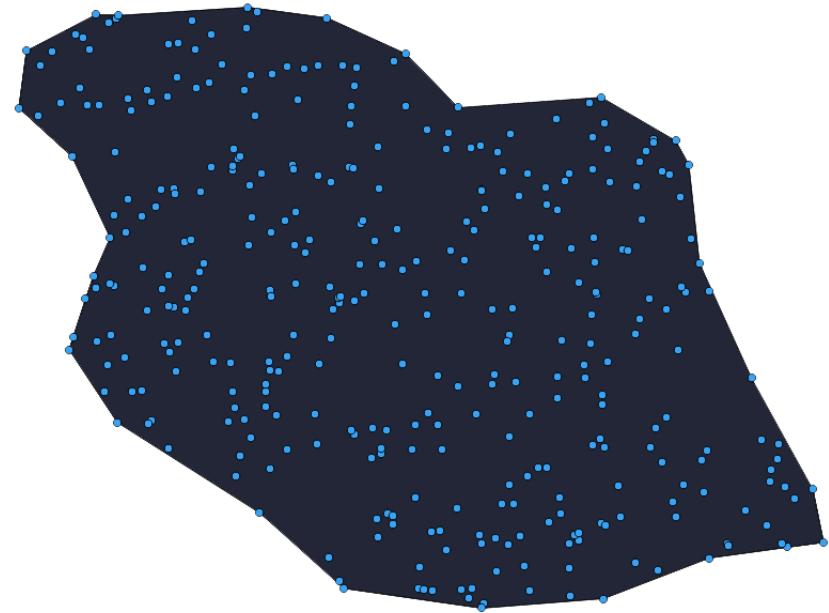


Figura 2.2: Concave Hull dell’insieme di punti importati.

area dipende dalla vicinanza di altri punti.

Più rarefatti sono i punti in una zona, più sarà grande l’area.

Utilizziamo il Diagramma di Voronoi, un particolare tipo di decomposizione di uno spazio metrico determinata dalle distanze rispetto ad un determinato insieme discreto di elementi dello spazio (ad esempio, un insieme finito di punti).

Utilizziamo il comando `Vector->GeometryTools->VoronoiPolygons`, come si vede in figura: 2.3.

**Intersezione dell’area interessata:** La tassellatura di Voronoi crea un rettangolo che esce fuori dalla zona misurata. Per questo ora intersechiamo l’area di contorno di tutti i punti e l’output appena ottenuto, così da avere una tassellatura solamente della regione interessata, come si vede in figura: 2.4.

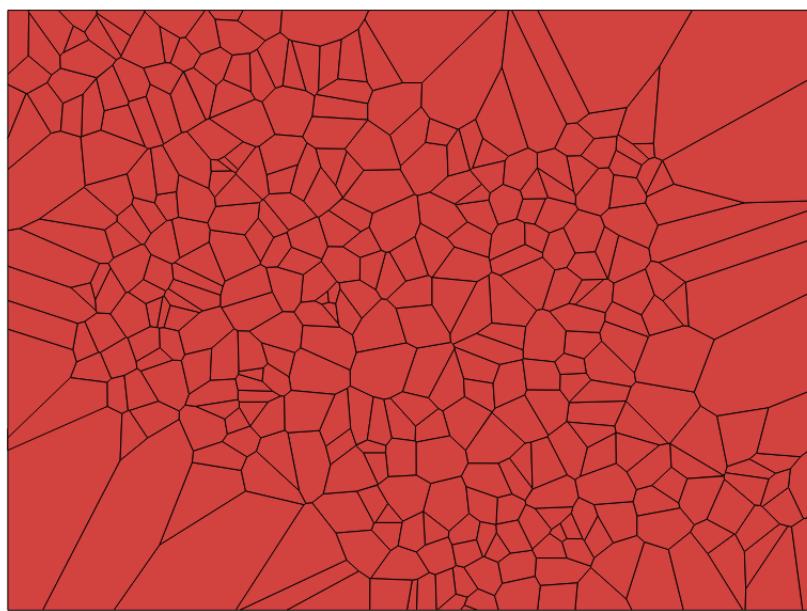


Figura 2.3: Diagramma di Voronoi dei punti importati.

