



Università
Ca'Foscari
Venezia

DIPARTIMENTO DI SCIENZE AMBIENTALI,
INFORMATICA E STATISTICA

Corso di Laurea in Informatica

Tesi di Laurea

**Il tempo: meteorologico e umano.
Illustrazione e sviluppo di un'applicazione software,
tra nuove competenze e infiniti tempi di attesa**

Relatrice
Prof.ssa Alessandra Raffaetà

Laureanda
Giulia Cogotti
Matricola 884383

Anno Accademico
2023-2024

Abstract

Il presente documento si occupa di analizzare lo sviluppo di un'applicazione Android/iOS, commissionata da ISMAR-CNR e prodotta dall'azienda Elan42, per la visualizzazione di dati meteorologici provenienti da stazioni, radar e satelliti. L'applicazione raccoglie i dati da diverse sorgenti, pubbliche e private. Ed è in questa fase che iniziano i problemi. Gli enti privati che gestiscono i dati tendono a non collaborare tra loro e a custodirli segretamente. Questo compromette il processo evolutivo del software e spiega come mai lo sviluppo proceda lentamente e non si sia ancora giunti alla creazione di una versione stabile dell'applicazione. Per i dati pubblici non risultano esserci particolari criticità ed è stato solo necessario comprenderne la tipologia. Questi dati, quando accessibili, vengono elaborati e inviati su richiesta dell'utente al lato grafico dell'applicazione che li visualizza. Tutto questo lavoro viene supportato da politiche di riuso del codice che consentono di creare modelli generali per l'importazione dei dati e per l'invio e la visualizzazione di essi. Inoltre un database locale aiuta nella memorizzazione delle informazioni per rendere efficiente l'accesso ai dati da parte dell'applicazione mobile. Una volta che saranno superati gli ostacoli, l'applicazione entrerà finalmente in produzione e, al rilascio della prima versione, sarà utilizzabile da qualsiasi utente!

Indice

Abstract	i
1 Introduzione	1
2 Applicazione	7
2.1 Funzionalità	7
2.2 Architettura	11
2.3 Sorgenti di dati	15
2.3.1 Stazioni osservative fisse	16
2.3.2 Osservazioni da satellite	25
2.3.3 Modelli di previsione	28
3 Sviluppo	39
3.1 Strumentazione	39
3.2 Importazione dati	47
3.3 Costruzione API	61
4 Conclusioni	73
Bibliografia	75
Ringraziamenti	83

Capitolo 1

Introduzione

Questa tesi si occupa di analizzare lo sviluppo, dall'architettura sottostante al prodotto presentato agli utenti, di un'applicazione mobile realizzata durante il periodo di stage dell'autrice, svolto presso l'azienda Elan42. L'applicazione è commissionata dall'Istituto di Scienze Marine del Consiglio Nazionale delle Ricerche (ISMAR-CNR), un ente pubblico esterno all'azienda.



Figura 1.1: Logo Elan42

Elan42 è una Digital Agency – *agenzia digitale* – con sede a Venezia. Essa fornisce servizi di hosting di alta qualità promuovendo alternative sostenibili con lo scopo di ridurre l'inquinamento digitale. Inoltre si occupa della realizzazione di siti web professionali. Elan42 reputa fondamentale il concetto di codice aperto – *open source* – come strumento di condivisione, conoscenza e trasparenza¹.

Il software open-source differisce dagli altri software perché ha un contratto di licenza meno restrittivo: invece di utilizzare una licenza restrittiva che impedisce di modificare il programma o condividerlo, ne viene

¹E42 srl. *Sito web Agenzia Elan42 (Online)*, <https://www.elan42.com/>.

incoraggiata la condivisione e la modifica del software. Chiunque lo desideri può distribuire, modificare o addirittura creare opere derivate basate su quel codice sorgente². Quindi no, open-source non significa solo che si ha libero accesso al codice sorgente³.

Un aspetto interessante è che Elan42 si rende disponibile alla realizzazione di vari progetti, anche se essi si collocano al di fuori dal proprio ambito specialistico. In particolare, il progetto che verrà discusso nel presente documento, ideato e commissionato da un cliente esterno all'agenzia, presenta un argomento non conforme a quelli tipici dell'azienda.



Figura 1.2: Logo ISMAR-CNR

ISMAR-CNR è il cliente che ha commissionato il progetto trattato. Esso rappresenta la più grande istituzione italiana che si occupa dello sviluppo scientifico nel campo della scienza dell'oceano⁴.

Il progetto consiste nello sviluppo di un'applicazione come nuova progettazione di una precedentemente creata⁵ ormai obsoleta e non più funzionante, che consenta la visualizzazione di dati meteorologici forniti da ISMAR-CNR.

L'applicazione raccoglie i dati da stazioni, radar e satelliti. Queste sorgenti di dati rappresentano il cuore dell'applicazione, senza i dati l'applicazione non può esistere. Un problema che comprometterà il processo evolutivo del software, nonché della tesi stessa, è il fatto che non sempre questi dati risultano disponibili. In particolare alcuni di essi sono gestiti da enti privati che non permettono l'utilizzo dei dati raccolti. Quando i dati sono disponibili vengono elaborati e inviati al lato grafico dell'applicazione che li visualizza in base alle esigenze dell'utente finale. Tutto questo lavoro viene supportato da politiche di riuso del codice che consentono di creare modelli generali per l'importazione dei dati e per l'invio e la visualizzazione di

²Diffingo Solutions Inc. *What is Open Source Software (Online)*, <https://web.archive.org/web/20081028104313/http://www.diffingo.com/oss/whyoss>.

³Open Source Org. *The Open Source Definition (Online)*, <https://web.archive.org/web/20070611152544/https://opensource.org/docs/osd>.

⁴ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/>.

⁵Ivi, <https://www.ismar.cnr.it/terza-missione/app/#2>.

essi. Inoltre un database locale aiuta nella memorizzazione delle informazioni per rendere efficiente l'accesso ai dati da parte dell'applicazione mobile.

Una volta terminata, l'applicazione entrerà finalmente in produzione e, al rilascio della prima versione, sarà utilizzabile da qualsiasi utente. Essa infatti si rivolge sia a un pubblico generale ma anche, in una certa misura, a un pubblico più specializzato ed esperto. Questo perché alcuni dati trattati sono utili nella vita quotidiana (per es. temperatura e vento) altri invece non sono di particolare interesse al di fuori dell'ambito specialistico (per es. clorofilla). Infine l'applicazione dovrà funzionare sia su sistemi Android che Apple iOS (desktop, mobile e tablet, dando precedenza alla versione mobile) e dovrà essere scaricabile dai rispettivi negozi online.

Da ora in avanti ci si riferirà all'applicativo creato equivalentemente con i termini *Ismar Data*, *applicazione*, e *progetto*.

Metodologia di lavoro. L'applicazione è sviluppata da un team. Quando un gruppo di persone deve collaborare per portare a termine un compito, occorre stabilire il metodo con il quale si lavora.

Il modo tradizionale di sviluppare un progetto software è il modello **waterfall**. Esso consiste nella suddivisione in rigide fasi svolte in sequenza; quando una fase finisce, inizia la successiva. Non consentendo agli sviluppatori di tornare indietro e rivedere le fasi già conclusive (da qui il nome waterfall - cascata -)⁶. L'alternativa più innovativa, nonché quella utilizzata per il progetto, è il **metodo agile**⁷. Esso consente una gestione del progetto in un modo collaborativo, nel quale il team consegna il prodotto a piccoli passi, dando la possibilità al cliente di effettuare cambiamenti. Inoltre permette di fermarsi e tornare alle fasi precedenti qualora si presenti la necessità di effettuare miglioramenti⁸. In realtà la metodologia agile è un insieme di valori e principi descritti nel manifesto redatto da K. Beck et al⁹. Ogni metodo che segue quei fondamenti è ritenuto agile¹⁰.

In particolare, nello sviluppo di *Ismar Data*, il framework che si adatta meglio alla situazione è **scrum**. È importante osservare che, data l'imprevedibilità e la poca certezza presente nello sviluppo dell'applicazione dovuto al problema della mancanza dei dati, il metodo descritto non viene applicato alla lettera. Il metodo scrum si basa su cinque valori fondamentali: «*Impegno, Focus, Apertura, Rispetto*

⁶E. Conrad. «Chapter 8 - Domain 8: Application Development Security». A cura di E. Conrad. Boston: Syngress, p. 131.

⁷Il termine agile è pronunciato all'inglese.

⁸K. Waters. *All about Agile: Agile Management Made Easy!* allaboutagile.com, p. 7.

⁹K. Beck et al. *Manifesto for agile software development*. Snowbird, UT.

¹⁰Waters, cit., p. 10.

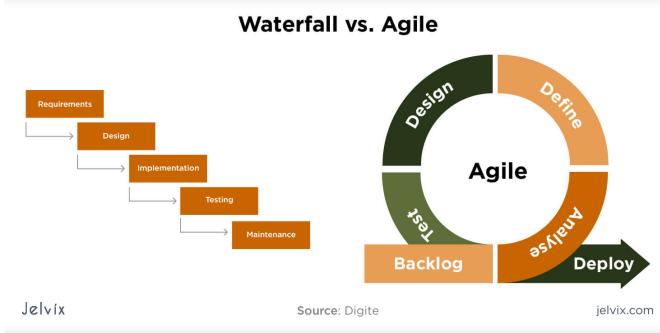


Figura 1.3: Waterfall vs. Agile^a.

^aFonte: Jelvix. *Waterfall vs. Agile (Immagine)*.

e Coraggio»¹¹. Il lavoro viene suddiviso in «sprint» della durata di due settimane, durante le quali ogni sviluppatore – developer¹² – svolge il compito che gli è stato assegnato. I compiti sono prelevati da un «backlog¹³», una lista contenente tutte le richieste del cliente in ordine di importanza. Non è detto che vengano effettivamente implementate tutte le funzionalità richieste, soprattutto nelle prime versioni dell'applicazione.

La gestione dei compiti viene fatta utilizzando l'applicativo **Asana**¹⁴. Esso è uno strumento di gestione del lavoro che, grazie alle sue funzionalità di coordinazione del team e monitoraggio delle attività¹⁵, facilita l'utilizzo del metodo scrum.

Perché continuare a leggere? I successivi capitoli analizzeranno l'applicazione, dando particolare rilevo alla descrizione delle sorgenti di dati e all'architettura sottostante, con un'attenzione alle problematiche presenti e alle possibili soluzioni. Ulteriore spazio sarà dato allo sviluppo concreto del progetto, esaminando nel dettaglio gli strumenti utilizzati, sia per la comunicazione che per la produzione, le funzionalità implementate e la gestione dei dati disponibili. Infine si trarranno le conclusioni sull'intero percorso, verificando se l'obiettivo è stato raggiunto oppure se qualcosa

¹¹K. Schwaber e J. Sutherland. *La guida Scrum. La Guida Definitiva a Scrum: Le Regole del Gioco*, p. 4.

¹²Ivi, pp. 5–6.

¹³Ivi, p. 11.

¹⁴Asana, Inc. *Asana: un modo più intelligente di lavorare (Online)*, <https://asana.com/it>.

¹⁵Ivi, https://help.asana.com/hc/it/articles/14250783001627-Come-iniziare-a-usare-Asana#h_01HEQV7CDGXHBY06W9PQAXB8RK.

Confronto grafico tra le due filosofie di lavoro, *waterfall* a sinistra e *agile* a destra. Si può notare come il metodo *agile* risulta essere più flessibile e quindi più adatto allo sviluppo di un software.

non ha funzionato come avrebbe dovuto. Quindi, ci si trova davanti ad una scelta: andare avanti o restare fermi. Andare avanti non sempre significa arrivare al traguardo, ma l'importante è provarci anche se ad ogni passo avanti ne corrispondono due indietro.

Capitolo 2

Applicazione

In questo capitolo verrà trattata *Ismar Data*. Inizialmente saranno analizzate le funzionalità offerte all'utente finale. Un mockup grafico, ovvero una serie di immagini create appositamente per rappresentare le schermate dell'applicazione, aiuterà nel mostrare i risultati che si vorrebbero ottenere tramite l'implementazione delle diverse funzionalità. In seguito verrà introdotta l'architettura ideale dell'applicazione; ideale perché, nel corso di questo percorso, si noterà che alcuni intoppi renderanno difficile rispecchiarla nella realtà. Infine sarà dato ampio spazio alle sorgenti di dati, descrivendole e analizzando i dati che esse forniscono e le modalità con le quali ci si può accedere. Questa analisi finale porterà alla luce il problema della mancanza dei dati e della conseguente difficoltà a procedere nella produzione.

2.1 Funzionalità

L'applicazione dovrà fornire una serie di funzionalità, sia per quanto riguarda l'utilizzo vero e proprio dell'applicazione da parte dell'utente finale (*front-end*), che per la parte sottostante dell'applicazione (*back-end*). Inizialmente il cliente richiede un certo numero di funzionalità che desidera vedere implementate, nelle sezioni successive saranno infatti descritte. È chiaro che, dato il tempo limitato e la necessità di produrre qualcosa di concreto, le prime versioni del software avranno solamente le funzionalità ritenute prioritarie. Questa scala di priorità si affina quando, procedendo nello sviluppo, si giunge alla conclusione di uno sprint.

Il **front-end** è la parte grafica dell'applicazione che l'utente utilizzerà dal suo dispositivo. Nonostante l'implementazione sia affidata ad un team esterno, risulta necessario approfondirla in quanto gli sviluppatori del back-end devono sapere quali dati e risorse dare al front-end per un corretto funzionamento del prodotto. In par-

ticolare, non essendo ancora stata avviata la produzione del front-end, si farà uso di un *mockup* grafico, basato sulla precedente applicazione¹. Esso viene creato appositamente per dare un'idea di come dovrebbe presentarsi l'applicazione quando sarà terminata in modo da aiutare gli sviluppatori ad avere un obiettivo chiaro. In aggiunta, per presentare al cliente i vari progressi, viene sviluppata una *demo web*². Essa sarà usata, come valida alternativa all'applicazione vera e propria, nei successivi capitoli quando si analizzerà il lavoro fatto.

L'applicazione, come illustrato in Figura 2.1, visualizza una mappa, navigabile, nella quale compaiono alcuni cerchi numerati ad indicare la posizione delle stazioni osservative fisse. A questo si aggiunge, nella stessa schermata, la possibilità di visualizzare le stazioni sotto forma di elenco. Ogni località, se cliccata, visualizza una nuova finestra contenente i suoi dati meteorologici sotto forma di *widget*³ e grafici, sulla base di cosa sarà necessario visualizzare. Ad esempio nella terza e quarta immagine si possono osservare alcuni widget relativi alla *Piattaforma Acqua Alta* che indicano, in ordine, la velocità e direzione del vento, il tempo climatico (in questo caso nuvoloso), la temperatura atmosferica in un intervallo di tempo prefissato. Inoltre l'utente ha la possibilità di selezionare, tramite appositi riquadri, i dati della stazione che vuole visualizzare.

È presente un menu, in Figura 2.2, contenente le seguenti opzioni che si occupano di visualizzare schermate differenti a seconda della preferenza scelta:

- *Mappa*: mappa delle stazioni osservative fisse.
- *Modelli di previsione*: mappe con le previsioni di onde e venti prodotte dai sistemi Nettuno ed Henetus.
- *Osservazioni da satellite*: mappe contenenti i dati di clorofilla e temperatura superficiale del mare rilevate dai satelliti.
- *Radar*: mappa contenete i dati rilevati dai radar.
- *Info*: informazioni generali sull'applicazione.
- *Webcam*: immagini prese dalle webcam poste sulle stazioni osservative fisse.

¹ISMAR-CNR, cit., <https://www.ismar.cnr.it/terza-missione/app/#2>.

²**demo** <démë> s. ingl. [abbrev. di demonstration «dimostrazione»] (pl. demos <démëʃ>), usato in ital. al masch. o al femm. (e comunem. pronunciato <démö>). – In informatica, versione dimostrativa di un programma, presentato per lo più, a scopo di campagna pubblicitaria, in forma semplificata e parziale; Treccani S.P.A. *Demo*. In *Vocabolario Treccani online*.

³Con il termine *widget* si intendono dei riquadri grafici, con cui l'utente può interagire, contenenti dati e altre informazioni.

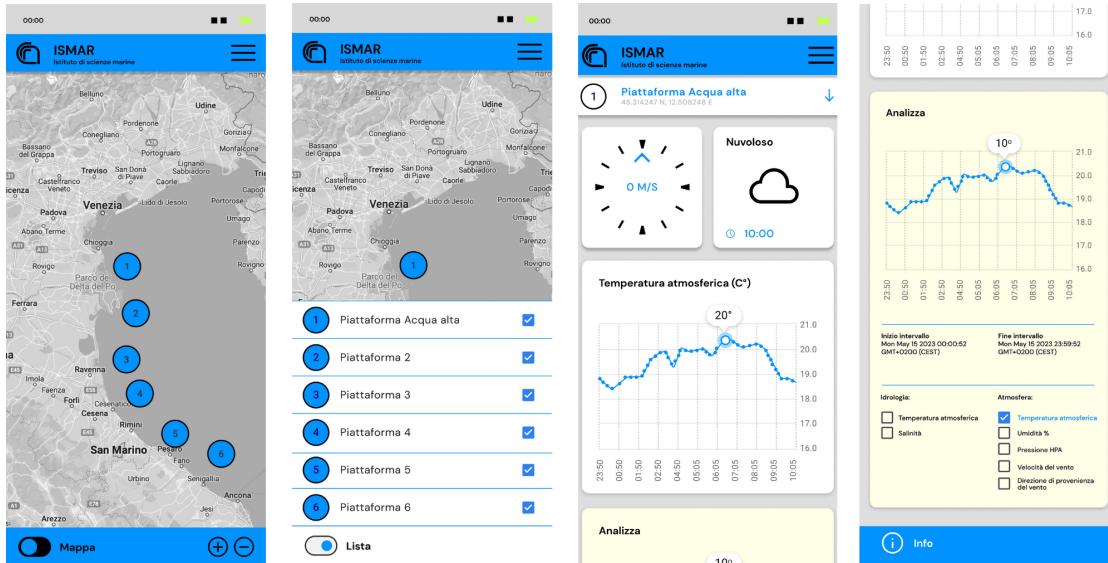


Figura 2.1: Schermate stazioni osservative fisse: mappa, lista e *widget*.^a.

^aFonte: mockup grafico.

L'esempio continua mostrando, nelle schermate successive al menu, la mappa con le frecce ad indicare rispettivamente direzione del vento e altezza d'onda rilevate dal modello di previsione Nettuno. Anche qua tramite appositi riquadri è possibile selezionare alcuni filtri, come l'area geografica di interesse. Infine è presente una breve descrizione del sistema visualizzato per spiegare di cosa si occupa Nettuno.

Le informazioni dettagliate su quali dati visualizzare per ogni sorgente saranno discusse nel corso del documento. Al momento, per le finalità del presente paragrafo, non risulta necessario affrontare quell'argomento.

Per quanto riguarda le webcam, osservabili in Figura 2.3, esse sono un esempio di funzionalità desiderata ma non implementata. Il motivo è semplice, inizialmente il cliente la considerava una funzionalità fondamentale ma, con il passare del tempo, si è reso conto che ci sono altre priorità. Quindi le webcam restano ma solo come funzionalità da aggiungere nelle versioni future dell'applicazione.

Il **back-end** è il lato che si occupa di reperire i dati, elaborarli e passarli al front-end. Esso fornisce funzionalità di importazione dei dati provenienti dalle diverse sorgenti (stazioni, radar e satelliti) nel modo più generale possibile, favorendo

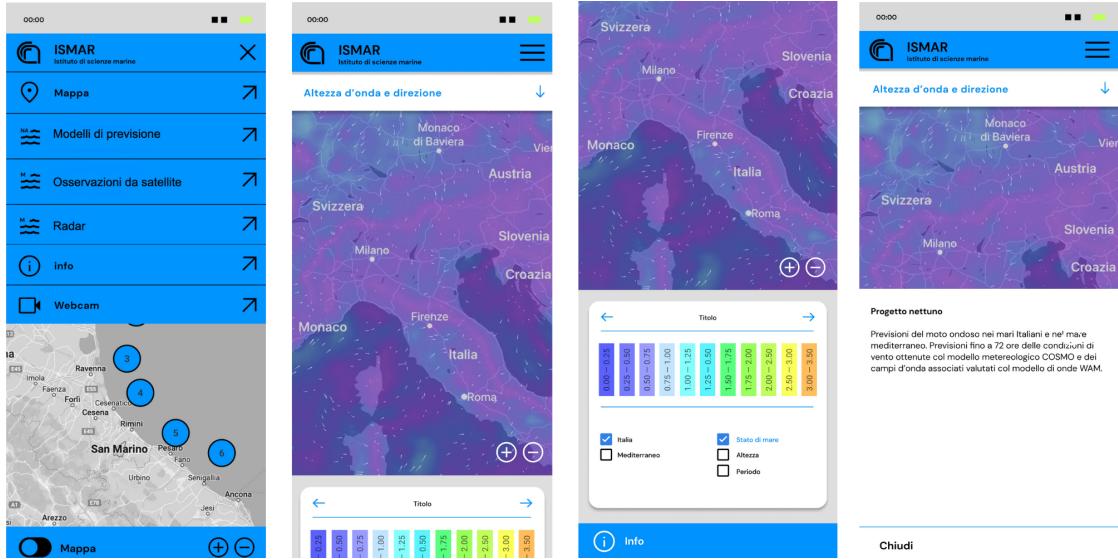
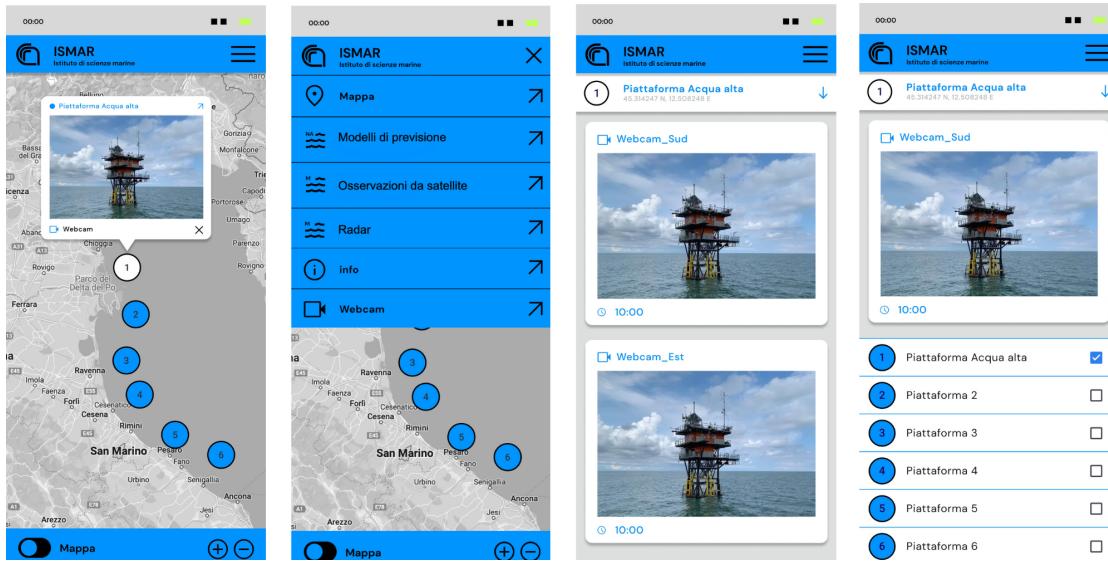


Figura 2.2: Schermate menu principale e previsioni onde e vento Nettuno^a.

^aFonte: mockup grafico.

il riuso del codice. I dati importati vengono poi elaborati e semplificati in modo da essere passati al front-end senza che esso abbia necessità di compiere operazioni complicate, le quali diminuiscono la velocità dell'applicazione. Devono essere poi implementati meccanismi per l'aggiornamento dei dati, mantenendoli per un determinato periodo di tempo e di conseguenza eliminando le informazioni meno recenti. Inoltre dovrà essere creato un database per il salvataggio dei dati provenienti dalle varie sorgenti e per altre informazioni utili al funzionamento dell'applicazione. Quindi dovranno essere esposte delle API che si occupano di reperire informazioni dal database locale e di passarle al front-end per la visualizzazione.

L'applicazione, una volta conclusa e approvata dal cliente, dovrà essere disponibile a tutti gli utenti. Questo significa che potrà essere scaricata dai principali marketplace – *negozi online* – esistenti: **Google Play Store** e **Apple Store**. Questo rientra nelle funzionalità perché, nello sviluppo, è necessario avere in mente dove si vuole arrivare. Quindi l'obiettivo è creare un'applicazione mobile disponibile per tutti gli utenti, a prescindere dal loro sistema operativo.

Figura 2.3: Schermate webcam.^a.

^aFonte: mockup grafico.

2.2 Architettura

Lo sviluppo di una grande applicazione necessita di avere un’architettura base nella quale viene definito ogni singolo componente e la relazione con gli altri, sia per separare logicamente il lavoro che per assegnarlo a diversi membri del team in modo che possano procedere allo sviluppo nello stesso momento.

Lo schema presente in Figura 2.4 mostra i diversi elementi che compongono l’infrastruttura di *Ismar Data*. La più grande divisione avviene tra **front-end** e **back-end**. Il primo rappresenta la vera e propria applicazione che l’utente andrà ad utilizzare, quindi consiste nello sviluppo della grafica e delle richieste al back-end per ottenere i dati da visualizzare. Il back-end invece rappresenta le fondamenta. Si occupa del reperimento dei dati, della loro elaborazione, del salvataggio all’interno di un database e della creazione delle **API**⁴ (*Application Programming Interface*) necessarie al reperimento di essi. Nei successivi paragrafi saranno trattati i singoli blocchi presenti nello schema, dando un’idea semplice del perché sono stati inseriti nel disegno.

⁴infra, pp. 61-63.

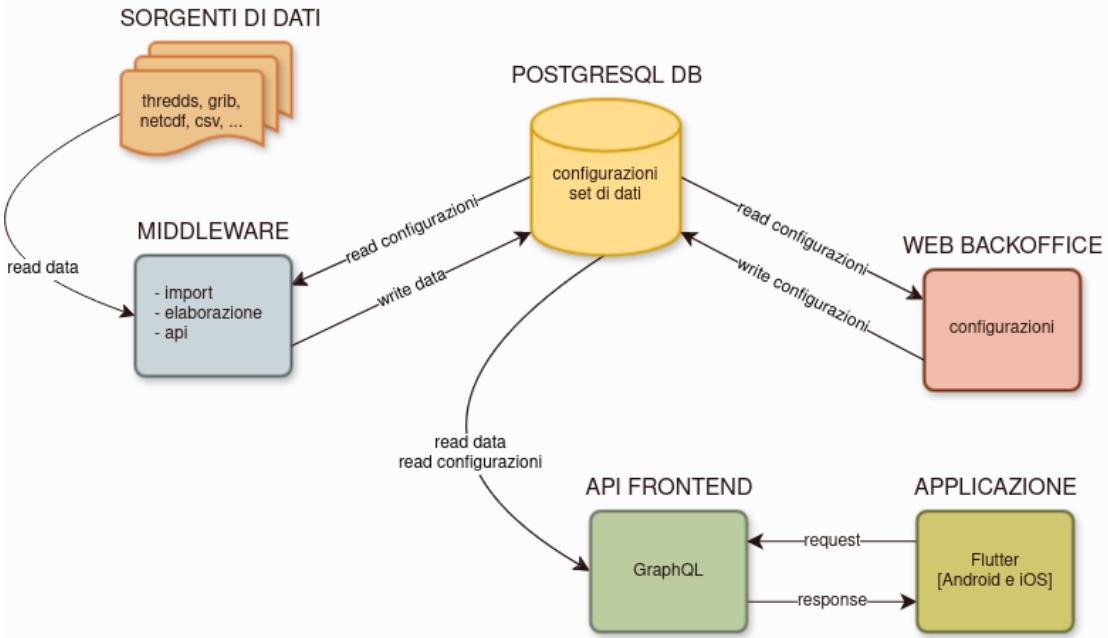


Figura 2.4: Schema architettura. (Diagramma creato utilizzando drawio^a).

^aJGraph Ltd. *Flowchart Maker & Online Diagram Software - Draw.io (Online)*, <https://www.drawio.com/>.

L'entità centrale è il database relazionale **postgresql**⁵. Al suo interno sono conservati tutti i dati necessari al funzionamento dell'applicazione. È stato scelto un database di questo tipo in quanto funziona su tutti i sistemi operativi maggiormente utilizzati, è *open-source*, è sicuro e interagisce bene con le altri componenti. Inoltre, essendo relazionale, consente di avere uno schema ben strutturato con entità (tabelle) e relazioni tra esse, utile per i dati *statici*. Con questo termine si intendono i dati riguardanti le infrastrutture vere e proprie (vedi esempio di piattaforma in Figura 2.5) che si occupano della raccolta dei dati tramite i sensori. Le informazioni riguardanti i sensori raramente evolvono nel tempo e hanno dati molto simili tra loro (ad esempio il tipo di sensore utilizzato viene spesso ripetuto, o l'ente proprietario), dunque il salvataggio in un database relazionale è la scelta più adatta. Per quanto riguarda invece i dati dinamici, la situazione si complica. Inizialmente era stato previsto un ulteriore database, non relazionale perché sono dati diversi tra loro, per il

⁵The PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database (Online)*, <https://www.postgresql.org/>.

salvataggio di essi. Si sarebbe dovuto interfacciare con il *middleware* e le *API front-end*. Il problema è che questi dati, come accennato nell'introduzione del documento, non sempre sono accessibili. Quindi la soluzione del secondo database è stata momentaneamente archiviata. I pochi dati disponibili, dopo un'attenta elaborazione, vengono salvati nel database principale PostgreSQL. Operazione possibile grazie a diversi meccanismi per l'aggiornamento delle nuove informazioni e l'eliminazione dei dati meno recenti, mantenendo uno storico, come da richiesta del cliente, di 10 giorni indietro dalla data odierna e di 5 giorni avanti per le eventuali previsioni.



Figura 2.5: Piattaforma Oceanografica Acqua Alta^a.

La Piattaforma Oceanografica Acqua Alta è situata a circa 8 miglia al largo del litorale di Venezia, in uno specchio di mare avente una profondità di circa 16 m (GPS 45.3142467 N, 12.5082483 E)^a.

^aISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/PTF-2018-07-scaled.pdf>.

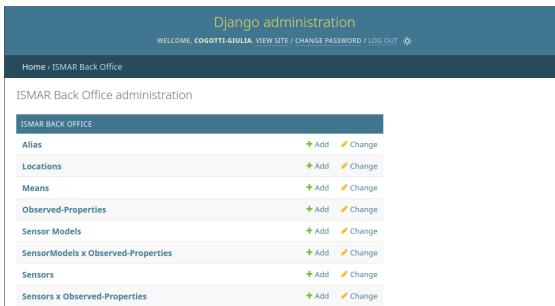
^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/PTF-2018-07-scaled.pdf>.

Il Web Backoffice è un semplice applicativo web, implementato tramite **Django**⁶, per la gestione e configurazione dei dati statici. Una schermata è mostrata in Figura 2.6. Django, tra le sue funzionalità, implementa anche un'interfaccia di amministrazione. Essa esiste come strumento per ispezionare e gestire i dati del database, non va considerata come un'interfaccia pubblica dei dati⁷.

Le sorgenti sono l'insieme di dati provenienti dalle varie fonti con tipi e modalità differenti. I dati rappresentano tutte le misurazioni effettuate dai sensori posti nelle

⁶Django Software Foundation and individual contributors. *The web framework for perfectionists with deadlines (Online)*, <https://www.djangoproject.com/>.

⁷A. Holovaty e J. Kaplan-Moss. *The Django Book*, pp. 102–103.



Esempio di interfaccia del sito di amministrazione, richiede credenziali di accesso e rispecchia i modelli del database.

Figura 2.6: Sito amministrazione backoffice^a.

^aFonte: screenshot dall'autrice.

stazioni e radar analizzati, dunque sono variabili nel tempo.

Il middleware è la componente che si occupa di prelevare i dati "grezzi", filtrarli ed eventualmente salvarli sul database. Questo passaggio è indispensabile per diverse ragioni. Innanzitutto le sorgenti raccolgono una mole di dati superiore a quella che effettivamente serve al funzionamento dell'applicazione, dunque prendere solo il necessario rende il progetto più leggero. Risulta opportuno ricordare che l'applicazione sarà utilizzata da un utente che si aspetta un tempo di risposta ragionevole, compreso tra i 0.1 e 10 secondi⁸. L'accesso ad alcune categorie di dati richiede un tempo non indifferente, decisamente superiore al limite, quindi il presalvataggio (effettuato durante l'orario di aggiornamento delle sorgenti, solitamente orario notturno) consente di evitare questo tipo di problematiche.

Il passaggio dei dati al lato grafico dell'applicazione avviene tramite API front-end che leggono i dati e le configurazioni presenti nel database. Come linguaggio di interrogazione al database si è scelto di utilizzare GraphQL⁹.

L'unica parte riguardante il front-end si focalizza sull'applicazione mobile. Essa sarà sviluppata utilizzando il framework Flutter¹⁰ in quanto risulta semplice e veloce da utilizzare. Una caratteristica fondamentale di Flutter è che consente di scrivere codice multi-piattaforma (nel caso specifico, Android e iOS) a partire da un codice base comune. Questo elimina i problemi derivanti dal dover scrivere un'applicazione

⁸R. B. Miller. «Response time in man-computer conversational transactions». AFIPS / ACM / Thomson Book Company, Washington D.C., pp. 268–276.

⁹The GraphQL Foundation. *GraphQL. A query language for your API (Online)*, <https://graphql.org/>.

¹⁰Google. *Flutter - Build for any screen (Online)*, <https://flutter.dev/>.

nativa per il sistema scelto. Il front-end sarà sviluppato da un team esterno, dunque non sarà analizzato nel presente documento.

2.3 Sorgenti di dati

I dati si suddividono in tre categorie principali: **stazioni osservative fisse, osservazioni da satellite e modelli di previsione**. È opportuno porre in luce immediatamente il *problema* alla base della raccolta di questi dati. Alcune sorgenti sono pubbliche (osservazioni da satellite e una parte delle stazioni osservative fisse), altre invece sono gestite da enti diversi e privati che tendono a non collaborare tra di loro. C'è stato il tentativo di uniformare il tutto ma, data la gestione privata dei server, non si è ancora giunti ad un accordo. Questo compromette il funzionamento dell'applicazione, sia per la difficoltà di reperire i dati che per assicurarsi che questi si aggiornino nel tempo. Al momento ISMAR-CNR, per evitare il congelamento della produzione dell'applicazione, ha preso la decisione di incaricare Elan42 della costruzione di un datacenter adatto alla memorizzazione dei dati privati provenienti dalle stazioni osservative fisse. Questa soluzione richiederà un quantitativo di tempo non indifferente. Si pensi che, dopo mesi, non ha ancora avuto inizio! Inizialmente ideato come una provvisoria soluzione adatta al contenimento di alcuni dati, si è giunti alla conclusione che fosse meglio creare qualcosa che funzionasse a lungo termine. Quindi, da un piccolo datacenter si è passati all'idea di creare un'infrastruttura che non solo contenesse i dati ma che si occupasse anche della loro elaborazione in più fasi: inizialmente vengono prelevati i dati dai sensori, in formato grezzo, a questo seguono due fasi di pulizia e interpolazione¹¹ di dati con applicazione di algoritmi appositi e infine, si arriva ad avere dei dati gestibili dal back-end dell'applicazione. Ecco che il tempo e i costi di produzione aumentano. Inoltre, continua a persistere il problema degli enti privati che non riescono, o non vogliono, dare accesso diretto ai sensori posti sul mare per il raccoglimento dei dati, né per il datacenter né per l'applicazione. Quindi, ISMAR-CNR ha proposto una soluzione provvisoria: ogni soggetto inserisce periodicamente i dati raccolti all'interno di una cartella condivisa, situata nei server ISMAR, e l'applicazione realizzata andrà a prelevarli da essa. Questo però non ha riscosso successo tra gli enti privati, dato che alcuni hanno acconsentito a collaborare, altri no. Inoltre non ci sarebbero certezze sulla disponibilità continua dei dati, dato che l'aggiornamento si basa su esseri umani che inseriscono periodicamente dei file nella cartella condivisa. Quindi così

¹¹ **interpolazione** [Der. del lat. *interpolatio -onis*, da *interpolare* comp. di *inter-* e v. *affine* a polire "pulire"] Procedimento per inserire tra due o più valori (in partic., dati sperimentali) altri valori in modo da ottenere una successione che abbia una certa regolarità, eventualmente rappresentabile con una funzione che abbia come suoi valori, sia pure approssimativamente, i valori di partenza; Treccani S.P.A. *Interpolazione. Dizionario delle Scienze Fisiche* (1996).

facendo non si risolverebbe completamente il problema. In conclusione, per quanto riguarda le sorgenti private, si è ancora ad un punto di stallo.

Qui di seguito saranno elencate le diverse fonti, pubbliche e private, evidenziando i dati raccolti, dove presenti, e le modalità di accesso ad essi.

2.3.1 Stazioni osservative fisse

Le stazioni osservative fisse richieste dall'applicazione riguardano le piattaforme poste sul mare. In particolare si dividono in stazioni meteorologiche fisse e radar. Le stazioni sono tutte private, quindi con i dati non completamente disponibili. Mentre i radar hanno dati pubblici accessibili, salvo malfunzionamenti del server fornitore.

Stazioni meteorologiche fisse. L'ambiente marino necessita di essere osservato e tutelato. In Italia esistono vari sistemi di osservazione indipendenti che si occupano della raccolta dei dati. In particolare la *Rete Italiana di siti fissi per l'osservazione del mare (IFON)* è composta da diverse piattaforme fisse in grado di trasmettere i dati acquisiti quasi in tempo reale. Esse garantiscono l'osservazione degli ambienti costieri, di mare aperto e profondi, per alcune variabili inserite nelle **EOV – Essential Ocean Variables**¹².

La rete **IFON** (*Italian Fixed-point Observatory Network*), composta da 17 infrastrutture costiere e di mare aperto, costituisce un sistema distribuito di monitoraggio a livello nazionale. In Figura 2.7 è possibile osservare la mappa contenente i siti per l'osservazione del mare appartenenti alla rete.

L'applicazione richiede la visualizzazione dei dati provenienti da alcuni impianti della rete, situati nel quadrante nord del mar Adriatico:

1. Piattaforma Oceanografica Acqua Alta.
2. Meda PALOMA.
3. Meda S1-GB.
4. Boa meteo oceanografica E1.

Queste strutture verranno brevemente descritte qui di seguito.

La Piattaforma Oceanografica **Acqua Alta** (vedi Figura 2.9), installata nel 1970 a circa 10 miglia nautiche dal Lido di Venezia, alle coordinate: Lat. 45° 18.830' N,

¹²M. Ravaioli et al. *La rete scientifica italiana di siti fissi per l'osservazione del mare – IFON Stato dell'arte e upgrades durante il Progetto RITMARE (2012 – 2016)*. A cura di M. Ravaioli, C. Bergami e F. Riminucci. Roma, CNR Pubblicazioni 2017, p. 5.

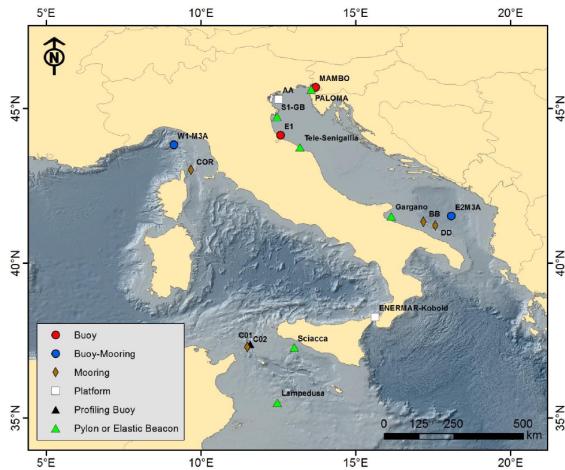


Figura 2.7: Mappa rete IFON^a.

^aFonte: M. Ravaioli et al. *La rete scientifica italiana di siti fissi per l'osservazione del mare – IFON Stato dell'arte e upgrades durante il Progetto RIT-MARE (2012 – 2016)*. A cura di M. Ravaioli, C. Bergami e F. Riminucci. Roma, CNR Pubblicazioni 2017, p. 7.

Lon. 12° 30.530' E, è una piattaforma dedicata esclusivamente alla ricerca oceanografica. Nella mappa in Figura 2.8 è possibile osservare la sua esatta posizione. La struttura è autonoma energeticamente grazie a pannelli solari, generatori eolici e generatori diesel configurabili da remoto in modo da poter raccogliere dati continuativi nel tempo. I dati registrati dal sistema, in media ogni 30 minuti, vengono inviati a terra tramite collegamento wireless a banda larga e raccolti nel centro dati di ISMAR-VE (sede ISMAR di Venezia). Le variabili climatiche rilevate riguardano temperatura dell'aria, umidità, pressione, direzione e velocità del vento, precipitazioni e radiazione solare. L'elenco aggiornato delle variabili climatiche rilevate è disponibile nel sito di ISMAR-CNR¹³. Questo è possibile grazie ai sensori posti a 1.6 m (livello superficiale), 6 m (livello intermedio) e 14 m (livello di fondo) di profondità¹⁴. Tra tutte le variabili climatiche, il cliente ha deciso che, per la prima versione dell'applicazione, si terrà conto solamente di 6 variabili (le uniche per le quali si è

¹³ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/infrastrutture-oceanografiche/piattaforma-acqua-alta/>.

¹⁴Ravaioli et al., cit., pp. 12–13.

Mappa dei siti della rete scientifica italiana di siti fissi per l'osservazione del mare. I tipi di struttura sono indicati nella legenda (Boe, piattaforme, mede, mooring, sistemi profilanti).



Figura 2.8: Sito Piattaforma Oceanografica Acqua Alta^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/07/mappa-PTF.pdf>.

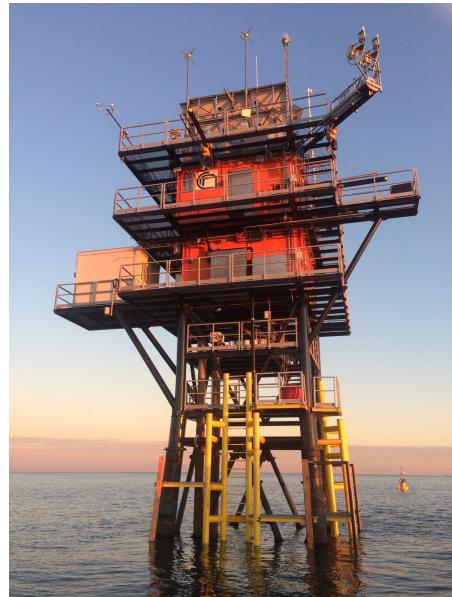


Figura 2.9: Foto Piattaforma Oceanografica Acqua Alta^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/PTF-2018-05-Pomaro-1-scaled.pdf>.

trovato il modo di accedere) osservate da diversi sensori:

1. Direzione vento (sensore Davis Vantage PRO);
2. Intensità vento (sensore Davis Vantage PRO);
3. Direzione vento (sensore SIAP Micros t033 TDV);
4. Intensità vento (sensore SIAP Micros t031 TDV);
5. Altezza d'onda (sensore SIAP Micros t021 TLU16);
6. Livello del mare (sensore SIAP Micros t039 TIDROM).

Le prime due utilizzano l'**API allmeteo**, che si occupa di trasferire dati meteorolo-

gici, non in tempo reale, dal back-end allmeteo ai server di terze parti¹⁵, accessibili solo con autenticazione. Le altre utilizzano le **API del comune di Venezia**, accessibili tramite url TLS (*Transport Layer Security protocol*) certificato con credenziali del tipo login/password. Un TLS è un protocollo progettato per consentire alle applicazioni client/server di comunicare su Internet senza intercettazioni, manomissioni o contraffazioni di messaggi¹⁶.



Figura 2.10: Ubicazione meda Paloma^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/mappa-Paloma.pdf>.



Figura 2.11: Foto meda Paloma in mare^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, https://www.ismar.cnr.it/wp-content/uploads/2023/06/PALOMA_cantoni_01.pdf.

¹⁵Postman. *allmeteoAPI (Online)*, <https://documenter.getpostman.com/view/11878734/TVYAf1Fn>.

¹⁶D. Howe. *Transport Layer Security protocol (Online)*, https://foldoc.org/Transport_Layer_Security_protocol.