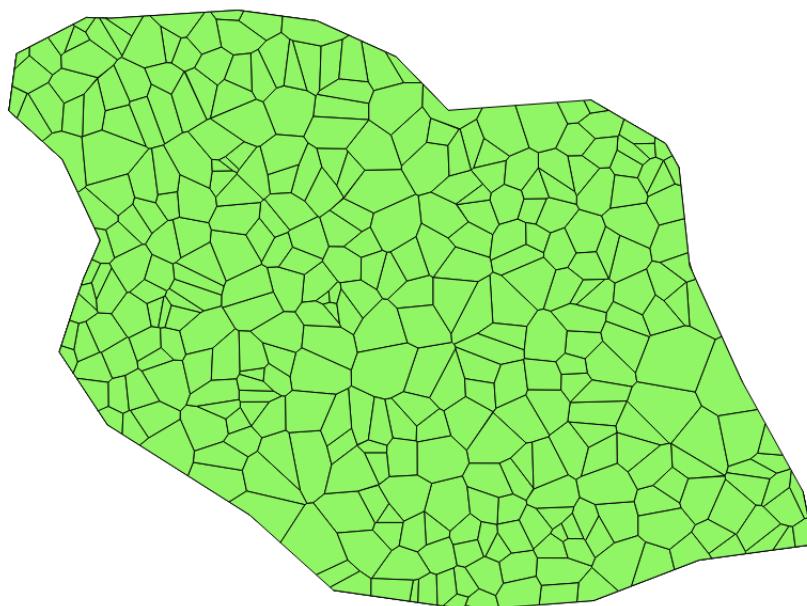


Figura 2.4: Intersezione tra Diagramma di Voronoi e Concave Hull.

**Esportazione in formato ShapeFile:** Ora siamo pronti per salvare il risultato ottenuto ed esportarlo in formato ShapeFile (\*.shp) così che può essere manipolato successivamente.

### 2.1.2 Elaborazione dei dati e creazione della mappa

Lo ShapeFile ora creato mantiene le proprietà di geo-localizzazione e la misura dei singoli dati.

Una volta importato manterrà le proprietà di geo-localizzazione e misura dei singoli dati.

Quindi l'algoritmo, per ogni area dello ShapeFile, andrà a colorarla in base al valore numerico della misurazione associata così creando una "Heat Map".

Generalmente più è intenso il colore è più alto sarà il valore.

Una "Heat Map" o mappa di calore è una rappresentazione grafica dei dati dove i singoli valori sono rappresentati da colori.

Questa parte è stata realizzata in R.

#### Import dello ShapeFile e creazione del DataFrame:

---

```

1 lnd <- readOGR("data/elba", "elba")
2 lnd <- (SpatialPolygonsDataFrame(Sr = spTransform(lnd,CRSobj =
    CRS("+init=epsg:4326")), data =lnd@data))
3 lnd.length<-length(lnd@data$id)
4 mydata<-data.frame(id=lnd@data$id, data1=lnd@data$data1)
5 lnd.f <- fortify(lnd,region = "id")

```

---

Con "readOGR" andiamo a importare una OGR Vector Map (nel nostro caso uno ShapeFile) in un Spatial Object.

Uno Spatial Object è una struttura dati che memorizza posizioni come oggetti (poligoni, linee e punti).

Con "SpatialPolygonsDataFrame" andiamo ad memorizzare in DataFrame "data.frame" gli attributi dei singoli poligoni: "latitudine, longitudine, id, data1".

Creiamo un secondo DataFrame ("mydata") nel quale copiamo "id, data1".

**Associazione univoca tra id e dato:**

---

```
1 lnd.f <- fortify(lnd,region = "id")
2 merge.shp.coef<-merge(lnd.f, mydata, by="id", all.x=TRUE)
3 final . plot<-merge.shp.coef[order(merge.shp.coef$order)]
```

---

Con questo codice rendiamo possibile il plotting della mappa tramite la libreria "ggplot2".

Inoltre ci assicuriamo che ad ogni "id" sia associato il dato corretto.

Salviamo tutto quanto nella struttura "final.plot".

**Creazione di un BoundingBox:**

---

```
1 b <- bbox(lnd)
```

---

Un BoundingBox è la scatola con la misura più piccola, di area o volume, entro cui sono contenuti tutti i punti.

Con questo otteniamo un set di quattro coordinate che contengono tutti i punti della mappa.

**Download mappa da internet:**

---

```
1 lnd.b1 <- ggmap(get_map(location = b,zoom = 12,scale="auto",maptype =
  "satellite"))
```

---

Questa funzione permette di scaricare da un server (Google Maps, OpenStreetMap, Stamen Maps o Naver Map) una sezione di mappa.

La sezione è individuata dal BoundingBox precedentemente creato.

E' possibile scegliere tra varie tipologie di mappe, tra cui: "terrain", "terrain-background", "satellite", "roadmap", "hybrid", "toner", "watercolor".

Come si può vedere nelle figure: 2.5, 2.6, 2.7, 2.8.