La meda¹⁷ **PALOMA** – *Piattaforma Avanzata Laboratorio Oceanografico Mare Adriatico* – (vedi Figura 2.11) si trova, come si può notare in Figura 2.10, al centro del Golfo di Trieste, a circa 8 miglia nautiche dalla costa triestina, su un fondale di 25 metri. È operativa dal 2002, e da allora fornisce dati meteo marini raccolti ogni 5 minuti e trasmessi a terra ogni 3 ore. I parametri meteorologici, misurati tramite sensori posti a 3, 15 e 14 metri di profondità, includono la temperatura, l'umidità, la pressione, le precipitazioni, la direzione e intensità del vento e la radiazione solare. Inoltre, a 3m di profondità viene misurato l'ossigeno dissolto. L'elenco aggiornato delle variabili climatiche rilevate è disponibile nel sito di ISMAR-CNR¹⁸. I dati acquisiti vengono trasmessi regolarmente ad un server dedicato presso la sede di ISMAR-TS (sede ISMAR di Trieste)¹⁹.

La **meda S1-GB** (vedi Figura 2.13) è situata alle coordinate Lat. 44° 44.4' N, Lon. 12 27.3' E, a circa 4 miglia nautiche a Sud della foce di Po di Goro (Delta del fiume Po - Adriatico Settentrionale, vedi Figura 2.12), su di un fondale di 22.5 m. Nasce come *boa S1* nel 2004, per poi essere potenziata sostituendo la *boa galleggiante* con una stazione fissa a meda elastica. L'elasticità fa in modo che quando avviene una collisione la meda si inclini momentaneamente per poi ritornare in posizione, riducendo i possibili danni dovuti allo scontro tra meda e veicolo marino²⁰ (in Figura 2.14 è possibile osservare il confronto tra *boa fissa* e *meda elastica*). Da qui il nome meda S1-GB (*già Boa meteo-oceanografica S1*). Essa è costituita da vari sensori posti a diversi livelli che acquisiscono parametri oceanografici, meteorologici e biogeochimici²¹. In particolare contiene la strumentazione oceanografica a due livelli di profondità, -2,5 m e -18,5 m. Le variabili analizzate riguardano la temperatura della superficie del mare, l'ossigeno dissolto, la clorofilla e così via. Esse

¹⁷ **méda** s. f. [voce settentr.: lat. *mēta* (v. *mèta1*)]. – 1. Nel linguaggio marin., propriam., nome di segnali disposti in mare, per lo più fissi, di forme e di colori varî, in metallo o in muratura, impiegati come avviso ai navigatori in corrispondenza di punti pericolosi (scogli affioranti, secche, ecc.), come punti di riferimento (per l'atterraggio, per l'imboccatura di un canale di accesso al porto, ecc.), per indicare, in acque ristrette, il tratto di mare navigabile con sicurezza: m. luminosa, quella sormontata da un fanale, generalmente rosso o verde, e dipinta dello stesso colore; m. semielastica, specie di *boa*, generalmente luminosa, ancorata al fondo del mare per mezzo di una robusta asta di ormeggio, in grado di galleggiare e di oscillare (anche verticalmente) per mantenere visibile il segnale in presenza di onde alte sino a tre metri; m. sonora, fornita di trasmettitore acustico (nautofono, ecc.); Treccani S.P.A. *Méda*. In *Vocabolario Treccani online*.

¹⁸ ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/infrastrutture-oceanografiche/paloma/>.

¹⁹ Ravaioli et al., cit., pp. 11–12.

²⁰ Resinex Trading S.r.l. *ELASTIC BEACONS. The widest range of elastic beacons in the world*, pp. 1–4.

²¹ Ravaioli et al., cit., pp. 13–15.



Figura 2.12: Posizione meda S1-GB^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/mappa-S-GB.pdf>.



Figura 2.13: Foto meda S1-GB^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, https://www.ismar.cnr.it/wp-content/uploads/2023/06/S1-GB_Foto1_cr1.pdf.

sono ampiamente descritte nel sito di ISMAR-CNR²².

La **boa meteo-oceanografica E1** si trova nell’Adriatico Settentrionale, a circa 4 miglia nautiche a Nord della città di Rimini. Il sistema è ancorato al fondo mare mediante catena e corpo morto su di un fondale di 10.5 m. Si trova alle coordinate Lat. 44° 08.5' N, Lon. 12° 34.2' E dal 2006 (vedi Figura 2.15 e Figura 2.16). Essa è dotata di strumentazione per il rilevamento di parametri meteorologici a 2.5m sul livello del mare e parametri oceanografici su due livelli di sensori installati alle profondità di 1.6m (livello superficiale) e 8m (livello profondo). Le variabili oceanografiche a 8m riguardano salinità, temperatura del mare e ossigeno dissolto, mentre in superficie viene misurata la clorofilla e la torbidità. Esse sono descritte nel sito di ISMAR-CNR²³.

I dati di queste ultime tre stazioni (Paloma, S1-GB ed E1) non sono pubblici e, nonostante i tentativi di cercare una soluzione al problema, non è stato fornito

²²Fonte: ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/infrastrutture-oceanografiche/meda-s1-gb/>.

²³ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/infrastrutture-oceanografiche/boa-e1/>.

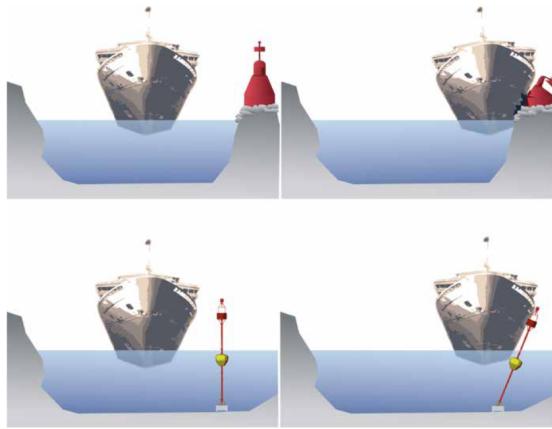


Figura 2.14: Confronto tra classica boa fisso (in alto) e innovativa meda elastica (in basso)^a.

^aFonte: Resinex Trading S.r.l. *ELASTIC BEACONS. The widest range of elastic beacons in the world*, p. 4.



Figura 2.15: Posizione boa E1^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/mappa-E1.pdf>.

Disegno che evidenzia il principale vantaggio di una meda elastica: evitare danni nonostante le collisioni. Questo vale sia per la meda che per il veicolo con cui avviene lo scontro. Un’ulteriore nota positiva riguarda la facilità di trasferimento nel caso in cui ci sia necessità di spostarla.



Figura 2.16: Foto boa E1^a.

^aFonte: ISMAR-CNR. *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/wp-content/uploads/2022/11/boe-costa-romagnola-ismar-cnr-2.pdf>.

l'accesso a nessuna variabile misurata.

Radar. Il *radar ad alta frequenza (HFR)* è uno strumento di telerilevamento terrestre che permette di ottenere misurazioni su correnti, onde e direzione dei venti²⁴. I radar sono in grado di fornire mappe di velocità superficiale in ampie regioni del mare (range fino a 200km) ad intervalli di tempo regolari, tipicamente di un'ora, consentendo il monitoraggio continuo delle correnti marine superficiali²⁵.

La rete radar **HFR-TirLig** di ISMAR-CNR fornisce accesso ai dati in tempo reale provenienti dalle stazioni attive lungo la costa del Mar Tirreno nordoccidentale e Mar Ligure (vedi Figura 2.17). La rete è composta da due stazioni che emettono a frequenze di 13,5 MHz e due che emettono a frequenze di 16,275 MHz, fornendo dati orari di corrente radiale (*singole misurazioni per ogni stazione*) con una risoluzione di portata rispettivamente di 1,5 km e 1 km e una copertura radiale prossima rispettivamente a 80 km e 40 km. Le misurazioni radiali delle singole stazioni vengono combinate su una griglia regolare, con risoluzione spaziale di 2km, su una copertura totale di circa 9000 km quadrati, al fine di ottenere le misurazioni totali. I dati radiali sono misurazioni effettuate per ogni singolo radar, mentre i dati totali rappresentano l'unione, tramite apposite tecniche, dei dati radiali (come se fossero dati di un unico radar). In questo modo i dati saranno meno precisi ma comunque validi²⁶.

I dati sono disponibili in un catalogo THREDDS²⁷ aggiornato contenente le variabili osservate in formato standard NetCDF-4²⁸.

I cataloghi THREDDS sono documenti XML, forniti dal THREDDS Data Server (TDS)²⁹, che elencano i set di dati e i servizi – protocolli – per l'accesso ad essi.

Come accennato nel paragrafo precedente, le misurazioni si distinguono in totali e radiali. Al momento, dato che l'applicazione si rivolge a un pubblico generale, si

²⁴A. Rubio et al. «HF Radar Activity in European Coastal Seas: Next Steps toward a Pan-European HF Radar Network». In: *Frontiers in Marine Science*, p. 2.

²⁵ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/infrastrutture-oceanografiche/rete-radar-costiera/>.

²⁶ID. *Near Real Time Surface Ocean Velocity by HFR-TirLig network*, <https://www.hfrnode.eu/networks/hfr-tirlig/>.

²⁷ISMAR-CNR. *HFR-TirLig_catalog (Dataset)*, https://thredds.hfrnode.eu:8443/thredds/NRTcurrent/HFR-TirLig/HFR-TirLig_catalog.html.

²⁸L. Corgnati et al. «Recommendation Report 2 on improved common procedures for HFR QC analysis. JERICO-NEXT WP5-Data Management», pp. 19–47.

²⁹Unidata/UCAR Community Programs (UCP). *NSF Unidata Software Documentation (Online)*, <https://doi.org/10.5065/D6N014KG>.



Figura 2.17: Rete radar ad alta frequenza (HF=High Frequency)^a.

^aFonte: ISMAR-CNR. *Rete radar HF (Immagine)*, <https://www.ismar.cnr.it/wp-content/uploads/2023/06/mappa-Radar-HF.pdf>.

è presa la decisione di implementare l'applicazione utilizzando i soli dati totali. In Figura 2.18 è possibile osservare la distinzione tra le cartelle all'interno del catalogo.

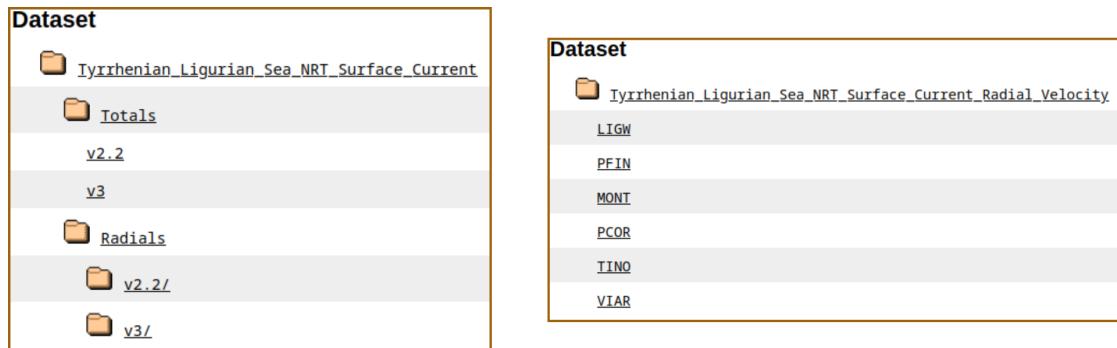


Figura 2.18: Dataset HFR. Distinzione tra dati totali e radiali all'interno del catalogo. I dati radiali avranno un'ulteriore divisione in sottocartelle, una per ogni radar presente nel sistema (figura a destra), mentre i dati totali visualizzeranno direttamente le informazioni sui dati.^a.

^aFonte: ISMAR-CNR. *HFR-TirLig_catalog (Dataset)*, https://thredds.hfrnode.eu:8443/thredds/NRTcurrent/HFR-TirLig/HFR-TirLig_catalog.html.

Tra le tante variabili osservate³⁰, le uniche rilevanti risultano essere la velocità dell'acqua del mare in superficie verso est – *surface eastward sea water velocity* – (**EWCT**) e la velocità del mare in superficie verso nord – *surface northward sea water velocity* – (**NSCT**), misurate entrambe in metri al secondo ($m * s^{-1}$). Vengono impiegate, come da standard meteorologico, nel calcolo del vento³¹.

2.3.2 Osservazioni da satellite

Le osservazioni da satellite richieste per l'applicazione sono relative all'osservazione della temperatura superficiale e della clorofilla presenti nel mar Mediterraneo in un'area geografica circoscritta. I dati sono disponibili in un catalogo THREEDS aggiornato.

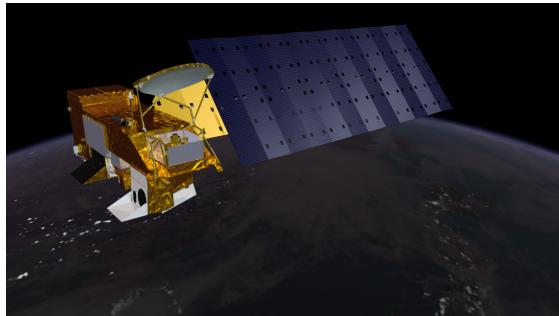


Figura 2.19: Satellite Acqua NASA^a.

^aFonte: R. Stöckli. *Satellite Aqua della NASA (Immagine)*, https://upload.wikimedia.org/wikipedia/commons/f/fb/The_Aqua_Satellite.pdf.

Il satellite Aqua orbita a 705km^a sopra la terra e trasporta una serie di sensori progettati per il monitoraggio dell'acqua e dell'atmosfera sul pianeta Terra.

^aL. Oreopoulos, L. Boisvert e P. Przyborski. *About Aqua / Aqua Project Science (Online)*, <https://aqua.nasa.gov/content/about-aqua>.

Clorofilla-a (Chla). La clorofilla rappresenta il pigmento fotosintetico predominante nel fitoplancton³². La *concentrazione di clorofilla-a (Chla)* è una variabile climatica che indica l'abbondanza del fitoplancton e il colore dell'oceano. Queste osservazioni permettono di comprendere la variabilità del fitoplancton, stato di eutrofizzazione

³⁰Cognati et al., cit., pp. 40–50.

³¹Atlassian. *Ecmwf confluence (Online)*, <https://confluence.ecmwf.int/pages/viewpage.action?pageId=111155337>.

³²**fitoplàncton** s. m. [comp. di fito- e plancton]. – In ecologia, l'insieme degli organismi vegetali che costituiscono il plancton; si distinguono un f. d'acqua dolce e un f. marino, costituiti da cloroficee, diatomee, dinoflagellati, ecc; Treccani S.P.A. *Fitoplàncton*. In *Vocabolario Treccani online*.

ne³³ e qualità dell'acqua. Dunque Chla risulta essere una delle variabili climatiche essenziali per caratterizzare il clima del pianeta³⁴.

La Chla viene rilevata da satellite utilizzando appositi algoritmi empirici (non analizzati nel presente documento) indicati come *algoritmi del colore dell'oceano*³⁵ (**OC**). I dati vengono prelevati ed elaborati tramite l'unione delle informazioni provenienti da diversi sensori, col cosiddetto *approccio multi-sensore*, per aumentare la probabilità di osservazioni valide di cielo sereno³⁶. Le osservazioni si concentrano nell'area geo-spaziale rappresentante il mare italiano, delimitata da:

- *Longitudine*: da 7.0 a 20.0 Risoluzione=0.012 gradi est
- *Latitudine*: da 35.0 a 46.0 Risoluzione=0.012 gradi nord

I dati sono disponibili in un catalogo THREDDS³⁷ aggiornato contenente le variabili osservate in formato standard NetCDF-4³⁸. La concentrazione di clorofilla è misurata in *milligram m⁻³*. È presente un visualizzatore web (*GODIVA2*) che consente la visualizzazione grafica dei dati provenienti dal dataset³⁹. Un esempio è mostrato in Figura 2.20.

L'unica variabile osservata risulta essere la Chla, dunque non ci sarà bisogno di alcuna selezione nella fase di importazione.

Sea surface temperature (SST). Gli oceani svolgono un ruolo importante nel sistema climatico, in parte grazie alla loro capacità di accumulare calore. L'inerzia

³³ **eutrofizzazione** s. f. [der. di eutrofico]. – In ecologia, il processo per cui una massa d'acqua (per es. un lago) diventa più eutrofica, sia per mutazione naturale sia per fertilizzazione artificiale, per es. da inquinamento; quest'ultima, detta e. culturale, è dovuta, fondamentalmente, ad accumulo di grosse quantità di sali fosforici provenienti da detergivi o da fertilizzanti, e di composti azotati presenti nei fertilizzanti stessi e nelle acque luride, ed è causa delle cosiddette fioriture di fitoplancton; ID. *Eutrofizzazione*. In *Vocabolario Treccani online*.

³⁴ J. Merder et al. «A novel algorithm for ocean chlorophyll-a concentration using MODIS Aqua data». In: *ISPRS Journal of Photogrammetry and Remote Sensing*, pp. 198–211.

³⁵ H. M. Dierssen. «Perspectives on empirical approaches for ocean color remote sensing of chlorophyll in a changing climate». In: *Proceedings of the National Academy of Sciences*, pp. 17073–17078.

³⁶ G. Volpe et al. «Mediterranean ocean colour Level 3 operational multi-sensor processing». In: *Ocean Science*, pp. 129–132.

³⁷ ISMAR-CNR. *dataset-oc-med-chl-multi-l3-chl_1km_daily-rt-v02 (Dataset)*, http://ritmare.artov.ismar.cnr.it/thredds/ritmare/SatelliteOS/OC/catalog.html?dataset=CHL_CASE12_A.

³⁸ Cognati et al., cit., pp. 19–47.

³⁹ J. D. Blower et al. «GODIVA2: interactive visualization of environmental data on the Web». In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, pp. 1035–1039.



Figura 2.20: Chla rilevata in data 22 maggio 2023, visualizzazione tramite GODIVA^a.

^aFonte: ISMAR-CNR. *dataset-oc-med-chl-multi-l3-chl_1km_daily-rt-v02 (Immagine)*, <http://ritmare.artov.ismar.it/thredds/godiva2/godiva2.html?menu=&layer=CHL&elevation=0&time=2023-05-22T00:00:00Z&scale=0.01,10&bbox=-238.354357,-338.673427,481.645643,223.826573&server=http://ritmare.artov.ismar.cnr.it/thredds/wms/xchlcase12>.

termica degli oceani viene comunicata all'atmosfera tramite energia scambiata sulla superficie del mare. Questi flussi energetici dipendono principalmente dalla *temperatura della superficie del mare (SST)* unitamente alla temperatura dell'aria, umidità e nuvolosità. In ogni caso la SST, come variabile climatica, svolge un ruolo chiave nell'analisi della regolazione del clima e dei suoi fenomeni di variabilità⁴⁰. È utilizzata da alcuni sistemi di previsione ma anche, ed è questo il caso, da enti pubblici e

⁴⁰C. Deser et al. «Sea Surface Temperature Variability: Patterns and Mechanisms». In: *Annual review of marine science*, p. 116.

privati che si occupano dell'analisi dell'ambiente marino⁴¹.

La SST è rilevata tramite l'elaborazione di immagini notturne acquisite sia sui satelliti geostazionari che in orbita polare⁴². Le osservazioni si concentrano nell'area geo-spaziale rappresentante il mare italiano, delimitata da:

- *Longitudine*: da 7.0 a 20.0 Risoluzione=0.012 gradi est
- *Latitudine*: da 35.0 a 46.0 Risoluzione=0.012 gradi nord

I dati sono misurati in *kelvin(K)*, unità di temperatura termodinamica⁴³.

I dati sono disponibili in un catalogo THREDDS⁴⁴ aggiornato contenente le variabili osservate in formato standard NetCDF-4⁴⁵. È presente un visualizzatore web – *GODIVA2* – che consente la visualizzazione grafica dei dati provenienti dal dataset⁴⁶. Un esempio è mostrato in Figura 2.21.

L'unica variabile osservata risulta essere la SST, dunque non ci sarà bisogno di alcuna selezione nella fase di importazione.

2.3.3 Modelli di previsione

I modelli di previsione permettono di predire il tempo climatico futuro. L'applicazione visualizzerà i diagrammi di due sistemi: **Nettuno** per il mar Mediterraneo, **Henetus** per il nord del mar Adriatico. Quando si pensa alle previsioni la prima cosa che viene in mente è se ci sarà il sole o la pioggia. Invece, le due strutture si occupano prevalentemente di onde, Nettuno anche del vento. Rivelazione un po' deludente, ma anche loro sono importanti! Ad esempio Henetus viene utilizzato per predire l'acqua alta a Venezia, fortemente influenzata dal moto ondoso del litorale veneto⁴⁷. Nei paragrafi successivi saranno analizzati entrambi in modo approfondito.

⁴¹ B. Buongiorno Nardelli et al. «Evaluation of different covariance models for the operational interpolation of high resolution satellite Sea Surface Temperature data over the Mediterranean Sea». In: *Remote Sensing of Environment*, p. 334.

⁴² Ivi, p. 335.

⁴³ T. Renner et al. *Quantities, units and symbols in physical chemistry*. The Royal Society of Chemistry, p. 87.

⁴⁴ Copernicus Marine Service e Institute of Marine Sciences. *Mediterranean SST Analysis, L4, 1km daily (Dataset)*, <http://ritmare.artov.ismar.cnr.it/thredds/ritmare/SatelliteOS/SST/catalog.html?dataset=SSTUHR>.

⁴⁵ Cognati et al., cit.

⁴⁶ Blower et al., cit.

⁴⁷ L. Bertotti et al. «The Henetus wave forecast system in the Adriatic Sea». In: *Natural Hazards and Earth System Science*, pp. 2972–2976.

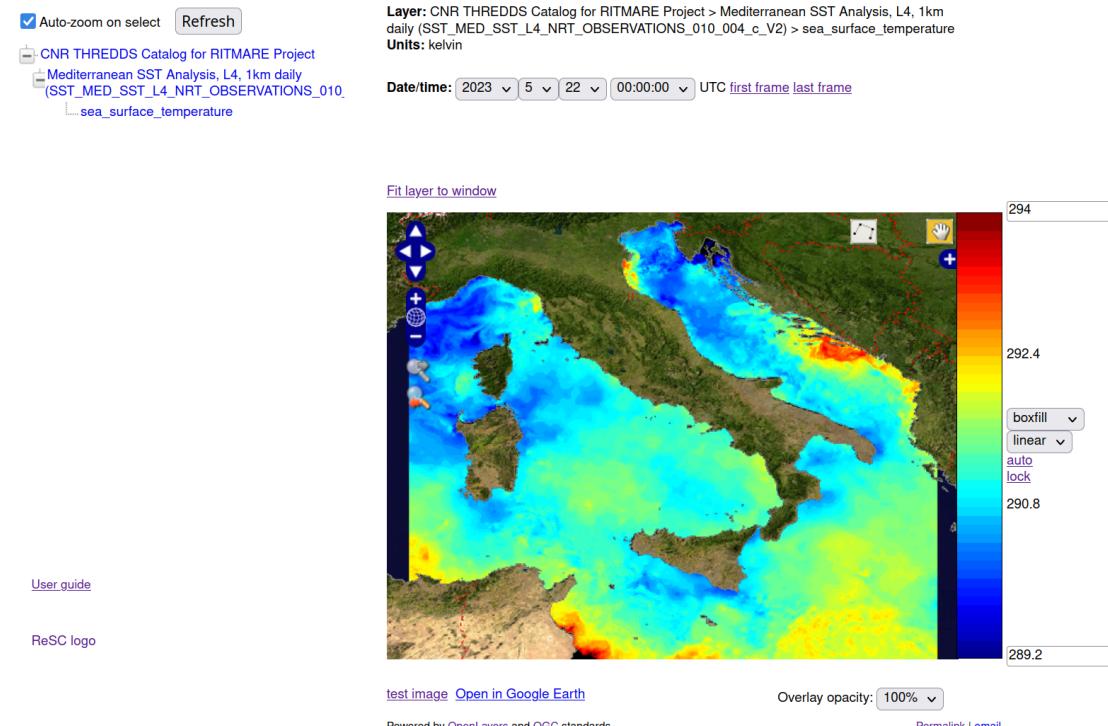


Figura 2.21: SST rilevata in data 22 maggio 2023, visualizzazione tramite GODIVA^a

^aFonte: ISMAR-CNR. *Mediterranean SST Analysis, L4, 1km daily (Immagine)*, http://ritmare.artov.ismar.cnr.it/thredds/godiva2/godiva2.html?menu=&layer=analysed_sst&elevation=0&time=2023-05-22T00:00:00Z&scale=289.2,294&bbox=66.889391,84.287338,78.139391,93.0764&server=http://ritmare.artov.ismar.cnr.it/thredds/wms/sstuhr.

Nettuno. Nettuno è un sistema di previsione dei venti e delle onde per il mar Mediterraneo. Esso è operativo dal 2009 e, ogni 24 ore, produce le previsioni fino a 72 ore delle condizioni meteo. Il modello meteorologico **COSMO** (*Consortium for small-Scale MOdelling*) si occupa di rilevare le condizioni del vento, mentre il modello spettrale di terza generazione **WAM** (*Mediterranean Wave Forecast*) valuta i campi d'onda⁴⁸.

Il mar Mediterraneo si estende per più di 3500 km in longitudine (6°W–36.5°E) e più di 1600 km in latitudine (30°–46°N). Nettuno garantisce la copertura dell'intera

⁴⁸ISMAR-CNR, *Istituto di scienze marine (Online)*, <https://www.ismar.cnr.it/infrastrutture/modellistica/modelli-operativi/#2>.

area geografica del Mediterraneo. Nettuno viene integrato due volte al giorno (a partire dalle 00:00 e dalle 12:00 UTC) per una previsione di 72 ore⁴⁹.

I dati non sono pubblici. ISMAR-CNR ha fornito, per conto dell'ente privato che gestisce i dati, alcuni file di esempio riguardanti le misurazioni effettuate in un arco temporale di 10 giorni nel mese di aprile 2024. Ovviamente il problema della mancanza di dati persiste. Non è possibile far funzionare un'applicazione che necessita di dati in continuo aggiornamento (in questo caso anche di previsione!) avendo a disposizione solo alcuni dati vecchi; l'unica nota positiva è la possibilità di analizzare i file per essere preparati quando si avrà a disposizione una sorgente reale da cui reperirli. I dati seguono il formato *GRIB*. **GRIB** (*General Regularly distributed Information in Binary form*) appartiene ai formati standard per l'archiviazione e lo scambio di dati meteorologici su griglia. Un GRIB file è definito dalla concatenazione di più file con estensione *.grib* o *.grb*⁵⁰. Il nome di ogni file segue il formato compresso *grib_WAM_yyyyymmddhh.tar.bz2*. Esso sta ad indicare che il contenuto del file si riferisce alle previsioni del modello di onde WAM, e contiene dati in formato grib. La data (*yyyymmdd*) fa riferimento al giorno in cui è stata fatta la previsione. Di seguito un numero (*hh*) indica l'orario, il quale può essere mezzanotte (ore 00) oppure mezzogiorno (ore 12). In Figura 2.22 è presente la lista dei file ricevuti. L'elemento selezionato ha nome *grib_WAM_2024041412.tar.bz2* e sta ad indicare il file con la previsione del giorno 14 aprile 2024 alle ore 12. All'interno del file sono contenuti esattamente 25 file grib del tipo *WAMyyyymmdd0000_hhh.grb*, ognuno dei quali si riferisce all'ora della previsione (*hhh*) con passo temporale di 3 ore, da 000 a 072. Questi sono i dati reali dei campi d'onda e di vento nel Mediterraneo.

In Figura 2.23 si possono osservare i 25 file relativi al 14 aprile 2024 alle ore 12. Ogni file grib contiene un certo numero di parametri, identificati da un numero, ad indicare le misurazioni effettuate:

- **140229**: altezza significativa, in metri, delle onde combinate del vento e del moto ondoso (*swh – Significant height of combined wind waves and swell*);
- **140230**: direzione media dell'onda (*mdw – Mean wave direction*);
- **140232**: periodo medio, in secondi, dell'onda (*mwp – Mean wave period*). Indica il tempo medio impiegato da due creste d'onda consecutive, sulla superficie dell'oceano/mare, per attraversare un punto fisso;

⁴⁹L. Bertotti et al. «Nettuno: Analysis of a Wind and Wave Forecast System for the Mediterranean Sea». In: *Monthly Weather Review*, pp. 3130–3132.

⁵⁰Atlassian, cit., <https://confluence.ecmwf.int/display/CKB/What+are+GRIB+files+and+how+can+I+read+them>.

Nome	Dimensione	Tipo	Data di modifica
grib_WAM_2024041412.tar.bz2	42,4 MiB	Tar archive (bzip2-compressed)	14/04/2024
grib_WAM_2024041512.tar.bz2	42,3 MiB	Tar archive (bzip2-compressed)	15/04/2024
grib_WAM_2024041600.tar.bz2	42,4 MiB	Tar archive (bzip2-compressed)	16/04/2024
grib_WAM_2024041612.tar.bz2	42,3 MiB	Tar archive (bzip2-compressed)	16/04/2024
grib_WAM_2024041700.tar.bz2	42,7 MiB	Tar archive (bzip2-compressed)	17/04/2024
grib_WAM_2024041800.tar.bz2	43,0 MiB	Tar archive (bzip2-compressed)	18/04/2024
grib_WAM_2024041812.tar.bz2	43,0 MiB	Tar archive (bzip2-compressed)	18/04/2024
grib_WAM_2024041900.tar.bz2	43,2 MiB	Tar archive (bzip2-compressed)	19/04/2024
grib_WAM_2024041912.tar.bz2	43,1 MiB	Tar archive (bzip2-compressed)	19/04/2024
grib_WAM_2024042012.tar.bz2	42,8 MiB	Tar archive (bzip2-compressed)	20/04/2024
grib_WAM_2024042100.tar.bz2	42,8 MiB	Tar archive (bzip2-compressed)	21/04/2024
grib_WAM_2024042112.tar.bz2	42,8 MiB	Tar archive (bzip2-compressed)	21/04/2024
grib_WAM_2024042200.tar.bz2	42,7 MiB	Tar archive (bzip2-compressed)	22/04/2024
grib_WAM_2024042212.tar.bz2	42,9 MiB	Tar archive (bzip2-compressed)	22/04/2024
grib_WAM_2024042300.tar.bz2	43,1 MiB	Tar archive (bzip2-compressed)	23/04/2024
grib_WAM_2024042312.tar.bz2	43,4 MiB	Tar archive (bzip2-compressed)	23/04/2024
grib_WAM_2024042400.tar.bz2	43,5 MiB	Tar archive (bzip2-compressed)	24/04/2024
grib_WAM_2024041500.tar.bz2	42,3 MiB	Tar archive (bzip2-compressed)	25/04/2024
grib_WAM_2024042000.tar.bz2	43,0 MiB	Tar archive (bzip2-compressed)	25/04/2024

Figura 2.22: File compressi contenenti le previsioni di Nettuno con dati in formato grib dal 14 aprile al 24 aprile 2024^a.

^aFonente: screenshot dell'autrice.

- **140249:** direzione del vento (*dwi – 10 metre wind direction*). Rappresenta la direzione da cui soffia il vento, in gradi in senso orario dal nord geografico, ad un'altezza di dieci metri sopra la superficie della Terra;
- **140231:** periodo, in secondi, di picco dell'onda (*pp1d –Peak wave period*). Questo parametro rappresenta il periodo delle onde oceaniche più energetiche generate dai venti locali;
- **140233:** coefficiente di drag con le onde (*cdww – Coefficient of drag with waves*). Indica la resistenza che le onde dell'oceano esercitano sull'atmosfera. A volte viene anche chiamato "coefficiente di attrito". Non ha un'unità di misura;
- **140245:** velocità del vento (*wind – 10 metre wind speed*). Questo parametro è la velocità orizzontale del vento in metri al secondo ($m * s^{-1}$), ad un'altezza di dieci metri sopra la superficie della Terra;
- **131:** componente U del vento (*u – U component of wind*). È la velocità orizzontale dell'aria che si muove verso est, espressa in metri al secondo

$(m * s^{-1})$. Un segno negativo indica quindi un movimento d'aria verso ovest;

- **132:** componente V del vento ($v - V$ component of wind). È la velocità orizzontale dell'aria che si muove verso nord, espressa in metri al secondo $(m * s^{-1})$. Un segno negativo indica quindi un movimento d'aria verso sud.

I parametri sono divisi in tabelle in base al settore nel quale sono raccolti i dati. Le prime tre cifre del numero del parametro indicano la tabella di appartenenza, ad eccezione della tabella standard che non ha un numero identificativo. In questo caso i prime sette parametri appartengono alla tabella numero 140 relativa alle onde, gli ultimi due (componenti U e V) alla tabella standard. L'elenco completo dei possibili parametri e tabelle si può trovare nel database gestito dal Centro europeo per le previsioni meteorologiche a medio termine⁵¹ (ECMWF) – European Centre for Medium-range Weather Forecasts.

In realtà, per *Ismar Data* non tutti i parametri sono rilevanti. L'applicazione dovrà visualizzare un solo grafico dinamico (posto su una cartina geografica) relativo all'altezza delle onde e all'intensità del vento, in particolare l'altezza delle onde sarà rappresentata da un livello colorato e l'intensità del vento dalle frecce ad indicarne la direzione (verso della freccia) e intensità (lunghezza della freccia). Questi dati si ottengono usando i parametri **140229** e **140230** per le onde, mentre **131** e **132** per il vento. In Figura 2.24 è presente un esempio dei risultati che si ottengono utilizzando i dati a disposizione.

Henetus - Mar Adriatico. **Henetus**, dal 1996, è il sistema di previsione del moto ondoso nel Mar Adriatico. Si basa sui campi di vento in formato **ECMWF** (European Centre for Medium-range Weather Forecasts) ed è in grado di ottenere risultati fino a cinque giorni di previsione⁵².

Il mar Adriatico si estende per circa 740km di lunghezza e 200km di larghezza (vedi area in Figura 2.25). Le previsioni utilizzate dal CNR sono quelle riguardanti il quadrante nord, osservato dalla torre oceanografica di ISMAR⁵³.

I dati non sono pubblici. Anche in questo caso il cliente ha fornito, per conto dell'ente privato che gestisce i dati, alcuni file di esempio riguardanti le misurazioni effettuate in un arco temporale di quattro mesi, dal 14 gennaio al 23 aprile 2024. Questo, ovviamente, comporta le stesse problematiche presenti per i file di Nettuno. Ogni file contiene i campi d'onda relativi a 24 ore sull'area del Mar Adriatico.

⁵¹European Centre for Medium-Range Weather Forecasts. *Parameter Database (Online)*, <https://codes.ecmwf.int/grib/param-db/?encoding=grib1>.

⁵²ISMAR-CNR, cit., <https://www.ismar.cnr.it/infrastrutture/modellistica/modelli-operativi/#3>.

⁵³Bertotti et al., «The Henetus wave forecast system in the Adriatic Sea», p. 2967.

Nome	Dimensione	Tipo	Data di modifica
WAM202404141200_000.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_003.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_006.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_009.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_012.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_015.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_018.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_021.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_024.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_027.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_030.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_033.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_036.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_039.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_042.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_045.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_048.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_051.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_054.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_057.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_060.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_063.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_066.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_069.grb	2,5 MiB	Sconosciuto	14/04/2024
WAM202404141200_072.grb	2,5 MiB	Sconosciuto	14/04/2024

Figura 2.23: Contenuto del file grib_WAM_2024041412.tar.bz2: 25 file grib di dati misurati con passo temporale di 3 ore, da 000 a 072.^a.

^aFonte: screenshot dell'autrice.

Il nome di ogni file segue il formato *OPEyyymdd12xxxx.gz* dove OPE significa run/previsione OPERATIVA e *gz* indica la compressione del file. La data (*yyymdd*) si riferisce all'anno, mese e giorno dell'*ultimo* campo della previsione. Il numero, sempre 12, indica l'ora di *inizio* della previsione. Ne risulta che il giorno di inizio è in realtà il precedente alla data presente nel nome del file, in quanto quest'ultima indica appunto la data dell'*ultimo* campo della previsione (al termine delle 24 ore di monitoraggio). Infine i quattro numeri (*xxxx*) possono assumere sei diversi valori di diverso significato:

- 0000: *analisi* in ore;
- 0024: *previsione* in ore del giorno 1;

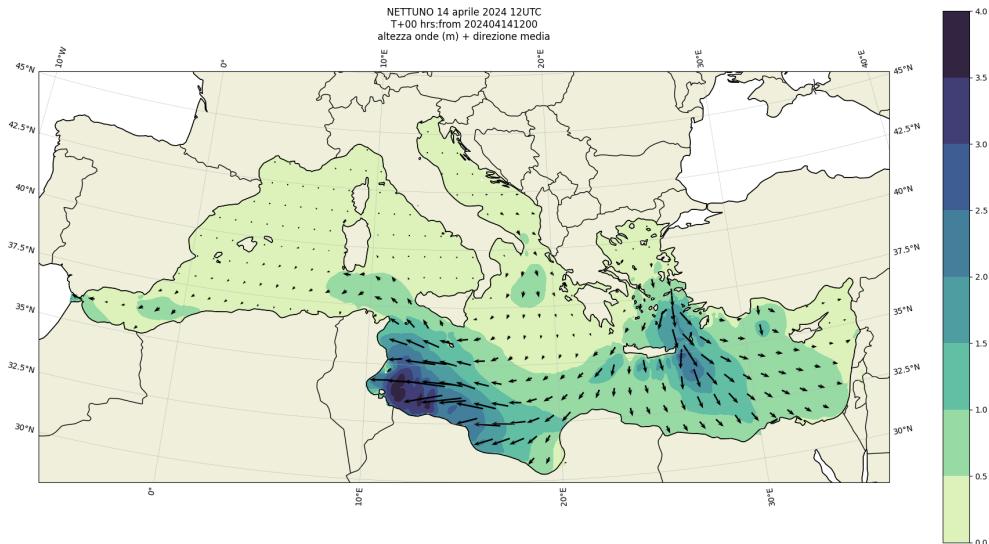


Figura 2.24: Esempio di grafico relativo all'altezza delle onde e all'intensità del vento misurate da Nettuno il 14 aprile 2024.^a

^aFonte: screenshot dell'autrice.

- 0048: *previsione* in ore del giorno 2;
- 0072: *previsione* in ore del giorno 3;
- 0096: *previsione* in ore del giorno 4;
- 0120: *previsione* in ore del giorno 5.

In Figura 2.26 è presente l'elenco dei file ricevuti. Il file selezionato, *OPE240114120072.gz*, rappresenta la previsione del terzo giorno (0072) a partire dal giorno 13/01/2024 alle ore 12; quindi la previsione del giorno 16/01/2024.

All'interno di questi file compressi è presente un unico file contenente le reali misurazioni. Il nome corrisponde al relativo file *OPEyyymmdd12xxxx.gz*, a meno dell'estensione. Il contenuto di questi file, a prima vista, risulta essere particolarmente complicato. Il gestore di questi dati ha quindi fornito una breve guida per aiutare gli sviluppatori nella comprensione. Un esempio, relativo al file *OPE240114120072* contenuto nel file selezionato in Figura 2.26, è mostrato in Figura 2.27. I campi d'onda sono raggruppati a intervalli di 3 ore e, per ogni ora, sono presenti quattro matrici di dimensione 97x73. Vengono considerati quattro parametri d'onda:

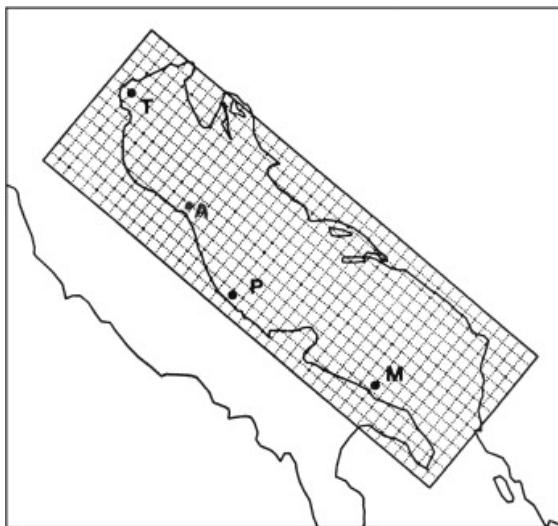


Figura 2.25: Geometria del mar Adriatico.

Immagine rappresentante il quadrante del mar Adriatico. I puntini rappresentano la posizione della torre oceanografica di ISMAR (T) e delle boe poste ad Ancona (A), Pescara (P) e Monopoli (M).

- **ONDAHS:** altezza onda;
- **ONDADIR:** direzione onda;
- **ONDAFM:** frequenza media dell'onda;
- **ONDAFP:** frequenza di picco dell'onda.

Ogni matrice contiene un parametro d'onda. I limiti geografici sono rappresentati da:

- latitudine: 40.0 - 46.00 gradi Nord;
- longitudine: 12.00 - 20.00 gradi Est.

con passo geografico 0.083×0.083 (1/12 x 1/12 di gradi) e il punto (1,1), posto in basso a sinistra, indica il punto sud-ovest.

La prima riga di ogni file contiene data e ora, dimensioni della matrice e limiti geografici. Da notare che la data non è la stessa del nome del file. Infatti si riferisce esattamente al giorno della previsione. Nell'esempio in questione si ha 2401161500 97 73 12.0 40.0 20.0 46.0. In questo caso la data è 16/01/2024 alle ore 15:00. Che indica esattamente la data e il giorno della previsione la previsione del terzo giorno (0072) a partire dal giorno 13/01/2024 alle ore 12 (nome del file *OPE240114120072*). Le righe successive contengono le matrici, con i parametri d'onda, e sono lette riga per riga dall'alto verso il basso (ovvero da nord a sud e da

Nome	Dimensione	Tipo	Data di modifica
OPE240114120000.gz	84,8 KiB	Archivio gzip	28/01/1978
OPE240114120024.gz	88,9 KiB	Archivio gzip	28/01/1978
OPE240114120048.gz	88,7 KiB	Archivio gzip	28/01/1978
OPE240114120072.gz	88,7 KiB	Archivio gzip	28/01/1978
OPE240114120096.gz	88,2 KiB	Archivio gzip	28/01/1978
OPE240114120120.gz	91,8 KiB	Archivio gzip	28/01/1978
OPE240115120000.gz	89,9 KiB	Archivio gzip	28/01/1978
OPE240115120024.gz	88,8 KiB	Archivio gzip	28/01/1978
OPE240115120048.gz	89,2 KiB	Archivio gzip	28/01/1978
OPE240115120072.gz	89,1 KiB	Archivio gzip	28/01/1978
OPE240115120096.gz	91,3 KiB	Archivio gzip	28/01/1978
OPE240115120120.gz	92,0 KiB	Archivio gzip	28/01/1978
...			
...			
...			
OPE240422120096.gz	85,5 KiB	Archivio gzip	28/01/1978
OPE240422120120.gz	82,3 KiB	Archivio gzip	28/01/1978
OPE240423120000.gz	89,3 KiB	Archivio gzip	28/01/1978
OPE240423120024.gz	87,9 KiB	Archivio gzip	28/01/1978
OPE240423120048.gz	88,1 KiB	Archivio gzip	28/01/1978
OPE240423120072.gz	86,2 KiB	Archivio gzip	28/01/1978
OPE240423120096.gz	82,2 KiB	Archivio gzip	28/01/1978
OPE240423120120.gz	81,1 KiB	Archivio gzip	28/01/1978

Figura 2.26: File compressi contenenti i campi d'onda rilevati dal sistema Henetus dal 14 gennaio al 23 aprile 2024.^a.

^aFonte: screenshot dell'autrice.

ovest ad est). Il grafico richiesto dal cliente è relativo all'altezza delle onde, quindi le uniche matrici rilevanti sono ONDAHS e ONDADIR.