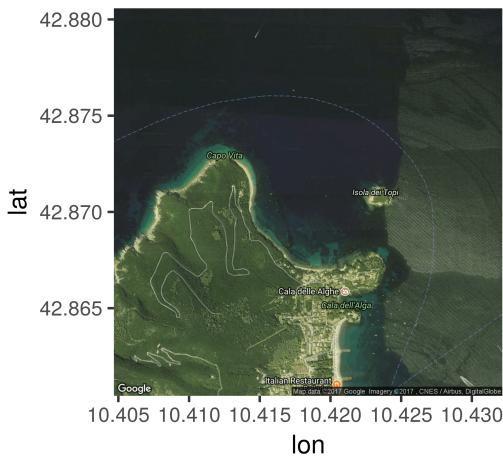


Figura 2.5: Hybrid Map

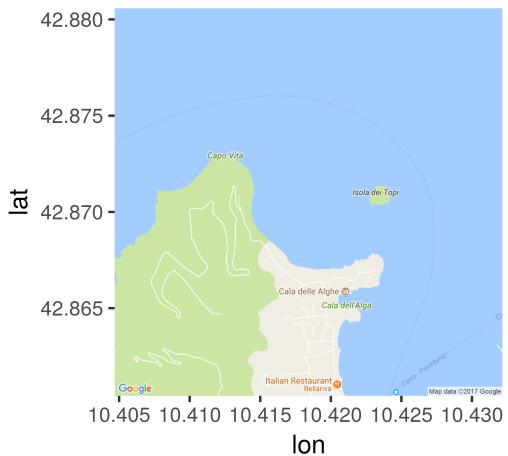


Figura 2.6: Rad Map

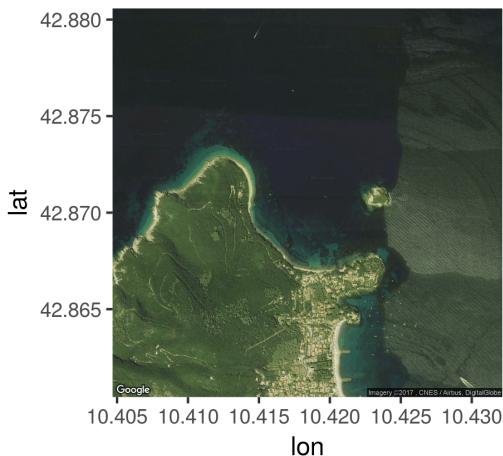


Figura 2.7: Satellite Map

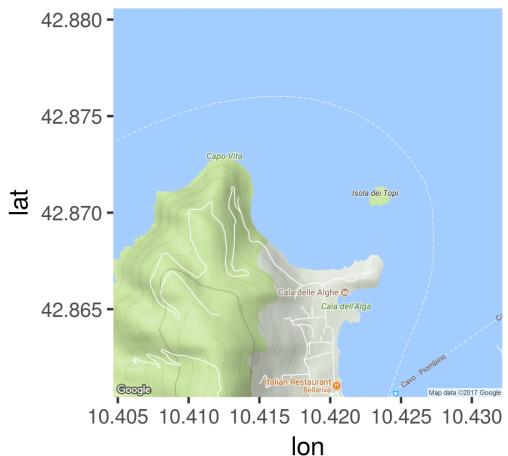


Figura 2.8: Terrain Map

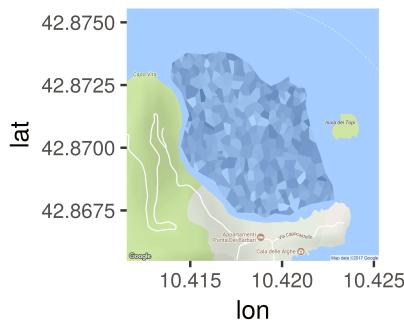


Figura 2.9: Gradazione 1

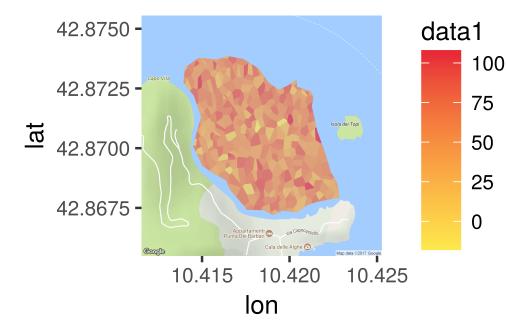


Figura 2.10: Gradazione 2

### Elaborazione e creazione della mappa:

---

```

1 lnd.b1 +
2 geom_polygon(data = final.plot, aes(x = long, y = lat, group = group, fill =
   data1), alpha = 0.5) +
3 scale_fill_continuous (low = "green", high = "red")

```

---

Con "geom\_polygon" possiamo disegnare su una mappa i poligoni memorizzati in "final.plot", andando a definire come parametri di geo-localizzazione "lat, long" e come valore "data1".

"scale\_fill\_continuous" permette di assegnare una gradazione di colore in base al parametro "fill" alla singola area.

E' possibile cambiare a piacimento la gradazione e l'intensità del colore. Come si vede nelle figure: 2.9, 2.10.

La mappa scaricata dal server e la mappa appena creata vengono unite insieme per creare un singolo file.

---

```

1 stat_density2d(data = final.plot, aes(x = long, y = lat), size = 0.6, colour
   = "black" ,n = 300)

```

---

E' possibile tramite stat\_density2d fare una 2D kernel density estimation, che permette di disegnare linee di densità in base al numero di dati in una certa zona, come si vede nella figura: 2.11 .

---

```

1 geom_point(data=final.plot, mapping=aes(long, lat),shape = 20,size = 0.1)

```

---

E' possibile tramite "geom\_point" disegnare i punti di intersezione fra le varie aree, come si vede nella figura: 2.12.