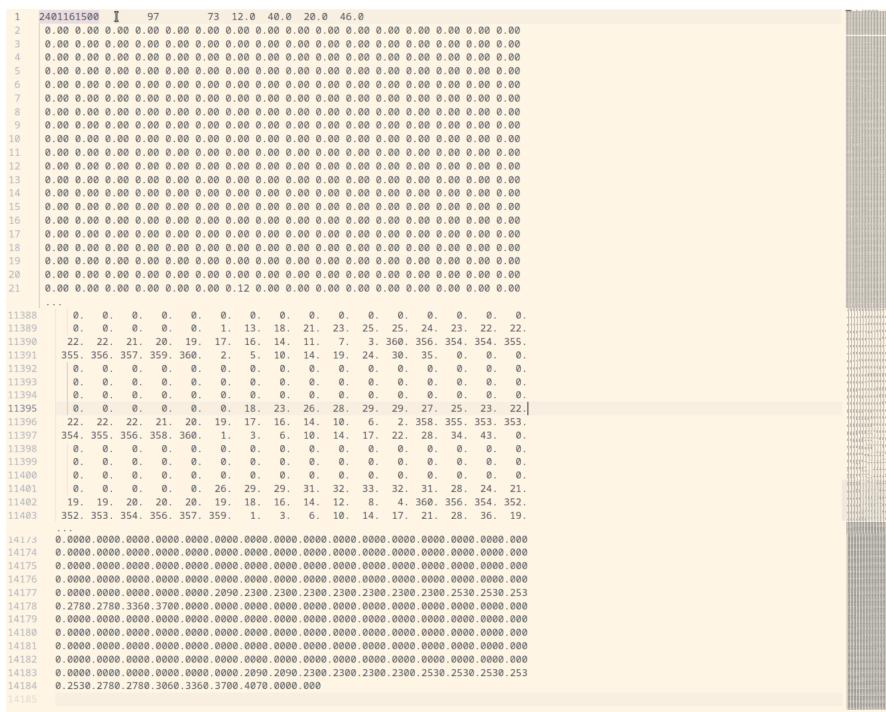


Figura 2.27: Esempio delle misurazioni dei campi d'onda di Henetus rappresentanti la previsione del 16/01/2024; terzo giorno (0072) a partire dal giorno 13/01/2024 alle ore 12.^a

^aFonte: screenshot dell'autrice.

Capitolo 3

Sviluppo

Questo capitolo si occupa di analizzare lo sviluppo di *Ismar Data*. Inizialmente vengono presentati gli strumenti di produzione e comunicazione, con approfondimento sulle tecnologie software utilizzate nella realizzazione del back-end dell'applicazione. In seguito sarà descritta la fase di importazione dei dati dalle sorgenti disponibili. Si noterà come, nonostante il diverso tipo di formato dei dati, ci sarà il tentativo di creare un codice il più generale possibile; a volte sarà possibile, altre volte no. Infine si analizzeranno le API, sia quelle di autenticazione, importanti per la sicurezza, che quelle che si interfacciano con il front-end. Tutto il lavoro fatto viene supportato da alcune immagini provenienti da una demo web, usata come provvisoria soluzione in attesa dello sviluppo delle reali interfacce grafiche, che mostreranno i risultati ottenuti dall'elaborazione dei dati e dall'utilizzo delle API.

3.1 Strumentazione

Il percorso di stage ha portato all'acquisizione di nuove conoscenze, non solo teoriche come visto in precedenza per le sorgenti di dati, ma anche pratiche. Questa sezione si occuperà di portare alla luce le conoscenze analizzando gli strumenti utilizzati, sia per lo sviluppo di *Ismar Data* che per la comunicazione tra il team.

Comunicazione. Quando si entra in una realtà lavorativa è necessario comprendere il modo in cui si affronta il lavoro. In questo caso esso si svolge in modalità mista, che significa che una parte di esso viene fatto online, grazie al cosiddetto **smart working**.

Il lavoro agile o smart working non è una diversa tipologia di rapporto di lavoro, bensì una particolare modalità di esecuzione della prestazione

di lavoro subordinato introdotta al fine di incrementare la competitività e di agevolare la conciliazione dei tempi di vita e lavoro¹.

Questo porta alla necessità di stabilire strumenti di comunicazione per consentire il corretto svolgimento del lavoro in team. Le comunicazioni si effettuano utilizzando il software **slack**² che, rispetto all'utilizzo delle classiche mail, consente scambi di messaggi rapidi all'interno di un contesto lavorativo tramite la creazione di appositi canali. Inoltre, come precedentemente accennato, viene usato **asana** per l'assegnazione e gestione dei compiti seguendo la filosofia *agile*. Ultimo, ma non meno importante, mezzo di comunicazione risulta essere **google meet**³. Una piattaforma di video chiamate fondamentale sia per le review settimanali, dovute all'applicazione del metodo *scrum*, che per qualsiasi delucidazione tra le persone del team.

Sviluppo. Il codice sorgente del progetto è gestito utilizzando la piattaforma **GitLab**. Il nome potrebbe far pensare al famoso **GitHub**, utilizzato da almeno un milione di sviluppatori nel mondo⁴, infatti sono abbastanza simili. Prima di elencare le differenze, occorre dare una panoramica generale su cosa siano le due piattaforme. Innanzitutto entrambe si basano su **Git**. Git è uno strumento *open source* per il controllo di versione⁵ distribuito. In un sistema di controllo di versione distribuito il client⁶, oltre ad avere l'ultima versione dei file, ha a disposizione l'intero repository⁷. In questo modo, se il server dovesse avere problemi, ogni persona che ha salvato il progetto in locale⁸ può poi ripristinarlo nel server⁹. Git, come illustrato in Figura 3.1, pensa ai dati come a una serie di *snapshot – istantanea* – di un filesystem in miniatura. Fondamentalmente scatta una fotografia di come appaiono i file in quel

¹Ministero del Lavoro e delle Politiche Sociali. *Smart working (Online)*, <https://www.lavoro.gov.it/strumenti-e-servizi/smart-working/Pagine/default>.

²Slack Technologies, LLC. *Slack: la tua piattaforma di produttività (Online)*, <https://slack.com/intl/it-it/>.

³Google. *Google meet: video chiamate e riunioni per tutti (Online)*, <https://meet.google.com/>.

⁴GitHub, Inc. *100 million developers and counting (Online)*, <https://github.blog/2023-01-25-100-million-developers-and-counting/>.

⁵Il controllo di versione è un sistema che salva le modifiche fatto a uno o più file nel tempo. In questo modo è possibile richiamare delle versioni specifiche nel tempo futuro. Decisamente più utile rispetto al semplice metodo del copia e incolla in un'altra cartella!

⁶Con il termine *client* si intende il computer del programmatore.

⁷**repository** <ripò ſitëri> s. ingl., usato in it. al masch. – Generico ambiente di storage, raggiungibile anche con un percorso web, dove vengono archiviati i pacchetti software che possono essere installati e aggiornati su un computer anche mediante operazioni programmate; Treccani S.P.A. *Repository. In Encyclopedia Treccani online: lessico del XXI Secolo (2013)*.

⁸Con il termine *in locale* si intende il disco fisico nel computer del programmatore.

⁹Con il termine *server* si intende l'elaboratore remoto.

momento, memorizzandoli solamente in caso il file sia stato modificato, altrimenti memorizza soltanto un collegamento al file già archiviato. Quindi un progetto è visto da Git come un flusso di snapshot.

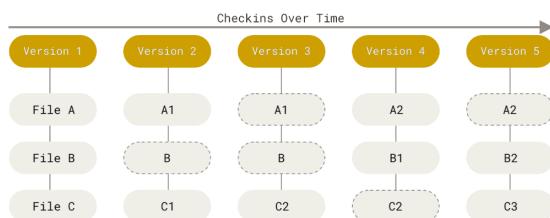


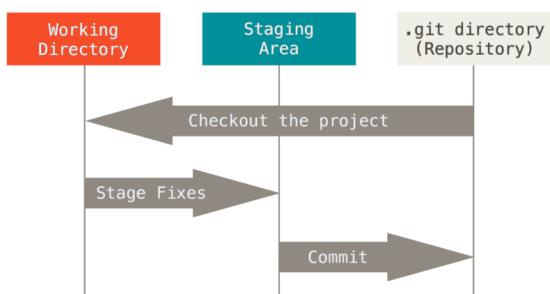
Figura 3.1: Flusso di snapshot.

Immagine rappresentante un esempio di flusso di snapshot effettuato da Git. I blocchi in giallo rappresentano le varie versioni nel tempo, mentre i blocchi in grigio i file del progetto. I contorni tratteggiati indicano un file rimasto invariato nel passaggio di versione.

Un progetto è un insieme di file. Ogni file Git può essere in uno dei seguenti tre stati:

- *Modified*: il file è stato modificato ma non è ancora stato inserito nel database.
- *Staged*: il file viene contrassegnato come modificato nella versione corrente, in modo da poter passare alla successiva.
- *Committed*: i dati vengono archiviati al sicuro nel database locale.

Ora che i termini principali sono stati chiariti, si può passare alla descrizione di un progetto Git. Esso si compone di tre sezioni (vedi Figura 3.2) principali: *working directory*, *staging area*, e *directory Git*.



Le tre sezioni principali di un progetto Git. Le frecce indicano i passi che devono essere fatti per passare da una parte all'altra.

Figura 3.2: Working directory (working tree), staging area e Git directory.

La working directory – *cartella di lavoro* –, anche chiamata working tree – *albero di lavoro* –, rappresenta i file di una singola versione del progetto, estratti dal database di Git e salvati in locale per essere poi modificati. La staging area – *area di stage* – è un file nel quale vengono salvate le informazioni per il prossimo commit. Infine la Git directory – *cartella .git* – è il cuore del progetto. Dentro la cartella Git salva tutti gli oggetti del database e i meta dati del progetto. Questo è il motivo per il quale essa viene copiata quando si *clona*¹⁰ un repository. Il flusso di lavoro Git consiste, bene o male, nei seguenti passi:

1. Modifica dei file nel *working tree*, mettendoli nello stato *modified*.
2. Selezionare i file modificati, stato *staged*, per il prossimo commit.
3. Eseguire i commit, presi dalla *staging area*, memorizzando lo snapshot permanentemente nella *git directory*.¹¹

Un ultimo aspetto che occorre evidenziare è il sistema di **branching**. Come detto in precedenza, Git salva le modifiche come snapshot. A partire dallo snapshot corrente è possibile creare un branch – *ramo* – separato. In questo modo, quando sarà effettuato un commit con le modifiche fatte, non andrà ad intaccare il ramo principale. In seguito, tramite l'operazione di **merge**, è possibile unire il branch separato al principale. Il branch creato di default è chiamato *master*¹². Questo sistema permette di avere sempre una versione stabile¹³.

Tornando al discorso principale, GitLab è una piattaforma web che consente di gestire repository Git. Tra le tante funzionalità, non rilevanti per questo testo, c'è la possibilità di creare il proprio repository sui propri server¹⁴ (*self-hosted*), questo è il motivo principale per cui nel progetto in questione viene usato esso anziché GitHub. Infatti il repository del progetto per la creazione di *Ismar Data* è ospitato sui server privati di Elan42¹⁵. La *repo*¹⁶ è gestita utilizzando il sistema di branching. In particolare, oltre al master branch, è presente un altro ramo fondamentale, chiamato

¹⁰ **clonare** v. tr. [der. di clone] (io clono, ecc.). **2. a.** Con uso fig., nel linguaggio dell'informatica, produrre una serie di copie identiche di un elemento di hardware o di software, con partic. riferimento ai calcolatori costruiti come imitazioni di altri di maggiore valore intrinseco e commerciale (ciascuna copia così ottenuta è detta clone); Treccani S.P.A. *Clonare*. In *Vocabolario Treccani online*.

¹¹ S. Chacon e B. Straub. *Pro Git: everything you need to know about Git*. Springer Nature, pp. 10–19.

¹² Ivi, pp. 63–104.

¹³ Con il termine *stabile* si intende funzionante e senza che vengano effettuate modifiche direttamente in essa.

¹⁴ GitLab B.V. *GitLab (Online)*, <https://about.gitlab.com/why-gitlab/#deployment>.

¹⁵ <https://repository.elan42.com/progetto-ismar/>

¹⁶ **repo**, abbreviazione informale di repository.

*develop*¹⁷. Il ramo master contiene la versione finale del progetto, mentre il ramo di *develop* contiene il codice in corso di sviluppo. Ogni volta che si aggiunge una funzionalità al progetto viene creato (per convenzione interna scelta dagli sviluppatori) un nuovo ramo chiamato */feature/<nome funzionalità da implementare>*. Durante la review del progetto, se ci sono feature branch completati, se ne effettua il merge in *develop* in attesa della convalida definitiva da parte del cliente che permetterà di spostare il tutto nel master branch. Un esempio di operazione di merge è mostrata in Figura 3.3.

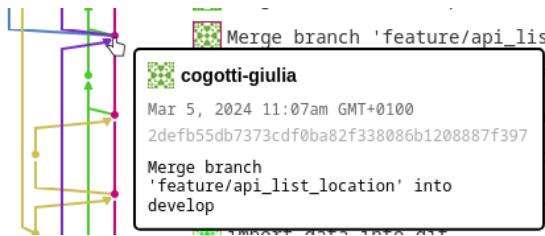


Figura 3.3: Merge di un feature branch in *develop*^a.

^aFonte: screenshot dell'autrice.

Flusso dei branch in GitLab. In particolare si evidenzia il merging del branch *feature/api_list_location* nel ramo di *develop*.

Ogni programmatore clona il progetto in locale e procede ad apportare le modifiche utilizzando l'editor che preferisce, spesso viene utilizzato **Visual Studio Code (VS Code)**. Questa risulta essere una delle scelte migliori perché supporta qualsiasi linguaggio di programmazione e, grazie al meccanismo dell'aggiunta di *estensioni*, permette di migliorare la produttività e avere una migliore esperienza utente. Un esempio può essere *Black Formatter*, estensione che permette di formattare il codice scritto in linguaggio Python tramite una semplice combinazione di tasti, senza che il programmatore debba preoccuparsene mentre scrive. Inoltre risulta essere utile quando più programmati mettono mano allo stesso file, tramite Black si evita di avere un formato visivo diverso che può creare conflitti quando si effettua un commit su Git. Un altro aspetto rilevante è la sua capacità di gestire i flussi di Git, sia a livello di riga di comando che a livello grafico. Infine VS Code è completamente gratuito e funziona su tutti i sistemi operativi di maggior utilizzo¹⁸.

¹⁷Quando l'applicazione entrerà in produzione sarà inserito anche un ramo di *staging* contenente la versione del progetto da presentare al cliente.

¹⁸Microsoft. *Visual Studio Code. Code editing. Redefined. (Online)*.

Tecnologie. Il progetto è sviluppato utilizzando il linguaggio di programmazione ad alto livello **python**. Python funziona su tutti i principali sistemi operativi (in particolare Linux, Unix, MAC OS X e Windows)¹⁹. Python fornisce una **libreria standard – standard library** – contenente moduli integrati (scritti in C) che forniscono accesso a funzionalità di sistema come I/O di file e altri moduli (scritti in Python) che implementano soluzioni per molti problemi che si verificano nella programmazione quotidiana, evitando che il programmatore debba implementarsi tutto da solo e permettendo quindi che si concentri sulle finalità del suo lavoro. Oltre alla standard library, è presente una raccolta di librerie create da vari sviluppatori e distribuite nella repository PyPI²⁰. L'applicazione visualizza dati che seguono alcuni formati standard. È possibile utilizzare alcune librerie di python che aiutano nella conversione dai formati grezzi a ciò che l'utente visualizzerà.

L'applicazione, lato back-end, si basa sul framework web **Django**. Django è gratis ed è open-source. Esso offre un aiuto agli sviluppatori per implementare applicazioni nel modo più veloce possibile. Django applica lo schema chiamato *Model-View-Controller (MVC)*. L'MVC è un modo di sviluppare software in modo che divide il progetto in diversi "livelli". Il codice per la definizione e accesso ai dati (M=model) viene separato dalla logica di instradamento delle richieste (C=controller), che a sua volta è separata dall'interfaccia utente (V=vista). Questo consente la modifica di una parte dell'applicazione in modo indipendente dalle altre parti²¹. In realtà, dato che la logica di instradamento delle richieste è gestita dal framework stesso, più che MCV risulta essere catalogato come *framework MTV*. In questo modello sono presenti tre livelli principali. Il primo si occupa dell'accesso ai dati (M=model), il secondo della presentazione di essi all'utente, quindi cosa e come visualizzarli (T=template), l'ultimo livello riguarda la logica di accesso ai modelli e come riferirsi al giusto template (V=view)²².

Il linguaggio utilizzato per prendere i dati dal back-end e passarli al front-end è **GraphQL**. GraphQL è quindi un linguaggio di interrogazione – *query* – per API. Tipicamente, le API in formato REST per prendere e passare i dati necessitano di fare molte chiamate, le API GraphQL invece effettuano una singola chiamata per dare all'applicazione tutto ciò di cui ha bisogno. Questo incrementa la velocità dell'applicazione anche se il dispositivo mobile ha una connessione ad internet lenta. Inoltre, le query GraphQL inviano solamente ciò che viene richiesto dall'applicazione. Questa funzionalità è sicuramente un vantaggio perché il client sa cosa ha richiesto e cosa ottiene, rendendo l'applicazione veloce e stabile. Ma non è finita

¹⁹Python Software Foundation, <https://wiki.python.org/moin/BeginnersGuide/Overview>.

²⁰ID. *Python Package Index - PyPI (Online)*, <https://pypi.org/>.

²¹Holovaty e Kaplan-Moss, cit., pp. 5–10.

²²Ivi, pp. 62–63.

qui! Considerando i diversi tipi di dati che devono essere passati, sarebbe veramente svantaggioso utilizzare un formato che struttura i dati tutti allo stesso modo. GraphQL fornisce invece adattabilità al contesto²³. L'interfaccia grafica **GraphiQL** consente di utilizzare un ambiente GraphQL direttamente su browser²⁴. La libreria `graphene` fornisce strumenti per implementare le API GraphQL direttamente in linguaggio python, senza dover scrivere utilizzando il linguaggio offerto da GraphQL²⁵. Grazie alla libreria python `graphene-django` è possibile includere GraphQL nel progetto, con anche l'interfaccia grafica a disposizione (vedi Listing 3.1). La libreria in questione è costruita sopra la semplice `graphene` e fornisce astrazioni che semplificano l'aggiunta delle funzionalità GraphQL al progetto Django. Quindi nel progetto sarà quest'ultima ad essere utilizzata (vedi Listing 3.1)²⁶.

```

1 INSTALLED_APPS = [
2     ...
3     "django.contrib.staticfiles", # Required for GraphiQL
4     "graphene_django",
5     ...
6 ]

```

Listing 3.1: settings.py

Specificando l'indirizzo a cui accedere (vedi Listing 3.2) si ottiene l'interfaccia grafica di GraphiQL in Figura 3.4, che rispecchierà i modelli e le query specificate nel codice python²⁷.

```

1 from graphene_django.views import GraphQLView
2 from django.views.decorators.csrf import csrf_exempt
3
4 urlpatterns = [
5     path("graphql/", csrf_exempt(GraphQLView.as_view(graphiql=True
6         ))),
7     ...
7 ]

```

Listing 3.2: urls.py

L'applicazione, lato back-end, dispone di un database **postgreSQL**. Esso segue il *modello relazionale*. È importante evidenziare che esso non sarà descritto con pre-

²³The GraphQL Foundation, cit.

²⁴GraphQL Contributors. *GraphiQL & the GraphQL LSP Reference Ecosystem for building browser & IDE tools (Online)*, <https://github.com/graphql/graphiql?tab=readme-ov-file#graphiql>.

²⁵P. Arminio et al. *Graphene. GraphQL in Python Made Easy (Online)*, <https://docs.graphene-python.org/en/latest/quickstart/>.

²⁶Ivi, <https://docs.graphene-python.org/projects/django/en/latest/>.

²⁷infra, pp. 59-62.

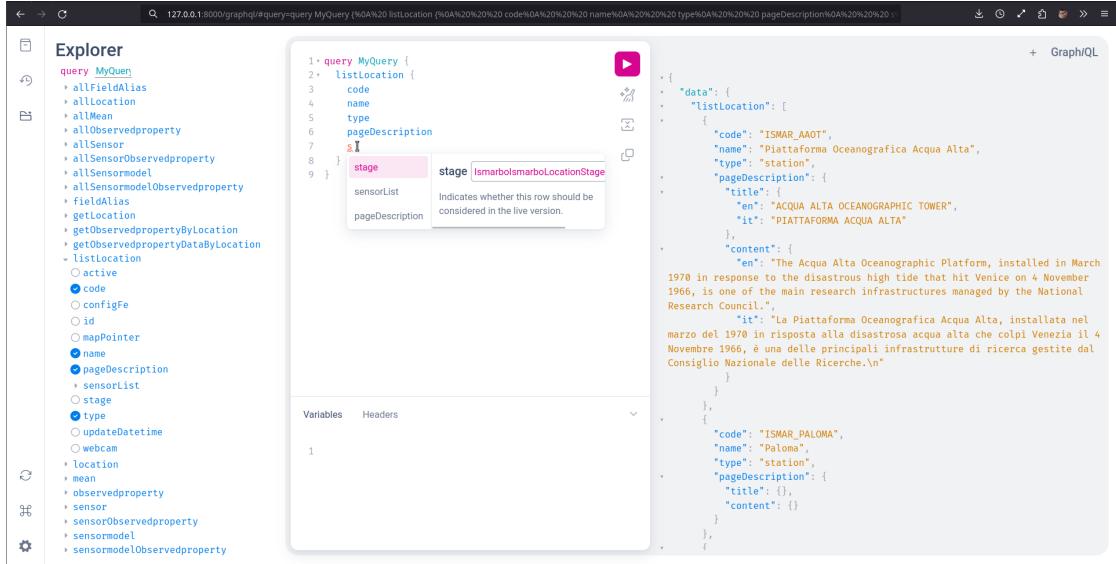


Figura 3.4: Interfaccia grafica GraphiQL. Al centro l'editor per le query, a destra il risultato dell'esecuzione della query e a sinistra la lista delle query costruite in python e integrate con la libreria graphene-django^a.

^aFonte: screenshot dell'autrice.

cisione, ne verrà data solo un'idea per chiarire alcuni termini che saranno utilizzati nei successivi capitoli. Lo schema è costituito da *entità* (tabelle) e *relazioni*. Ogni tabella contiene record (detti anche righe o n-uple) differenti. Ogni riga è formata da un certo numero di elementi. Questo numero deve essere uguale per tutti i record della tabella. Ogni n-upla deve essere distinta dalle altre, almeno per un campo, chiamato *chiave*. Esistono tre operatori principali: *selezione* per l'estrazione dei dati, *proiezione* per la scelta di certi campi e *giunzione*. Quest'ultimo è fondamentale per l'unione di due tabelle data la chiave in comune.²⁸

SQL²⁹ è un linguaggio per database relazionali. SQL fornisce sia funzionalità DML (*Data Manipulation Language*), ovvero modifica e interrogazione ad un database, che di DDL (*Data Definition Language*), che consiste in comandi per definire lo schema di un database relazionale. Le tabelle rispecchiano il contenuto di quelle del modello, quindi gli elementi sono gli stessi. In particolare le colonne della tabel-

²⁸R. Doretti. *Data base. Concetti e disegno*. Gruppo Editoriale Jackson, pp. 28–40.

²⁹Inizialmente acronimo di *Structured Query Language*, ma ormai considerato come un nome proprio. Vi sono due pronunce analoghe anglosassoni: "es-que-el" (in italiano "esse-qu-elle") e, come se fosse una parola intera, "sequel".

la vengono chiamati attributi. Qui il concetto di chiave assume molta importanza. È obbligatorio specificare una *chiave primaria –primary key (PK)* – che identifica univocamente una riga della tabella. Ultimo aspetto da evidenziare è il vincolo di *chiave esterna – foreign key (FK)* – che crea un legame tra i valori di un attributo della tabella su cui è definito e i valori di un attributo in un'altra tabella che è la chiave primaria di tale tabella. In realtà potrebbe accadere anche che i due attributi si trovano nella tabella stessa ma, ai fini della costruzione del database del progetto, questa situazione non è rilevante in quanto essa non si presenta.³⁰.

PostgreSQL è quindi un sistema di database relazionale a oggetti open source che usa ed estende il linguaggio SQL con molte funzionalità mirate ad aiutare gli sviluppatori nel costruire applicazioni e proteggere i dati. PostgreSQL funziona su tutti i sistemi operativi di maggior utilizzo e supporta numerose estensioni³¹. Una di queste è *PostGIS* che estende le capacità di PostgreSQL aggiungendo supporto per la gestione dei dati geo spaziali. È citato qua perché il salvataggio dei dati meteorologici in un database postGIS è una strada che si potrebbe percorrere appena si avranno certezze sui dati³².

3.2 Importazione dati

La fase di import è l'operazione che consiste nell'acquisizione dei dati dalle sorgenti e nella loro elaborazione al fine di renderli utilizzabili dall'applicazione.

Sorgenti di dati pubbliche. I dati pubblici riguardano le osservazioni da satellite e i radar. Questi dati sono tutti contenuti in cataloghi THREDDS. L'accesso ad essi può avvenire tramite diversi protocolli (vedi lista in Figura 3.5), quelli di interesse per *Ismar Data* sono **WMS** e **NetcdfSubset**.

Tutti i protocolli sono basati sull'architettura client-server. Lo schema classico del modello è presente in Figura 3.6. Questa architettura descrive un modello nel quale un computer, detto client, richiede delle risorse a un altro elaboratore, chiamato server, attraverso una connessione di rete. Il server riceve la richiesta, la elabora e risponde al client. Potrebbe accadere che ci siano più elaboratori, sia da un lato che dall'altro. Inoltre, spesso allo schema base viene aggiunto un database dove il

³⁰P. Atzeni et al. *Basi di dati*. McGraw-Hill Education, pp. 89–100.

³¹The PostgreSQL Global Development Group, cit., <https://www.postgresql.org/about/>.

³²PostGIS PSC & OSGeo. *PostGIS (Online)*, <https://postgis.net/>.

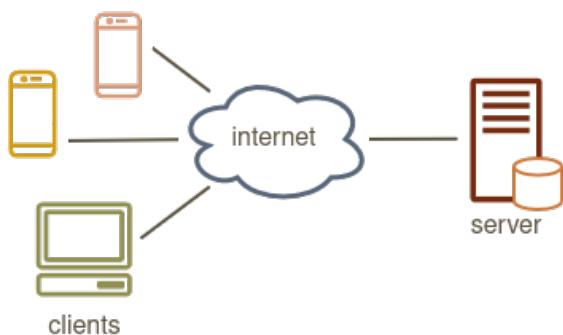
Access:

1. **OPENDAP:** [/thredds/dodsC/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/dodsC/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
2. **DAP4:** [/thredds/dap4/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/dap4/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
3. **WCS:** [/thredds/wcs/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/wcs/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
4. **WMS:** [/thredds/wms/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/wms/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
5. **NetcdfSubset:** [/thredds/nccs/grid/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/nccs/grid/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
6. **CdmRemote:** [/thredds/cdmremote/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/cdmremote/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
7. **ISO:** [/thredds/iso/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/iso/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
8. **NCML:** [/thredds/ncl/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/ncl/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)
9. **UDDC:** [/thredds/uddc/EUHFR_NRTcurrent_HFR-TirLig-Total_v3](https://thredds/uddc/EUHFR_NRTcurrent_HFR-TirLig-Total_v3)

Figura 3.5: Accesso dati HFR-TirLig^a.

^aFonte: ISMAR-CNR. *HFR-TirLig_catalog (Dataset)*, https://thredds.hfrnode.eu:8443/thredds/NRTcurrent/HFR-TirLig/HFR-TirLig_catalog.html?dataset=EUHFR_NRTcurrent_HFR-TirLig-Total_v3.

Lista di protocolli per l'accesso ai dati del catalogo thredds della rete HFR-TirLig. I dataset di CHLa e SST hanno a disposizione solo OPENDAP, WMS e NetcdfSubset.



Schema classico del modello client-server dove essi comunicano attraverso la rete.

Figura 3.6: Modello client-server (Diagramma creato utilizzando drawio^a).

^aJGraph Ltd. *Flowchart Maker & Online Diagram Software - Draw.io (Online)*, <https://www.draw.io/>.

server conserva dati e informazioni. L'architettura client-server è uno dei modelli più utilizzato quando si tratta di richieste nelle quali compaiano i due attori³³.

Il **WMS** (*Web Map Service Interface Standard*) fornisce mappe di dati con riferimenti spaziali in modo dinamico a partire da informazioni geografiche. Per mappa

³³S. Mbuguah, V. Mony e G. Nyabuto. «Architectural Review of Client-Server Models». In: *International Journal of Scientific Research and Engineering Trends*, pp. 139–142.

non si intende i dati veri e propri, ma una rappresentazione sotto forma di immagine digitale adatta ad essere visualizzata in uno schermo. Questa immagine può essere in formato pdf, GIF o JPEG, o addirittura SVG (*Scalable Vector Graphics*)³⁴. Sono possibili diverse operazioni, di cui due fondamentali. La prima *GetCapabilities* serve ad ottenere metadati (i metadati sono informazioni strutturate che descrivono e rendono più facile l'accesso alle informazioni) sul servizio, insieme alle possibili operazioni e alla lista dei *layer* disponibili. Ogni mappa disponibile è un layer con un nome a cui riferirsi. Per ottenere l'immagine della mappa si utilizza la seconda operazione *GetMap*, specificando il layer e il formato nei parametri³⁵.

NetCDF – *network Common Data Form* – è un modello per dati scientifici orientati agli array. Esistono diverse librerie distribuite gratuitamente che implementano il supporto a questo modello per la creazione, l'accesso e la condivisione dei dati. Le raccolte di dati create con NetCDF includono sempre informazioni su cosa contengono e sono accessibili velocemente da qualsiasi dispositivo. Inoltre, un archivio rende possibile l'accesso a tutte le versioni³⁶ create. Il modello classico rappresenta le informazioni scientifiche in un dataset netCDF usando dimensioni, variabili e attributi. Le variabili contengono i dati salvati come array multidimensionali aventi lo stesso tipo di forma. La forma è intesa come una lista da zero ad n dimensioni (0 rappresenta una variabile con un singolo valore, 1 rappresenta un vettore, 2 rappresenta una matrice, detta anche griglia, e così via fino ad N). Le dimensioni sono inoltre usate per le griglie comuni e per il sistema di coordinate. Gli attributi invece contengono metadati, ovvero informazioni sulle proprietà di una variabile o dell'intero dataset³⁷.

NetCDF Subset Service (NCSS) è un servizio web per un sottoinsieme di dati NetCDF. Gli array di dati vengono suddivisi in sottoinsiemi, specificano le coordinate terrestri (per es. utilizzando latitudine/longitudine) oppure intervalli di date, preservando comunque la risoluzione e l'accuratezza del set di dati originale. Un dataset in questo servizio corrisponde a un documento XML e fornisce abbastanza dettagli per rendere possibile una richiesta da parte del client. I dataset dei cataloghi thredds sono tutti una collezione di griglie. Ciò significa che hanno coordinate orizzontali (x ed y), optionalmente anche verticali, hanno il tempo e un insieme di coordinate³⁸. Nei cataloghi analizzati, la risposta del server produce un file netCDF4.

³⁴Open Geospatial Consortium Inc. *OpenGIS®Web Map Server Implementation Specification*. A cura di J. de la Beaujardiere, p. 5.

³⁵Ivi, pp. 21–38.

³⁶Id. *OGC Network Common Data Form (NetCDF)*. a cura di B. Domenico, p. 7.

³⁷Ivi, pp. 10–15.

³⁸Unidata/UCAR Community Programs (UCP), cit., https://docs.unidata.ucar.edu/tds/current/userguide/netcdf_subset_service_ref.html.

Dalla descrizione si evidenzia come il formato WMS sia particolarmente comodo per ottenere un'immagine senza troppi sforzi, mentre netCDFsubset necessita del salvataggio di un file contenente l'intero dataset che deve poi essere elaborato per produrre un qualcosa di concreto. Contrariamente a quanto sembra, netCDFsubset è la scelta migliore. Questo perché, tramite la libreria **gdal** (*Geospatial Data Abstraction Library*), disponibile anche per Python, è possibile tradurre facilmente i formati di dati geospaziali³⁹. WMS non è però da escludere, infatti viene momentaneamente utilizzato per plottare⁴⁰ i dati nella demo web.

Le funzioni che si occupano di interagire con il server dei cataloghi thredds e della elaborazione dei dati raccolti, seguono il principio del riuso del codice, sfruttando ciò che già esiste per tutti i cataloghi (radar, Chla, SST). Ora che questo è chiaro si può procedere all'analisi dell'implementazione dell'import, prima netCDFsubset e in seguito WMS

La procedura "etl" in Listing 3.3 contiene tutte le funzionalità per scaricare i dati dai cataloghi e convertirli tramite gdal. Inizialmente quindi i dati sono scaricati dal catalogo con una funzione (vedi Listing 3.4) che, dato il link al file netCDF sul server, lo scarica in una cartella predefinita, controllando che il server dia un codice di risposta positivo.

```

1  def etl(day: str, dataset: str, variable: str,
2         add_variable_to_filepath: bool = False):
3      ...
4      # url for each thredds catalog
5      # ncss protocol used
6      # accept=netcdf output format
7      # download dai cataloghi thredds
8      if dataset == "sst":
9          url = f"http://ritmare.artov.ismar.cnr.it/thredds/ncss/
10             sstuhr?var=\{variable\}&disableLLSubset=on&disableProjSubset=
11             on&horizStride=1&time=\{day\}&addLatLon=true&accept=netcdf"
12      elif dataset == "chl":
13          url = f"http://ritmare.artov.ismar.cnr.it/thredds/ncss/
14             xchlcse12?var=\{variable\}&disableLLSubset=on&
15             disableProjSubset=on&horizStride=1&time=\{day\}&addLatLon=
16             true&accept=netcdf"
17      elif dataset == "hfr":
18          url = f"https://thredds.hfrnode.eu:8443/thredds/ncss/EUHFR
19             \_NRTcurrent\_HFR-TirLig-Total\_v3?var=\{variable\}&north
20             =44.4960&west=7.5069&east=10.4930&south=43.2539&
```

³⁹GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*, <https://gdal.org/>.

⁴⁰**plottare** v. tr. [adattam. dell'ingl. *(to) plot* «tracciare una mappa, un piano», con influsso diretto di *plotter*]. – Nel gergo anglicizzante dell'informatica, disegnare, tracciare un diagramma mediante il *plotter*; Treccani S.P.A. *Plottare*. In *Vocabolario Treccani online*.

```
disableLLSubset=on\&disableProjSubset=on\&horizStride=1\&time
=\{day\}\&vertCoord=0"
13   else:
14     logging.error(f"Wrong input as dataset: \{dataset\}")
15     raise ValueError()
16
17   print(f"URL: {url}")
18   temp_path = tempfile.gettempdir() + f"/{dataset}_{day}.nc"
19
20   try:
21     # scaricamento file netcdf nella cartella
22     download_successful = download_netcdf(url, temp_path)
23     if not download_successful:
24       raise DownloadError("Download failed.")
25
26     if add_variable_to_filepath:
27       destination_path = os.path.join(
28         BASE_DIR, "ismarbo/static/ismarbo/thredds",
29         dataset, variable
30       )
31     else:
32       destination_path = os.path.join(
33         BASE_DIR, "ismarbo/static/ismarbo/thredds",
34         dataset
35       )
36     timestamp = get_time(temp_path)
37
38     destination_file = os.path.join(
39       destination_path,
40       f"\{timestamp}.tif",
41     )
42
43     # conversione dati tramite gdal
44     conversion_successful = convert_nc_to_geotiff(
45       temp_path,
46       destination_file,
47       variable,
48     )
49
50     if not conversion_successful:
51       raise ConversionError("Conversion failed.")
52
53   return timestamp
54
55 except (DownloadError, ConversionError, DatabaseLoadError,
ValueError) as e:
  logging.error(f"Process terminated: {e}")
```

```
56     logging.info("All operations completed successfully.")
```

Listing 3.3: Porzione della procedura utilizzata per scaricare, convertire e salvare i dati dei cataloghi.

```
1 def download_netcdf(url, temp_path)
```

Listing 3.4: Definizione funzione per scaricare in local un file netCDF dato il link al server.

Successivamente avviene l'operazione principale: la conversione dei file netcdf ottenuti in formato **GeoTIFF** (*Geographic Tagged Image File Format*) tramite gdal. Per comprendere cosa sia GeoTIFF occorre analizzare **TIFF**. TIFF (*Tagged Image File Format*) è un formato di file basato su tag per salvare immagini (in bianco e nero, in scala di grigi e a colori) raster. Data la sua indipendenza dall'architettura del computer, sia lato software che lato hardware, e la sua adattabilità ad essere aperto da qualsiasi programma, è molto utilizzato. GeoTIFF è quindi definito come un insieme di tag TIFF che descrivono tutte le informazioni geografiche, in questo modo i dati possono essere collegati a una proiezione di una mappa⁴¹. Tutto questo viene fatto all'interno della funzione "*convert_nc_to_geotiff*" (vedi Listing 3.5) che, dato il file netcdf, il percorso in cui salvare il risultato ottenuto, la variabile di interesse (per i dataset composti da più variabili, come i radar), e il sistema di coordinate standard utilizzato, converte il dataset di partenza in geoTIFF attraverso:

- `gdal.Open(nc_file)` : apre un dataset netCDF⁴²;
- `gdal.Translate(output_tiff, nc_dataset, options=translate_options)` : converte il dataset in formato geoTIFF, salvando il file TIFF⁴³.

```
1 def convert_nc_to_geotiff(nc_file, output_tiff, variable_name=None, band=1, crs="EPSG:4326")
```

Listing 3.5: Definizione funzione per la conversione dei file netCDF in geoTIFF.

L'ultima importante funzione "*load_geotiff_to_postgis*" (vedi Listing 3.6) è quella che consente di salvare i dati GeoTIFF nel database PostgreSQL utilizzando le funzionalità offerte dal plugin postGIS. Al momento è stata solamente progettata per usi futuri, appena sarà chiaro se salvare tutto (dati statici e dinamici) all'interno dello stesso database o se separare queste tipologie di dati.

⁴¹S. Sazid Mohammad e R. Ramakrishnan. «GeoTIFF - A standard image file format for GIS applications». In: *Map India - Image Processing & Interpretation*, pp. 1–4.

⁴²GDAL/OGR contributors, cit., https://gdal.org/tutorials/raster_api_tut.html#opening-the-file.

⁴³Ivi, https://gdal.org/programs/gdal_translate.html#gdal-translate.

```

1 def load_geotiff_to_postgis(geotiff_path, database, table,
2     host="localhost", port=5432, user="your_username", password=
3         "your_password")

```

Listing 3.6: Definizione funzione per il salvataggio dei dati GeoTIFF all'interno del database PostgreSQL.

I dati devono essere aggiornati periodicamente, eliminando i vecchi dati e sostituendoli con gli ultimi disponibili. Questo viene fatto tramite il meccanismo dei **cron-job**. Cron è lo scheduler dei lavori basato sul tempo nei sistemi operativi Unix-like per consentire l'esecuzione di lavori – *job* – periodicamente in determinati orari, date o intervalli⁴⁴. L'implementazione (vedi Listing 3.7) sul progetto è semplificata grazie a **django-cron** che evita la complicata costruzione di uno script Python apposito o l'utilizzo di un comando di gestione per cron (ancora peggio!)⁴⁵.

```

1 CRONJOBS = [
2     (
3         "0 5 * * *", # 5:00am
4         "ismarbo.tasks.sst_cron_etl",
5         f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/sst/
6 logs/cronlogs.log 2>&1",
7     ),
8     (
9         "30 5 * * *", # 5:30am
10        "ismarbo.tasks.chl_cron_etl",
11        f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/chl/
12 logs/cronlogs.log 2>&1",
13     ),
14     (
15         "4 */1 * * *", # al minuto 4 ogni ora
16         "ismarbo.tasks.hfr_cron_etl",
17         f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/hfr/
18 logs/cronlogs.log 2>&1",
19     ),
20     (
21         "0 3 * * *", # 3:00am
22         "ismarbo.tasks.sst_cleanup",
23         f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/sst/
24 logs/cleanuplogs.log 2>&1",
25     ),
26     (
27         "10 3 * * *", # 3:10am
28         "ismarbo.tasks.chl_cleanup",
29         f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/chl/
30 logs/cleanup.log 2>&1",

```

⁴⁴Maintenance Team. *Arch Wiki (Online)*, <https://wiki.archlinux.org/title/Cron>.

⁴⁵Tivix Inc. *Django-cron's documentation (Online)*, <https://django-cron.readthedocs.io/en/latest/introduction.html>.

```

26     ) ,
27     (
28         "20 3 * * *", # 3:20am
29         "ismarbo.tasks.hfr_cleanup",
30         f">> {BASE_DIR}/ismarbo/static/ismarbo/thredds/hfr/
31         logs/cleanup.log 2>&1",
32         ),
33     ]

```

Listing 3.7: Definizione dei cron job per i dati dei cataloghi thredds. Lo scaricamento dei dati avviene chiamando le funzioni `sst_cron_etl`, `chl_cron_etl` e `hfr_cron_etl`, mentre l'eliminazione tramite le rispettive `cleanup`.

L'accesso ai dati tramite protocollo WMS avviene solamente per la demo web grazie a **Leaflet**, un esempio è mostrato in Figura 3.7. Leaflet è una libreria JavaScript open-source per visualizzare mappe interattive in modo semplice e veloce. Inoltre supporta il sistema di coordinate utilizzato dai cataloghi, altrimenti non sarebbe di alcuna utilità⁴⁶. Leaflet utilizza l'url base, ovvero senza indicare alcun metodo, del servizio WMS per creare una mappa. Specificando il layer di interesse e le varie opzioni si ottiene immediatamente l'immagine da visualizzare (vedi codice in Listing 3.8 e Listing 3.9). Ai fini della visualizzazione, è importante notare che il nome del layer da utilizzare non è quello indicato dal documento XML GetCapabilites di WMS. Per ottenere il giusto nome del layer è consigliato utilizzare un qualsiasi programma di visualizzazione di WMS, come QGis⁴⁷. Ad esempio in Listing 3.8 il nome del layer ottenuto tramite QGis risulta essere `"analysed_sst"`, mentre il nome indicato dal documento XML è `"sst"`.

```

1  var overlayMaps = {
2      "Sea Surface Temperature": L.tileLayer
3          .wms("http://ritmare.artov.ismar.it/thredds/wms/
4          sstuhr", {
5              layers: "analysed_sst",
6              format: "image/pdf",
7              transparent: true,
8              version: "1.3.0",
9              attribution: "CNR THREDDS Catalog for RITMARE
10             Project - Sea Surface Temperature",
11              TIME: "2014-01-01T00:00:00.000Z",
12          })
13      ...

```

Listing 3.8: Utilizzo di Leaflet per ricavare l'immagine della mappa per la variabile SST, a cui saranno aggiunte funzionalità per l'interazione

⁴⁶V. Agafonkin. *Leaflet: an open-source JavaScript library for mobile-friendly interactive maps (Online)*, <https://leafletjs.com/index.html>.

⁴⁷Ivi, <https://leafletjs.com/examples/wms/wms.html>.

```

1 ...
2 Chlorophyll: L.tileLayer
3   .wms("http://ritmare.artov.ismar.cnr.it/thredds/wms/
4   xchlcase12", {
5     layers: "CHL",
6     format: "image/pdf",
7     transparent: true,
8     attribution: "CNR THREDDS Catalog for RITMARE Project
- Chlorophyll",
9     TIME: "2014-01-01T00:00:00.000Z",
10    })
11 ...

```

Listing 3.9: Utilizzo di Leaflet per ricavare l'immagine della mappa per la variabile Chla, a cui saranno aggiunte funzionalità per l'interazione

Il layer può poi essere sovrapposto ad una mappa base del territorio (vedi codice in Listing 3.10). Una mappa base è una mappa di riferimento alla quale è possibile sovrapporre più livelli di dati geografici, in particolare per la demo sono utilizzate quelle fornite da CARTO⁴⁸. L'effetto di queste operazioni è mostrato in Figura 3.7, dove, nella prima schermata è presente un menu dal quale è possibile selezionare i layer da sovrapporre alla mappa base CARTO. Le successive due schermate rappresentano rispettivamente il layer di sst e CHLa, selezionato dal menu.

```

1 // inizializzazione mappa
2 // impostazione della sua visualizzazione sulle coordinate
3 // geografiche scelte
4 var map = L.map("map").setView([44, 13], 7);
5
6 // aggiunta delle basemaps
7 var baseLayers = {
8   CARTO: L.tileLayer(
9     "https://{s}.basemaps.cartocdn.com/light_nolabels/{z}
10    /{x}/{y}{r}.pdf",
11    {
12      attribution: 'Map data © <a href="https://www
13      .openstreetmap.org/">OpenStreetMap</a> contributors, <a href="
14      https://carto.com/attribution">CARTO</a>',
15    }
16  )
17 ...

```

Listing 3.10: Utilizzo di Leaflet per ottenere una mappa base, fornita da CARTO, per le coordinate specificate.

⁴⁸CARTO. *CARTO basemaps (Online)*, <https://carto.com/basemaps>.