Figura 2.11: stat\_density2d



Figura 2.12: geom\_point

### Salvataggio dell'immagine:

---

```

1 img_name <- paste("map_resoultos/", Sys.time(), ".png")
2 ggsave (img_name, dpi = 2000)

```

---

Con questo codice salviamo l'immagine creata in un file ”.png”.

### 2.1.3 Salvataggio della misurazione nel database

---

```

1 mydb = dbConnect(MySQL(), user='user', password='password',
                   dbname='dbname', host='host')
2 mydata<-data.frame(name='nome_luogo', lat=final.plot$lat[1],
                      lng=final.plot$long[1])
3 dbWriteTable(mydb, name='placess', value=mydata)
4 mydata<-data.frame(name='nome_misurazione')
5 dbWriteTable(mydb, name='types', value=mydata)
6 mydata<-data.frame(id_place='id_place', id_type='id_type',
                      date='01/01/2017', url_img=img_name)
7 dbWriteTable(mydb, name='mesures', value=mydata)

```

---

Con questo codice andiamo ad accedere alla struttura dati e a memorizzare il nome del luogo con le rispettive coordinate, il tipo e la data della misurazione.

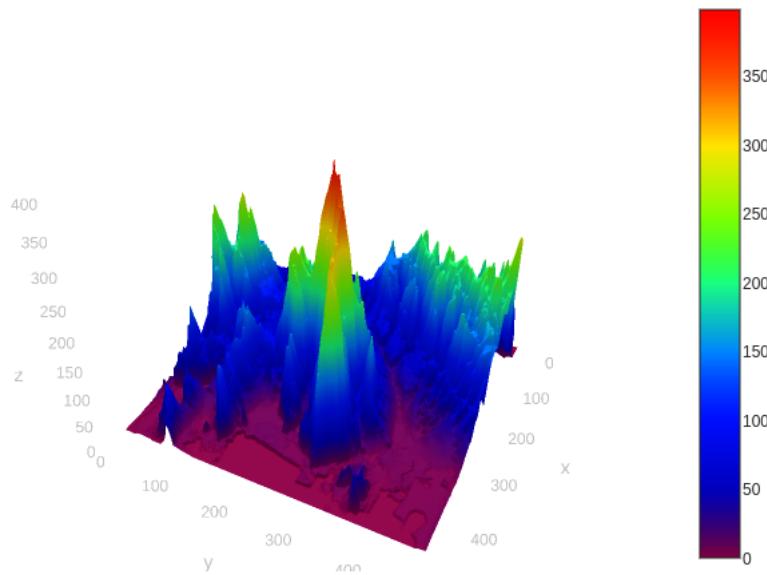


Figura 2.13: Esempio di grafico 3D generato.

#### 2.1.4 Grafici 3D

Per quanto riguarda i grafici 3D è stata usata la libreria "plotly". La logica è quella di una matrice  $n \times m$  dove per ogni singola coordinata è associato il valore numerico della misurazione.

$$M = \begin{bmatrix} 1.25 & 1.33 & \dots & 1.11 \\ 1.71 & 1.46 & \dots & 0.19 \\ \vdots & \vdots & \ddots & \vdots \\ 1.99 & 1.52 & \dots & 1.21 \end{bmatrix}$$

Ogni coordinata  $(i, j)$  corrisponde a una coordinata  $(lat, long)$ .

La libreria fa in modo che data questa matrice restituisca come output un grafico 3D interattivo suddiviso anch'esso in  $n \times m$  caselle e per ognuna di queste caselle sarà presente una "punta" di altezza e colorazione relativa alla misura ad essa associata.

E' possibile esportare il grafico sia come semplice immagine, sia come pagina ".html" interattiva.

Questo tipo di grafico è ideale per descrivere per esempio altitudini di montagne oppure profondità di fondali, come si vede in figura: 2.13.

### 2.1.5 Librerie usate

Per gli algoritmi di creazione delle mappe, sono state utilizzate le seguenti librerie:

- **library(plyr)**: per gestire i formati ShapeFile.
- **library(rgdal)**: per gestire dati geo-spatiali.
- **library(ggplot2)**: per la creazione della mappa, include anche le legende.
- **library(ggmap)**: per richiedere mappe online.
- **library(RMySQL)**: per accedere e manipolare il database.
- **library(plotly)**: per generare grafici 3D.

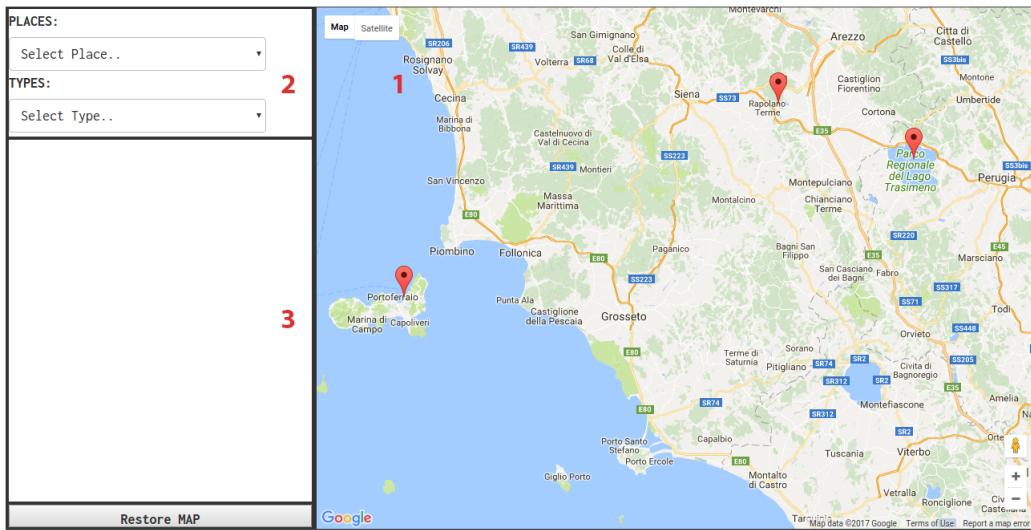


Figura 2.14: Sito web dove è possibile consultare i dati.

1: map\_container, 2: search\_container, 3: result\_container.

## 2.2 Visualizzazione delle mappe

Nella Figura: 2.14 è possibile vedere uno screenshot della pagina principale del portale internet creato per la visualizzazione dei dati.

### 2.2.1 Descrizione del portale

**map\_container:** mostra una mappa con un "position marker" per ogni luogo dove è stata fatta una misurazione.

L'utente ha la possibilità di cliccare sul marker e visualizzare il nome del luogo e le coordinate geo-spatiali.

Cliccando sul marker inoltre si innesca un evento che porta il nome del luogo sul search\_container, per una consultazione più intuitiva.

**search\_container:** mostra due form, da dove è possibile scegliere la località e successivamente i tipi di misurazione che sono stati effettuati.

La caratteristica principale di questo form è che ad ogni ricerca, accede al database e restituisce i dati contenuti in tempo reale.

Questo è stato possibile utilizzando la tecnologia AJAX.

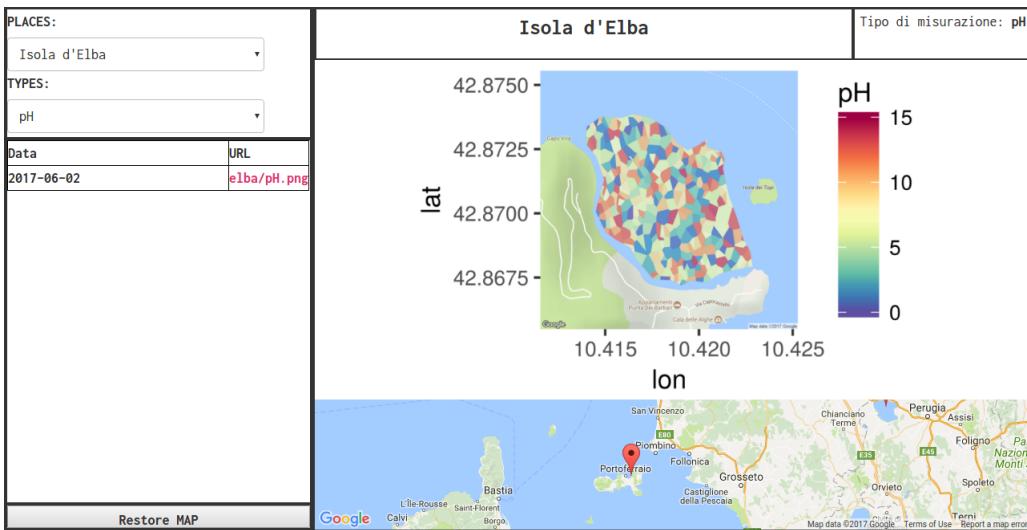


Figura 2.15: map\_results.

**result\_container:** mostra i risultati sul database con i parametri selezionati dal form precedente.

**map\_result:** mostra la mappa della misurazione appena cercata.

## 2.2.2 Descrizione degli script principali

### dbcontroller.php:

```

1 class DBController {
2     require('config/config.php');
3     function __construct() {
4         $this->conn = $this->connectDB();
5     }
6     function connectDB() {
7         $conn =
8             mysqli_connect($this->host,$this->user,$this->password,$this->database);
9         return $conn;
10    }
11    function runQuery($query) {
12        $result = mysqli_query($this->conn,$query);
13        while($row=@mysqli_fetch_assoc($result)) {
14            $resultset [] = $row;
15        }
16    }
17    function insertQuery($query) {
18        $result = mysqli_query($this->conn,$query);
19        if($result) {
20            return true;
21        } else {
22            return false;
23        }
24    }
25    function updateQuery($query) {
26        $result = mysqli_query($this->conn,$query);
27        if($result) {
28            return true;
29        } else {
30            return false;
31        }
32    }
33    function deleteQuery($query) {
34        $result = mysqli_query($this->conn,$query);
35        if($result) {
36            return true;
37        } else {
38            return false;
39        }
40    }
41    function selectQuery($query) {
42        $result = mysqli_query($this->conn,$query);
43        $rows = mysqli_num_rows($result);
44        if($rows > 0) {
45            $resultset = array();
46            while($row=@mysqli_fetch_assoc($result)) {
47                $resultset [] = $row;
48            }
49            return $resultset;
50        } else {
51            return false;
52        }
53    }
54}

```

---

```

14     }
15     if (!empty($resultset))
16         return $resultset ;
17     }
18 }
```

---

Questo script contiene una classe che ha il compito di recuperare i dati di accesso da un file di configurazione protettoe di gestire la connessione e le query al database.