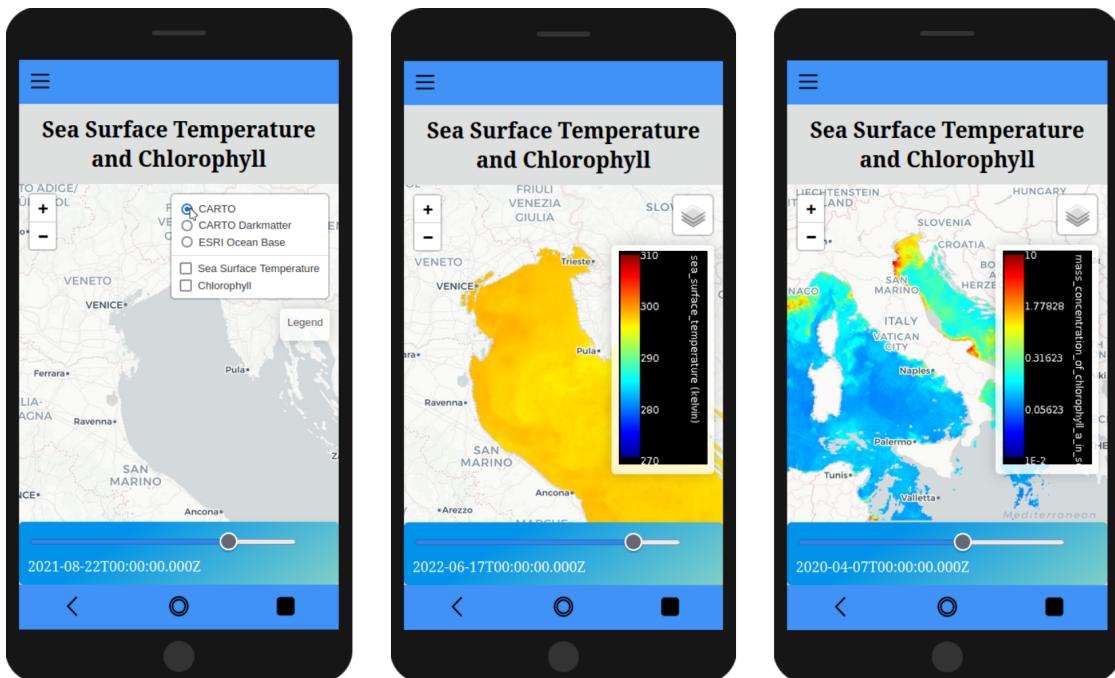


Figura 3.7: Schermate della demo web che evidenziano il funzionamento dei layer prelevati dal server WMS per temperatura del mare e clorofilla, con relativo posizionamento al di sopra di una mappa base^a.

^aFonte: screenshot dell'autrice.

La visualizzazione dei dati rilevati dai radar sarà dinamica, tramite un flusso di frecce in movimento ad indicare direzione e velocità del mare. Un esempio, tramite istantanee, è mostrato in Figura 3.8.

Sorgenti di dati private. La fase di importazione, per le sorgenti private, risulta essere particolarmente critica. Questo deriva dal problema, ampiamente discusso in questa tesi, riguardante la mancanza di dati. È opportuno ricordare che le sorgenti private riguardano i modelli di previsione e le stazioni osservative fisse, escludendo i radar che, come si è visto, hanno dati pubblici in cataloghi thredds.

Per quanto riguarda i modelli di previsione, quindi Nettuno ed Henetus, il cliente ha fornito dei file contenenti dati di esempio di periodi passati. Non avendo un accesso continuo ai dati non è possibile sviluppare una strategia di import. L'unica funzione sviluppata riguarda la conversione dei file grib di Nettuno in geoTIFF tra-

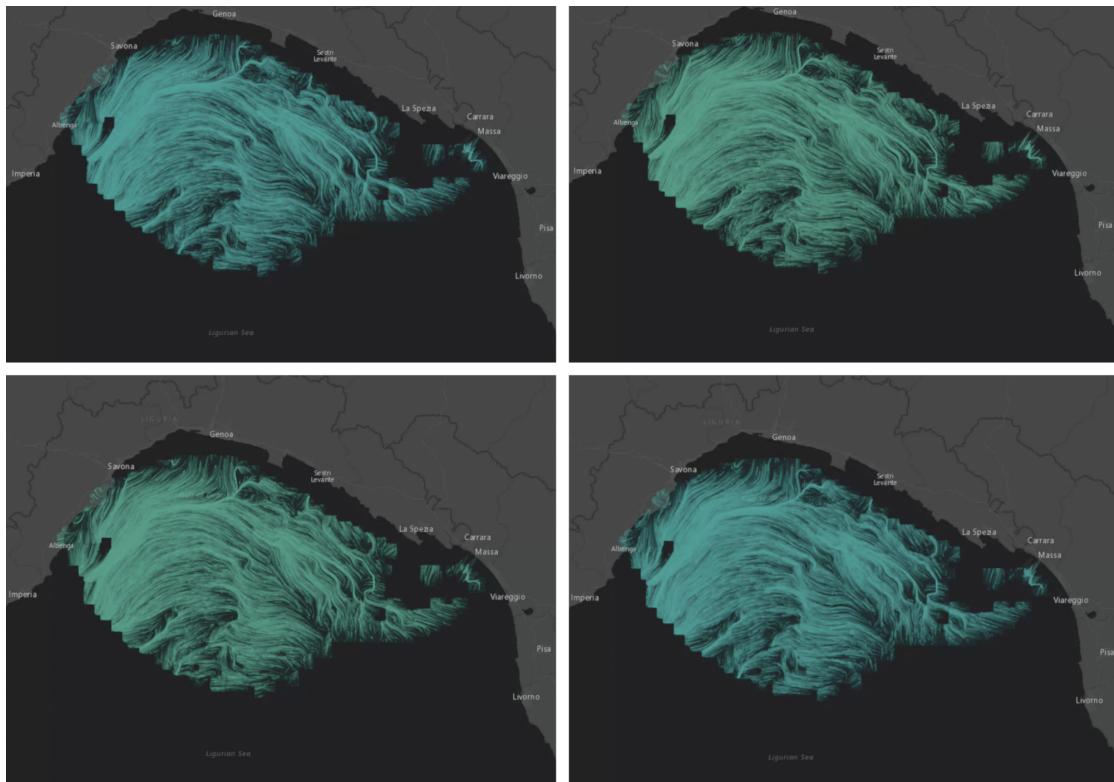


Figura 3.8: Instantanee del flusso dei dati rilevati dai radar nella giornata del 30 maggio 2024 alle ore 12:15 (la diversa tonalità di colore viene usata per leggibilità)^a.

^aFonte: screenshot dell'autrice.

mite la libreria gdal (vedi Listing 3.11), seguendo il modello utilizzato per i cataloghi thredds.

```
1 def convert_grib_to_geotiff(input_file, output_file):
```

Listing 3.11: Definizione funzione per convertire un file grib in geoTIFF.

Invece, per le stazioni osservative fisse, non si è ancora implementata una vera e propria strategia generale di import, perché le uniche variabili disponibili sarebbero le sei della piattaforma Acqua Alta. Per quanto riguarda le altre stazioni, Paloma, S1-GB ed E1, non si è ancora giunti ad un accordo per l'accesso ai dati. Quindi ora saranno analizzate le modalità di accesso alle API, di allMeteo e del comune di Venezia, per le variabili della piattaforma Acqua Alta e la conseguente elaborazione dei dati ottenuti.

Vengono implementate due funzioni differenti per effettuare le richieste alle API di allMeteo (intensità e direzione vento) e del comune di Venezia (livello del mare, altezza d'onda, direzione e intensità del vento). Quando si fa una richiesta usando l'API allmeteo occorre specificare un token per l'accesso, il sensore da cui prelevare i dati e infine un intervallo di date (vedi funzione in Listing 3.12).

```

1  def fetch_aaot_wind(
2      token: str,
3      from_time: datetime,
4      to_time: datetime,
5      devices: List[str] = ["2101LW013"],
6  )

```

Listing 3.12: Definizione funzione per ricavare i dati di intensità e direzione vento dall'api allMeteo.

Il risultato sarà lo storico, in formato json, delle misurazioni (effettuate ogni dieci minuti a partire dalle 00:07) nell'arco di tempo specificato. In Listing 3.13 è presente un esempio dei risultati ottenuti richiedendo i dati dal 28 maggio al 30 maggio 2024 alle ore 00:07 (dai diversi test effettuati si è notato che l'intervallo inizia e termina due ore prima dell'orario specificato). Oltre alla data e ora della misurazione, sono presenti alcune informazioni sul sensore, come il numero di serie, il numero identificativo e il livello di batteria. E poi ci sono i dati reali di intensità del vento (wind_ave10, wind_max10, wind_min10) e la direzione (dir_ave10, dir_max10, dir_hi10, dir_lo10)

```

1 [
2     {
3         "sn": "2101LW013",
4         "timestamp": "2024-05-27 22:07:22",
5         "device_id": 31810,
6         "battery": "4.1",
7         "wind_ave10": "0.98950344",
8         "wind_max10": "3.742631",
9         "wind_min10": "0",
10        "dir_ave10": 336,
11        "dir_max10": 342,
12        "dir_hi10": 353,
13        "dir_lo10": 319
14    },
15    {
16        "sn": "2101LW013",
17        "timestamp": "2024-05-27 22:17:22",
18        "device_id": 31810,
19        "battery": "4.1",
20        "wind_ave10": "1.8389517",
21        "wind_max10": "4.164245",
22        "wind_min10": "0",
23        "dir_ave10": 329,

```

```

24     "dir_max10": 64,
25     "dir_hi10": 65,
26     "dir_lo10": 232
27 },
28 ...
29 {
30     "sn": "2101LW013",
31     "timestamp": "2024-05-29 21:47:23",
32     "device_id": 31810,
33     "battery": "4.1",
34     "wind_ave10": "1.6267841",
35     "wind_max10": "5.73184",
36     "wind_min10": "0",
37     "dir_ave10": 309,
38     "dir_max10": 240,
39     "dir_hi10": 13,
40     "dir_lo10": 245
41 }
42 ]

```

Listing 3.13: Porzione di storico delle misurazioni di direzione e intensità del vento effettuate nel periodo tra il 28 maggio e il 30 maggio 2024 dal sensore Davis Vantage PRO posto sulla piattaforma Acqua Alta. I dati vengono richiesti all'API allmeteo.

Nella richiesta alle API del comune di Venezia occorre specificare le credenziali, l'intervallo di date e le variabili d'interesse. La risposta ottenuta, in formato json, contiene le misurazioni effettuate ogni cinque minuti. La funzione (vedi Listing 3.14) che ricava i dati richiede solo il periodo di interesse; la logica dell'accesso tramite credenziali e certificato TLS è implementata all'interno.

```
1 def fetch_marea(day_0, day_f)
```

Listing 3.14: Definizione funzione per ricavare i dati di livello del mare, altezza d'onda, direzione e intensità del vento.

In Listing 3.15 è presente un esempio dei dati delle quattro variabili (livello del mare, altezza onde, direzione e intensità del vento) nell'intervallo dal 23 maggio al 24 maggio 2024. Se non viene specificata alcuna data si ottengono soltanto le misurazioni effettuate nei 10 minuti precedenti all'orario corrente.

```

1 [
2   {
3     "DATA": "2024-05-23 00:00:00",
4     "piattaforma acqua alta siap - livello - metri": "0.56",
5     "piattaforma acqua alta siap - onda_altezza_significativa
- metri": "0.13",
6     "piattaforma acqua alta siap - vento_velocita - m/s": "
2.00",

```

```

7     "piattaforma acqua alta siap - vento_direzione - Gradi": "324.00"
8   },
9   {
10     "DATA": "2024-05-23 00:05:00",
11     "piattaforma acqua alta siap - livello - metri": "0.54",
12     "piattaforma acqua alta siap - onda_altezza_significativa
- metri": null,
13     "piattaforma acqua alta siap - vento_velocita - m/s": "1.60",
14     "piattaforma acqua alta siap - vento_direzione - Gradi": "311.00"
15   },
16   {
17     "DATA": "2024-05-23 00:10:00",
18     "piattaforma acqua alta siap - livello - metri": "0.53",
19     "piattaforma acqua alta siap - onda_altezza_significativa
- metri": null,
20     "piattaforma acqua alta siap - vento_velocita - m/s": "1.50",
21     "piattaforma acqua alta siap - vento_direzione - Gradi": "316.00"
22   },
23   ...
24   {
25     "DATA": "2024-05-23 23:55:00",
26     "piattaforma acqua alta siap - livello - metri": "0.65",
27     "piattaforma acqua alta siap - onda_altezza_significativa
- metri": null,
28     "piattaforma acqua alta siap - vento_velocita - m/s": "4.70",
29     "piattaforma acqua alta siap - vento_direzione - Gradi": "236.00"
30   },
31   {
32     "DATA": "2024-05-24 00:00:00",
33     "piattaforma acqua alta siap - livello - metri": "0.63",
34     "piattaforma acqua alta siap - onda_altezza_significativa
- metri": "0.19",
35     "piattaforma acqua alta siap - vento_velocita - m/s": "5.00",
36     "piattaforma acqua alta siap - vento_direzione - Gradi": "231.00"
37   }
38 ]

```

Listing 3.15: Porzione delle misurazioni di livello del mare, altezza d'onda, direzione e intensità del vento, effettuate nel periodo tra il 23 maggio e il 24 maggio 2024 dai sensori SIAP posti sulla piattaforma Acqua Alta. I dati vengono richiesti all'API del comune di Venezia.

L'idea dell'import dei dati delle stazioni osservative fisse sarebbe quella di ottenere i dati, elaborarli e salvarli nel database PostgreSQL. Il problema è che, per quanto visto finora, non c'è coerenza tra i dati restituiti dalle API. Inoltre non si sa ancora il formato delle variabili mancanti. Quindi, al momento, viene difficile pensare a una vera e propria strategia per trattare questi dati in modo da uniformarli il più possibile. Ed è il motivo per cui le due API analizzate, al momento, restituiscono semplicemente il file json ricevuto senza un'ulteriore elaborazione.

3.3 Costruzione API

Il concetto di **API** (*Application Programming Interface*) risulta essere abbastanza astratto. Il termine è frequentemente utilizzato nelle interazioni tra gli sviluppatori ma, quando viene posta la domanda "cosa significa?" nessuno sa mai rispondere⁴⁹. Ora si tenterà di dare una definizione precisa e che chiarisca la nozione in modo che chiunque, non necessariamente programmatore, possa comprenderla.

Un'API è un insieme di protocolli⁵⁰ che permette a diverse componenti software di comunicare e trasferire dati tra loro. In poche parole si pone come livello intermedio tra due elementi. Questi due elementi vengono chiamati *client* e *server*. Il client, facendo una determinata azione (che potrebbe essere il click su un pulsante) invia una richiesta al server per accedere a una risorsa. La richiesta dipende dal tipo di API implementata, ma in generale contiene sempre le seguenti componenti:

- **endpoint**: un URL⁵¹ – *Uniform Resource Locator* – che fornisce l'accesso alla risorsa, come mostrato in Figura 3.9.

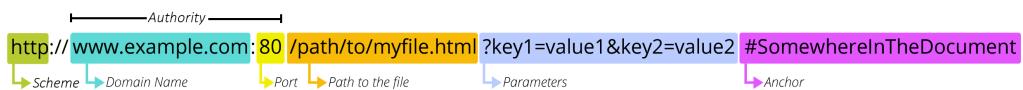


Figura 3.9: Esempio di schema standard di un URL^a.

^aVi, https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL/mdn-url-all.pdf.

- *metodo*: tipo di operazione da eseguire sulla risorsa, come aggiornamento, eliminazione, creazione dei dati.
- *parametri*: istruzioni specifiche.

⁴⁹Fonte: esperienza personale dell'autrice.

⁵⁰Con il termine *protocollo* si intende un insieme di regole.

⁵¹Mozilla Corporation. *Resources for Developers, by Developers_ (Online)*, https://developer.mozilla.org/en-US/docs/Learn/Common_questions/Web_mechanics/What_is_a_URL.

- *header richiesta*: intestazione nella forma di coppie chiave-valore che contengono informazioni aggiuntive, usata spesso nell'API di autenticazione che necessitano di credenziali per l'accesso.
- *corpo della richiesta*: i dati che servono per creare, aggiornare o eliminare una risorsa.

Il server a questo punto invia una risposta al client, che tipicamente contiene:

- *codice di stato*: codici a tre cifre che indicano l'esito della richiesta, i più comuni sono *200 OK* che esprime l'esito positivo e *404 Non trovato* quando il server non trova i dati richiesti.
- *header di risposta*: simili alle intestazioni di richiesta, solo che forniscono informazioni aggiuntive sulla risposta del server.
- *corpo della risposta*: dati e contenuti richiesti dal client, oppure un messaggio di errore se qualcosa è andato storto.

Come esempio sarà usata l'API di *Google Libri*⁵². In questo caso il client è il browser utilizzato dall'utente e il server è Google. Ora, dato come *parametro* il codice isbn di un libro, il client invia una richiesta al server per ottenere tutte le informazioni di quel determinato libro. La richiesta ha la seguente forma:

```
1 http GET https://www.googleapis.com/books/v1/volumes?q
=9788835711438
```

Listing 3.16: richiesta del client

dove *GET* è il metodo utilizzato, *https://www.googleapis.com/books/v1/volumes* l'URL dell'endpoint e *9788835711438* il parametro della richiesta. Il server poi risponde inviando i dati del libro in formato convenzionale *json* e insieme ad esso il codice *200 OK* per segnalare che la richiesta è andata a buon fine.

In questo esempio la presenza di header non è rilevante, né da parte del server né da parte del client. In altri contesti sono invece fondamentali.

```
1 HTTP/1.1 200 /* codice di stato */
2 {
3     "items": [
4         {
5             "etag": "0vjmWCIXa3g",
6             "id": "bIgzEAAAQBAJ",
7             "kind": "books#volume",
8             "selfLink": "https://www.googleapis.com/books/v1/volumes/
bIgzEAAAQBAJ",
```

⁵²Google for Developers. *Google Books APIs (Online)*, <https://developers.google.com/books>.

```

9      "authors": [
10         "TJ Klune"
11     ],
12     "description": "Linus Baker e' un assistente sociale
13     impiegato al Dipartimento della Magia Minorile. [...] su cosa
14     significhi accorgersi che, a volte, si puo' scegliere la vita
15     che si vuole. E, se si e' abbastanza fortunati, magari quella
16     vita ci sceglie a sua volta.",
17     "imageLinks": {...},
18     "industryIdentifiers": [
19       {
20         "identifier": "9788835711438",
21         "type": "ISBN_13"
22       },
23       {
24         "identifier": "8835711436",
25         "type": "ISBN_10"
26       }
27     ],
28     "infoLink": "https://play.google.com/store/books/details?
29 id=bIgzEAAAQBAJ&source=gbs_api",
30     "language": "it",
31     "publishedDate": "2021-07-13",
32     "title": "La casa sul mare celeste"
33     /* altre informazioni sul libro */
34   }
35 }
```

Listing 3.17: risposta del server

Le API sono utilizzate perché offrono diversi vantaggi. Automatizzano i lavori ripetitivi, migliorando la produttività degli sviluppatori e accelerando lo sviluppo del software. Forniscono un livello di sicurezza richiedendo autenticazione e autorizzazioni nelle richieste di accesso a dati sensibili. Infine, sono utili per l'accesso a infrastrutture di terze parti quando diverse aziende collaborano⁵³.

A questo punto dovrebbe essere chiaro il significato di API e si può entrare nei dettagli di quelle realizzate per *Ismar Data*.

Autenticazione.

L'autenticazione è il meccanismo che associa una richiesta in entrata a un insieme di credenziali identificative, come l'utente da cui proviene la richiesta o il token con cui è stata firmata. Le policy di autorizzazione e

⁵³Postman, Inc. *What is an API? (Online)*, <https://www.postman.com/what-is-an-api/>.

limitazione possono quindi utilizzare tali credenziali per determinare se la richiesta deve essere consentita⁵⁴.

L'autenticazione utilizzata segue lo schema http⁵⁵ dell'autenticazione tramite token. Il token viene dato al client da un server autorizzato. Il client utilizzerà il token ricevuto per accedere alle risorse protette ospitate dal server.

La costruzione delle API di autenticazione è fatta utilizzando Django Rest Framework⁵⁶, semplicemente perché si integra bene con il progetto stesso.

Innanzitutto occorre aggiungere il framework alle app installate nel progetto e specificare lo schema di autenticazione che si vuole utilizzare.

```

1 INSTALLED_APPS = [
2     ...
3     'rest_framework.authtoken',
4 ]
5 ...
6 ...
7
8 REST_FRAMEWORK = {
9     "DEFAULT_AUTHENTICATION_CLASSES": [
10         "ismarbo.authentication.BearerAuthentication",
11     ],
12 }
```

Listing 3.18: settings.py

In questo caso, come accennato nel paragrafo di introduzione, si andrà ad utilizzare l'autenticazione tramite token. Spesso viene indicata anche come autenticazione bearer.

Un token di sicurezza con la proprietà che qualsiasi soggetto in possesso del token (un "bearer") può utilizzare il token in qualsiasi modo in cui qualsiasi altro soggetto in possesso di esso può farlo. L'uso di un bearer token non richiede che il portatore dimostri il possesso del materiale della chiave crittografica (prova di possesso)⁵⁷.

Successivamente si andrà ad eseguire un comando da terminale per aggiornare (o creare, in caso non esistesse già) il database. In questo modo sarà creata la

⁵⁴Django REST Framework. *Django REST Framework (Online)*, <https://www.django-rest-framework.org/>.

⁵⁵È possibile che nell'evoluzione dello sviluppo si presenti la necessità di utilizzare altri schemi, al momento si è deciso di proseguire seguendo un percorso semplice.

⁵⁶Django REST Framework, cit., <https://www.djangoproject.com/>.

⁵⁷M. B. Jones e D. Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750, p. 3.

nuova tabella chiamata "*authtoken_token*" che, come dice il nome, conterrà i token e i riferimenti agli utenti in possesso di esso.

```
1 python manage.py migrate
```

Infine è necessario garantire che ogni utente abbia il suo token. Il problema in questione può avere tre soluzioni distinte. La prima, quella utilizzata, si basa sulla cattura di segnali generati dall'utente che generano un'assegnazione automatica del token. La seconda riguarda la creazione di un endpoint API nel quale l'utente, dato username e password, riceve il proprio token. L'ultima metodologia, poco flessibile, consiste nella creazione dei token nell'interfaccia admin di django.