### **initMap():**

---

```

1 function initMap() {
2     var map = new google.maps.Map(document.getElementById('map'), {
3         zoom: 8,
4         center: myLatlng,
5         mapTypeId: 'satellite'
6     });
7     var infoWindow = new google.maps.InfoWindow;
8 }
```

---

Questa funzione permette di richiamare da server Google una mappa, dato il tipo e la posizione di caricamento. La mappa viene restituita nel contenitore 'map'.

### **MySQLtoXML.php:**

---

```

1 $query = "SELECT * FROM places WHERE 1";
2 $results = $db_handle->runQuery($query);
3 header("Content-type: text/xml");
4 foreach( $results as $rows) {
5     $node = $dom->createElement("marker");
6     $newnode = $parnode->appendChild($node);
7     $newnode->setAttribute("id",$rows['id_place']);
8     $newnode->setAttribute("name",$rows['name']);
9     $newnode->setAttribute("lat", $rows['lat']);
10    $newnode->setAttribute("lng", $rows['lng']);
```

---

```

11 }
12 echo $dom->saveXML();

```

---

Questa pagina ha il compito di accedere al database e creare un file XML con le informazioni relative ai singoli luoghi dove sono state effettuate misurazioni.

---

```

1 $url = "scripts/MySQLtoXML.php"
2 downloadUrl($url, function(data) {
3     var xml = data.responseXML;
4     var markers = xml.documentElement.getElementsByTagName('marker');
5     Array.prototype.forEach.call(markers, function(markerElem) {
6         var id = markerElem.getAttribute('id');
7         var name = markerElem.getAttribute('name');
8         var point = new google.maps.LatLng(
9             parseFloat(markerElem.getAttribute('lat')),
10            parseFloat(markerElem.getAttribute('lng'))));
11     })

```

---

Il risultato di questa operazione è preso dal continuo della funzione ”initMap()” che provvede al posizionamento dei singoli marker nella mappa.

### **getTypes():**

---



---

```

1 function getTypes(val) {
2     $.ajax({
3         type: "POST",
4         url: "scripts/getTypes.php",
5         data:'id_place='+val,
6         success: function(data){
7             $("#typesList").html(data);
8         }
9     });
10 }

```

---

Questa funzione ha il compito di prendere il valore del form ”typesList” e di inviarlo tramite AJAX alla pagina ”getTypes.php”.

La funzione è richiamata ogni volta che il valore del form cambia.

---

```
1 if (!empty($_POST["id_place"])) {  
2     $id_place = $_POST["id_place"];  
3     $query =  
4         "  
5             SELECT DISTINCT types.name, types.id_type  
6             FROM types,mesures,places  
7             WHERE mesures.id_place = places.id_place  
8             AND mesures.id_type = types.id_type  
9             AND places.id_place = ".$id_place."  
10            ";  
11     $results = $db_handle->runQuery($query);  
12 }
```

---

La pagina "getTypes.php" controlla che ci sia stata effettivamente una scelta nel form precedente, in caso affermativo allora salva in una variabile il valore che gli è stato passato ed effettua una query al database.

I risultati sono salvati in un array multidimensionale e sono pronti per essere stampati nella formattazione corretta.



Figura 2.16: 300 Random Points.

Figura 2.17: 1000 Random Points.

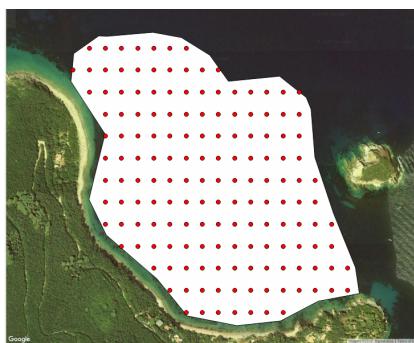


Figura 2.18: Regular Points.

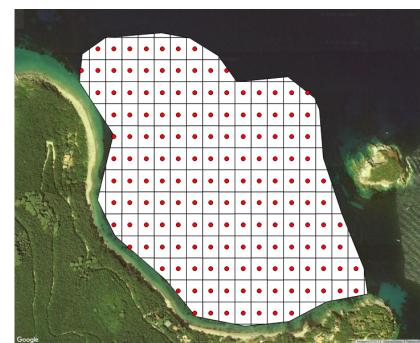


Figura 2.19: Suddivisione regolare.

## 2.3 Simulazione utilizzando dati auto-generati

Ai fini di provare l'effettivo funzionamento del codice, sono state implementate alcune funzioni per poter simulare i dati di una misurazione.  
Le simulazioni sono effettuate su QGIS.

### 2.3.1 Random points inside a polygon

Tramite questa funzione è possibile simulare, dato uno ShapeFile di contorno, una serie di punti casuali al suo interno.

Si può osservare nelle figure: 2.16, 2.17.

### 2.3.2 Regular points inside a polygon

Tramite questa funzione è possibile simulare, dato uno ShapeFile di contorno, una serie di punti a distanza regolare l'uno dall'altro.

Questo si può osservare nelle figure: 2.18, 2.19.

# Capitolo 3

## Applicazioni e sviluppi futuri

### 3.1 Drone marino

La realizzazione di questo progetto è andata di pari passo con un altro lavoro di tesi [11] che ha contribuito alla realizzazione di un drone marino in grado di navigare ed effettuare misurazioni sulla qualità dell'acqua e sulla morfologia dei fondali marini registrandone la posizione tramite GPS.

Le misurazioni possibili sono le seguenti:

- Ossigeno dissolto
- Salinità
- pH
- Velocità dell'acqua
- Potenziale di ossidriduzione
- Temperatura dell'acqua
- Profondità marina

Una volta raccolti i dati il sistema è in grado di renderli pronti per essere analizzati e caricati sulla struttura dati, come si vede nelle figure: 3.1, 3.2.

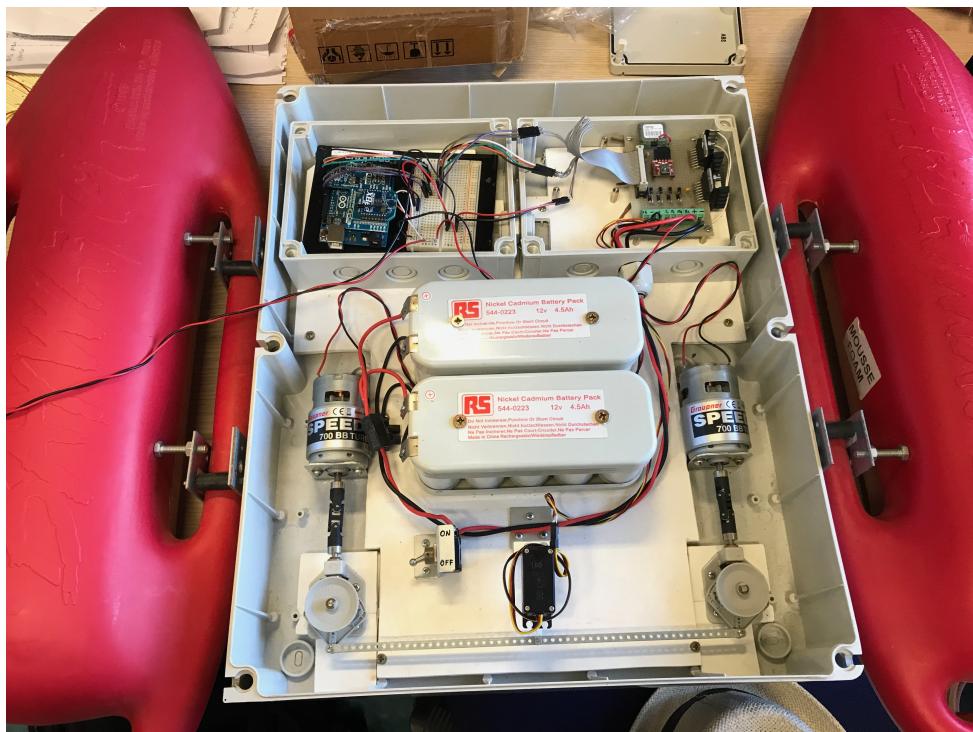


Figura 3.1: Immagine del drone in progettazione.

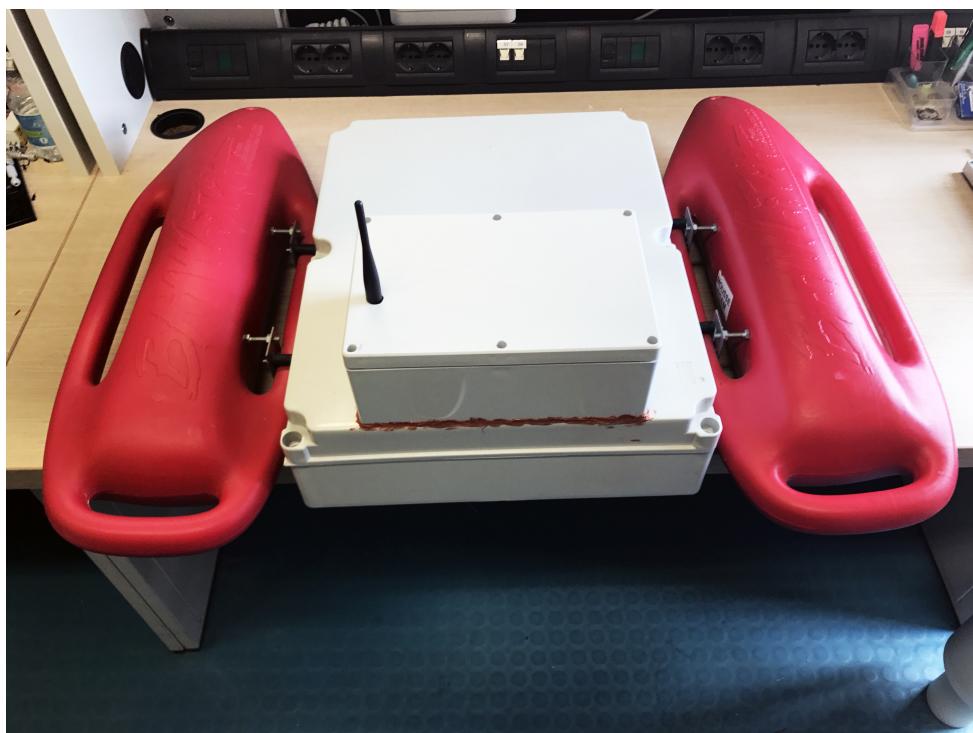


Figura 3.2: Immagine del drone in fase di montaggio.

## 3.2 Ampliamento alla costa Toscana

Uno degli obiettivi di questo progetto è quello di creare uno standard per la raccolta dei dati e creare la base per una struttura dati dove qualunque istituzione possa elaborare i propri dati e renderli pubblici.

Purtroppo in questo momento la raccolta dei dati non è uniforme fra le varie istituzioni e questo rende il lavoro di elaborazione e pubblicazione molto più difficile da automatizzare.

Istituzioni come ARPAT (Agenzia regionale per la protezione ambientale della Toscana) offrono servizi di visualizzazione datati non geo localizzati, quindi è impossibile sapere esattamente la posizione del rilevamento del dato e di conseguenza limita alcuni ambiti di ricerca.

Così con questo progetto si vuole stimolare una collaborazione tra le varie istituzioni per arrivare ad uno standard universale.

# Conclusione

Durante questo progetto di tesi è stato progettato e realizzato uno strumento semplice ma efficace di elaborazione e visualizzazione di dati.

Questo progetto vuole essere una base per lo sviluppo futuro di questa applicazione integrando nuovi metodi di elaborazione e nuove tecnologie e rendendo l'esperienza della visualizzazione delle misurazioni semplice ma dettagliata.

Quindi riuscire ad creare un portale con informazioni dettagliate e facili da visualizzare della costa Toscana collaborando con altre istituzioni e università potrebbe essere un punto di partenza per la creazione di un servizio territoriale che renda possibile la visualizzazione dei dati già presenti che non sono ancora visualizzabili.

# Bibliografia

- [1] "*QGIS Map Design*", Anita Graser and Gretchen N Peterson.
- [2] "*Advanced R*", Hadley Wickham.
- [3] "*JavaScript: The Good Parts*", Douglas Crockford.
- [4] "*HTML and CSS: Design and Build Websites*", Jon Duckett.
- [5] "*Learning XML, Second Edition*", Erik T. Ray.
- [6] "*CSS and Documents*", Eric A. Meyer.
- [7] "*PhP: Learn PHP Programming*", Troy Dimes.
- [8] "*The Principles of Object-Oriented JavaScript*", Nicholas C. Zakas.
- [9] "*The Ajax Protocol*", Alex Lukeman.
- [10] "*Sql Guide* ", Inc. BarCharts.
- [11] "*Realizzazione di un drone marino per l'acquisizione di parametri morfologici e della qualità dell'acqua.*" Simone Garuglieri.

# Ringraziamenti

Voglio ringraziare in primo luogo la mia famiglia che mi ha sempre sostenuto nei momenti di difficoltà, ricordandomi sempre quali sono i valori fondamentali della vita. In particolare mio padre l'uomo a cui mi ispiro.

Il professor Alessandro Pozzebon per avermi seguito ed aiutato nello sviluppo di questa tesi ed il personale correlato al Laboratorio di Telecomunicazioni e Telematica.

Il professor Simone Giuliani, che negli anni delle scuole superiori mi ha trasmesso passione e conoscenze che ho fatto mie per sempre.

Il professor Duccio Papini, che nei momenti di necessità è sempre stato disponibile rivelandosi un professore ed una persona eccezionale. Un particolare ringraziamento va ai miei compagni di studi: Gianfranco, Giovanni, Antonella, Renato e tutti gli altri, con i quali abbiamo affrontato insieme le difficoltà del mondo universitario.

In particolare Mihai, che mi ha insegnato che ogni difficoltà si supera lavorando duramente.

Simone il miglior compagno di studi ed uno degli amici più preziosi che io abbia.

Margherita e Mattia che all'inizio di questo percorso di studi mi hanno aiutato e consigliato.

Matteo ed Antonio che oltre ad avermi motivato sempre a dare il meglio di me, mi hanno accompagnato in avventure in giro per il mondo rendendoli per me come una seconda famiglia.

Infine a Maëva che in ogni secondo mi ha dato sostegno, coraggio ed amore senza il quale la mia vita non sarebbe stata così felice.