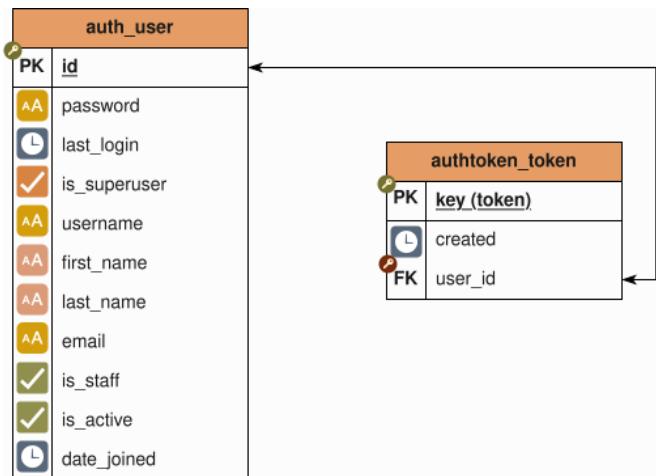
```
1 @receiver(post_save, sender=settings.AUTH_USER_MODEL)
2 def create_auth_token(sender, instance=None, created=False, **kwargs):
3     """
4         A function to make sure that whenever a user is created
5             its token will be generated
6     """
7     if created:
8         Token.objects.create(user=instance)
```

Listing 3.19: models.py

Lo snippet di codice sopra è collocato nel file "*models.py*" in modo che sia importato automaticamente da django all'avvio. Il segnale "*post_save*" viene generato alla fine dell'esecuzione del metodo "*save()*" che si occupa di salvare i dati dell'utente all'interno del database. Questo provoca l'esecuzione del codice della funzione che genera il token e lo assegna all'utente.

Per verificare l'effettivo funzionamento dell'autenticazione occorre creare un'API sulla quale effettuare i test.

```
1 class ListUsers(APIView):
2     """
3         View to list all users in the system.
4
5         * Requires bearer authentication.
6
7         Test with httpie
8         http GET 127.0.0.1:8000/users/ 'Authorization: bearer <
9             insert token here>'
10
11         authentication_classes = [authentication.
12             BearerAuthentication]
13
14         def get(self, request, format=None):
15             """
```



Porzione di database rappresentate le due tavole *auth_user* e *authtoken_token* e i relativi attributi. Esse sono collegate tramite l'id dell'utente.

Figura 3.10: Porzione database PostgreSQL. (Schema creato utilizzando drawio^a).

^aJGraph Ltd. *Flowchart Maker & Online Diagram Software - Draw.io (Online)*, <https://www.drawio.com/>.

```

14     Return a list of all users.
15     """
16
17     usernames = [user.username for user in User.objects.all()]
18     return Response(usernames)

```

Listing 3.20: api.py

L'API in Listing 3.20 si occupa di stampare a video la lista degli utenti presenti nel database. La richiesta ha successo (vedi Listing 3.21) se il token inserito nell'header della richiesta è valido, altrimenti fallisce con errore (vedi Listing 3.22).

```

1   $ http GET 127.0.0.1:8000/users/ 'Authorization: bearer
2   d9a658c85c67054c6c3626848ffac953fe3a9c6f'
3   HTTP/1.1 200 OK
4   Allow: GET, HEAD, OPTIONS
5   Content-Length: 27
6   Content-Type: application/json
7   Cross-Origin-Opener-Policy: same-origin
8   Date: Fri, 05 Apr 2024 15:16:17 GMT
9   Referrer-Policy: same-origin
10  Server: WSGIServer/0.2 CPython/3.11.8
11  Vary: Accept
12  X-Content-Type-Options: nosniff

```

```

12 X-Frame-Options: DENY
13
14 [
15     "system",
16     "cogotti-giulia"
17 ]

```

Listing 3.21: HTTP/1.1 200 OK

```

1 $ http GET 127.0.0.1:8000/users/ 'Authorization: bearer 943
2 b12684a39527e57cb0b4e70699883ca36e63c'
3 HTTP/1.1 401 Unauthorized
4 Allow: GET, HEAD, OPTIONS
5 Content-Length: 27
6 Content-Type: application/json
7 Cross-Origin-Opener-Policy: same-origin
8 Date: Fri, 05 Apr 2024 15:18:16 GMT
9 Referrer-Policy: same-origin
10 Server: WSGIServer/0.2 CPython/3.11.8
11 Vary: Accept
12 WWW-Authenticate: Bearer
13 X-Content-Type-Options: nosniff
14 X-Frame-Options: DENY
15 {
16     "detail": "Invalid token."
17 }

```

Listing 3.22: HTTP/1.1 401 Unauthorized

L'autenticazione tramite token è facile da implementare ma presenta alcune problematiche da considerare. Innanzitutto chiunque possiede la stringa del token può accedere alla risorsa se non si implementano ulteriori livelli di protezione. Infatti ne viene consigliato l'utilizzo solo tramite protocollo *https*⁵⁸. Il token può contenere informazioni sensibili. Un utente malintenzionato potrebbe modificare alcune informazioni del token – *un esempio può essere il periodo di validità* – ottenendo accessi che non dovrebbe avere. Infine, può accadere che l'utente malintenzionato utilizzi un token valido in passato per accedere a risorse nel presente⁵⁹.

L'evoluzione nello sviluppo dell'applicazione potrebbe portare alla necessità di cercare metodi migliori, potenziando l'efficienza e la sicurezza. Al momento, data l'incertezza presente, l'autenticazione bearer rimane l'implementazione scelta ed utilizzata realmente.

⁵⁸Django REST Framework, cit., <https://www.django-rest-framework.org/>.

⁵⁹Jones e Hardt, cit., pp. 10–11.

API front-end. Le API *front-end* si occupano di passare i dati necessari a visualizzare le informazioni nell'applicazione. Esse sono implementate in stile GraphQL tramite la libreria python.

L'applicazione deve visualizzare la lista delle stazioni osservative fisse, sia come forma di pallini sulla mappa che come lista tradizionale. Una singola API è in grado di fornire questi dati, sarà poi il front-end ad occuparsi della chiamata e dell'inserimento nella giusta interfaccia.

I dati delle stazioni sono statici e dunque risiedono, insieme ai radar e ai satelliti, nell'entità "*ismarbo_location*" (vedi porzione di schema in Figura 3.11) all'interno del database PostgreSQL. D'ora in poi si userà il termine *location* per indicare gli elementi della tabella in questione.

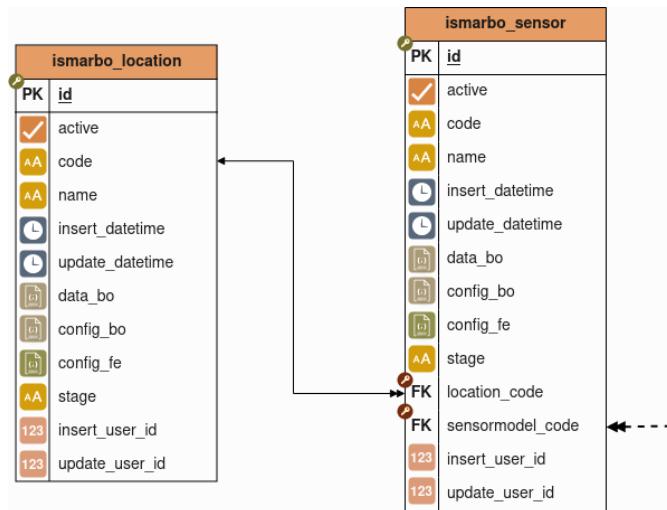


Figura 3.11: Porzione database. (Schema creato utilizzando drawio^a).

Tabelle del database rappresentanti le *location* (*ismarbo_location*) e i sensori (*ismarbo_sensor*). Relazione rappresentata dalla linea che collega i campi *code* e *location_code*. La freccia singola simboleggia il fatto che un sensore può appartenere ad un'unica *location*, mentre la freccia doppia che una *location* può avere più di un sensore.

^aJGraph Ltd. *Flowchart Maker & Online Diagram Software - Draw.io (Online)*, <https://www.draw.io/>.

Gli attributi di ogni location sono:

- *code*: codice identificativo univoco (per es. ISMAR_AAOT).
- *id*: identificatore numerico auto incrementato all'aggiunta di nuovi dati, inserito perché il team ha deciso che è sempre bene averlo.
- *name*: nome della location (per es. Piattaforma Oceanografica Acqua Alta).

- *active*: indica se la location funziona oppure no. Fondamentale in caso di guasto non risolvibile in poco tempo, dato che le location sono poste principalmente sul mare (piattaforme, boe e radar) o addirittura nello spazio (satelliti). Grazie al campo *active* l'applicazione continua a funzionare.
- *stage*: indica se la stazione deve essere considerata nella versione in produzione dell'applicazione, al momento sono tutte in "test".
- *config_fe*: le informazioni della location (per es. latitudine, longitudine, tipo, etc.) in formato json.
- *insert_user_id* e *update_user_id*: riferimento all'id dell'utente che ha inserito o aggiornato i valori della *location*.
- *insert_datetime* e *update_datetime*: data e ora di inserimento e aggiornamento della location.
- *data_bo* e *config_bo*: eventuali dati e configurazioni interne del backoffice.

Ogni location, per misurare realmente le variabili climatiche, utilizza dei sensori. Essi risiedono nella tabella "*ismarbo_sensor*". I campi all'interno sono simili a quelli presenti nella location, ad eccezione delle chiavi esterne. La chiave esterna `location_code` permette di collegare i sensori alla propria location, mentre `sensor_model_code` collega il sensore alla tabella contenente informazioni sul modello del sensore.

Python, grazie alle funzionalità offerte da Django, ha dei meccanismi per rispecchiare lo schema del database in modelli, salvati nel file "*models.py*", con una classe per ogni entità. Il file "*schema.py*" invece descrive il modello di dati e fornisce un server GraphQL con un set associato di metodi di *resolver* che sanno come recuperare i dati⁶⁰. Tutte le API sono state create utilizzando i metodi di resolver. Un resolver ha la forma `def resolve_foo(parent, info, **kwargs)` dove `foo` è il nome del campo dichiarato nell'oggetto `Query`.

L'API per la lista delle stazioni osservative fisse (vedi Listing 3.23) segue lo schema base evidenziato sopra. Al momento l'unico filtro utilizzato è quello che verifica se la stazione è attiva o no quando non si è in fase di sviluppo (`settings.debug` false).

```

1 # lista delle stazioni osservative fisse
2 list_location = graphene.List(IsmarboLocationType)
3
4 def resolve_list_location(root, info, **kwargs):
5     query = IsmarboLocation.objects.all()
6     if settings.DEBUG is False:

```

⁶⁰Arminio et al., cit., <https://docs.graphene-python.org/projects/django/en/latest/queries/>.

```
7     query = query.filter(active=True)
8 return query
```

Listing 3.23: API lista stazioni osservative fisse

Le altre API create sono molto simili, semplicemente utilizzano modelli differenti e restituiscono, sempre tramite resolver, i dati per cui sono state costruite. Lo sviluppo del front-end non ha ancora avuto inizio, quindi il team di back-end ha deciso di creare API generali che restituiscano tutti i dati senza filtri. Appena avrà inizio si effettuerà un incontro tra i due team per decidere la strada migliore da percorrere, quindi quale lato applicherà i filtri e quali dati sono realmente necessari.

Un esempio di utilizzo di queste API è presente in Figura 3.12. Le immagini mostrano i risultati delle richieste. In particolare nella prima immagine è presente il risultato dell'API che restituisce la lista delle stazioni osservative fisse. Le altre due immagini rappresentano i widget, ovvero le variabili climatiche osservate, delle stazioni Piattaforma Oceanografica Acqua Alta e Paloma, ottenuti selezionandole dalla lista. Si può notare come i dati non siano disponibili, ad eccezione di un grafico creato con dati appositi per testare il funzionamento dell'API. Questo conferma la lentezza dello sviluppo e l'impossibilità di raggiungere un traguardo fino a quando il problema dei dati non sarà risolto.

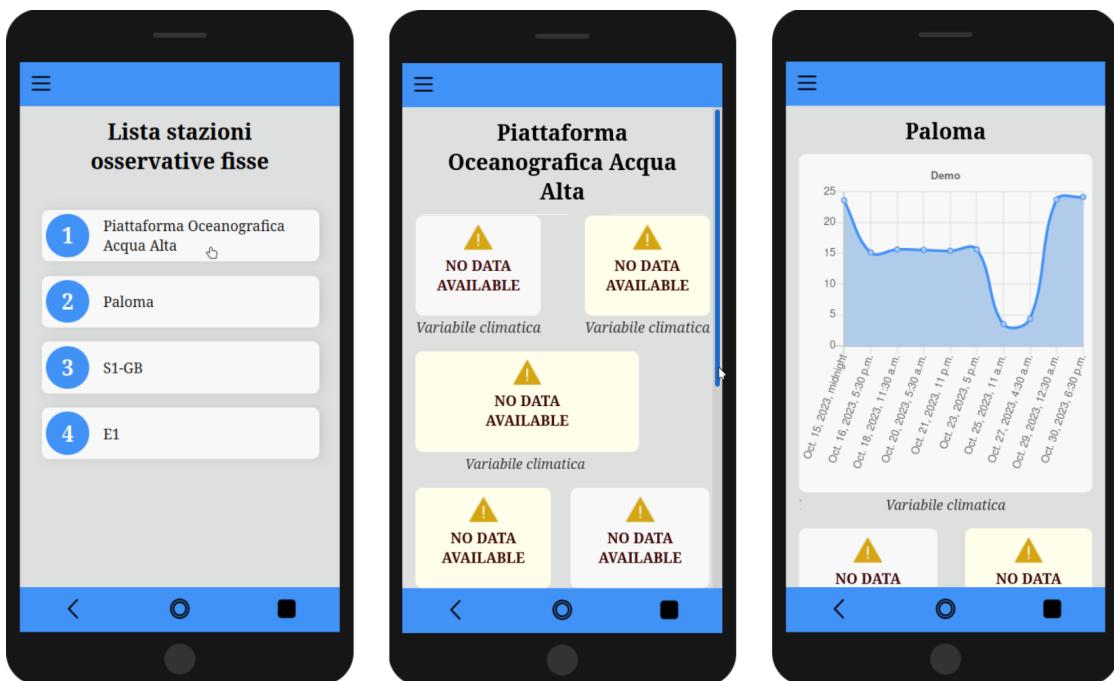


Figura 3.12: Schermate della demo web che utilizzano le API appena descritte per visualizzare la lista delle stazioni osservative fisse e i widget con i dati, non disponibili, delle variabili climatiche associate ad ognuna di esse. L'unico grafico è stato realizzato tramite dati di test^a.

^aFonte: screenshot dell'autrice.

Capitolo 4

Conclusioni

In questa tesi si è esaminato lo sviluppo di *Ismar Data*, applicazione meteorologica creata da Elan42 e commissionata da ISMAR-CNR. Sono state analizzate le funzionalità richieste dal cliente e l'architettura ideale per realizzarle. Ampio spazio è stato dato all'analisi delle sorgenti di dati, pubbliche e private, con i relativi problemi di accesso ai rispettivi dati. Infine si è dato rilievo alla produzione vera e propria del prodotto, illustrando gli strumenti utilizzati e la fase di importazione dei dati. Ora sorge spontaneo porsi la seguente domanda: alla fine di questo percorso, *sono stati raggiunti gli obiettivi?* Una risposta negativa è esattamente quella che ci si aspetta dopo aver letto il testo, e infatti è andata così. Sicuramente l'applicazione tratta un tema, quello meteorologico, poco familiare all'azienda produttrice. Questo fattore ha alzato inizialmente il livello di difficoltà, ma, dopo qualche settimana di studio il grado è diminuito drasticamente. Quindi, *cosa non ha funzionato?* Ricordando il problema della mancanza dei dati, ampiamente discusso in questa tesi, dovuto al fatto che non sono stati forniti gli accessi alle sorgenti private, si ha immediatamente una risposta chiara: non ha funzionato la comunicazione e collaborazione tra i vari enti proprietari. Infine, *come si sarebbe potuto risolvere il problema?* Dato che i tentativi di accordarsi non hanno funzionato, l'unica soluzione sarebbe stata prevenire il problema. ISMAR-CNR avrebbe dovuto assicurarsi, prima di incaricare Elan42 della produzione, che ci fossero tutti i dati a disposizione per poter procedere nello sviluppo in modo da rispettare le scadenze, riducendo le richieste in caso qualche ente privato non avesse voluto collaborare. In questo modo si sarebbe evitato di trascinare il progetto per un tempo tendenzialmente infinito. La speranza è che questa esperienza insegni qualcosa ai partecipanti al progetto e che queste vicende rimangano solamente brevi episodi nella vita di una stagista.

Bibliografia

- Agafonkin, V. *Leaflet: an open-source JavaScript library for mobile-friendly interactive maps (Online)*. Ultima consultazione: 4 maggio 2024. URL: <https://leafletjs.com/index.html>.
- Arminio, P., S. Akbary, D. Fee, J. Foster et al. *Graphene. GraphQL in Python Made Easy (Online)*. Ultima consultazione: 22 aprile 2024. URL: <https://graphene-python.org/>.
- Asana, Inc. *Asana: un modo più intelligente di lavorare (Online)*. Ultima consultazione: 10 aprile 2024. URL: <https://asana.com/it>.
- Atlassian. *Ecmwf confluence (Online)*. Ultima consultazione: 6 maggio 2024. URL: <https://confluence.ecmwf.int/>.
- Atzeni, P., S. Ceri, P. Fraternali, S. Paraboshi et al. *Basi di dati*. 4^a ed. McGraw-Hill Education, 2014. ISBN: 978-8838665875.
- Beck, K., M. Beedle, A. Van Bennekum, A. Cockburn et al. *Manifesto for agile software development*. Snowbird, UT, 2001. URL: <https://agilemanifesto.org/iso/it/manifesto.html>.
- Bertotti, L., P. Canestrelli, L. Cavaleri, F. Pastore et al. «The Henetus wave forecast system in the Adriatic Sea». In: *Natural Hazards and Earth System Science* 11 (2011), pp. 2965–2979. DOI: 10.5194/nhess-11-2965-2011.
- Bertotti, L., L. Cavaleri, L. Loffredo e L. Torrisi. «Nettuno: Analysis of a Wind and Wave Forecast System for the Mediterranean Sea». In: *Monthly Weather Review* 141 (2013), pp. 3130–3141. DOI: 10.1175/MWR-D-12-00361.1.
- Blower, J. D., K. Haines, A. Santokhee e C. L. Liu. «GODIVA2: interactive visualization of environmental data on the Web». In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1890 (2009), pp. 1035–1039.
- Buongiorno Nardelli, B., A. Pisano, C. Tronconi e R. Santoleri. «Evaluation of different covariance models for the operational interpolation of high resolution satellite Sea Surface Temperature data over the Mediterranean Sea». In: *Remote Sensing of Environment* 164 (2015), pp. 334–343. ISSN: 0034-4257. DOI:

- <https://doi.org/10.1016/j.rse.2015.04.025>. URL: <https://www.sciencedirect.com/science/article/pii/S0034425715001674>.
- CARTO. *CARTO basemaps (Online)*. Ultima consultazione: 5 maggio 2024. URL: <https://carto.com/basemaps>.
- Chacon, S. e B. Straub. *Pro Git: everything you need to know about Git*. Springer Nature, 2014. ISBN: 978-1484200773. URL: <https://library.oapen.org/bitstream/handle/20.500.12657/28155/1/1001839.pdf>
- Conrad, E. «Chapter 8 - Domain 8: Application Development Security». *Eleventh Hour CISSP*. A cura di E. Conrad. Boston: Syngress, 2011, pp. 129–145. ISBN: 978-1-59749-566-0. DOI: <https://doi.org/10.1016/B978-1-59749-566-0.00008-4>. URL: <https://www.sciencedirect.com/science/article/pii/B9781597495660000084>.
- Copernicus Marine Service e Institute of Marine Sciences. *Mediterranean SST Analysis, L4, 1km daily (Dataset)*. Ultima consultazione: 9 aprile 2024. URL: <http://ritmare.artov.ismar.it/thredds/ritmare/SatelliteOS/SST/catalog.html?dataset=SSTUHR>.
- Cognati, L., C. Mantovani, A. Novellino, A. Rubio et al. «Recommendation Report 2 on improved common procedures for HFR QC analysis. JERICO-NEXT WP5-Data Management». JERICO-NEXT-WP5-D5.14-V2.0 (2024). DOI: <https://doi.org/10.25607/10.25607/0BP-944.2>. URL: <https://repository.oceanbestpractices.org/handle/11329/1441.2>.
- Deser, C., M. Alexander, S.-P. Xie e A. Phillips. «Sea Surface Temperature Variability: Patterns and Mechanisms». In: *Annual review of marine science* 2 (2010), pp. 115–43. DOI: [10.1146/annurev-marine-120408-151453](https://doi.org/10.1146/annurev-marine-120408-151453).
- Dierssen, H. M. «Perspectives on empirical approaches for ocean color remote sensing of chlorophyll in a changing climate». In: *Proceedings of the National Academy of Sciences* 107.40 (2010), pp. 17073–17078.
- Diffingo Solutions Inc. *What is Open Source Software (Online)*. Ultima consultazione: 20 aprile 2024. 2008. URL: <https://web.archive.org/web/20081028104313/http://www.diffingo.com/oss/whyoss>.
- Django REST Framework. *Django REST Framework (Online)*. Ultima consultazione: 4 aprile 2024. URL: <https://www.django-rest-framework.org/>.
- Django Software Foundation and individual contributors. *The web framework for perfectionists with deadlines (Online)*. Ultima consultazione: 23 marzo 2024. URL: <https://www.djangoproject.com/>.
- Doretti, R. *Data base. Concetti e disegno*. Gruppo Editoriale Jackson, 1985. ISBN: 88-7056-174-7. URL: <https://archive.org/details/databseconcettiedisegno/>.

- E42 srl. *Sito web Agenzia Elan42 (Online)*. Ultima consultazione: 27 marzo 2024.
URL: <https://www.elan42.com/>.
- European Centre for Medium-Range Weather Forecasts. *Parameter Database (Online)*. Ultima consultazione: 28 maggio 2024. URL: <https://codes.ecmwf.int/grib/param-db/?encoding=grib1&table=140>.
- GDAL/OGR contributors. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. 2024. DOI: 10.5281/zenodo.5884351.
URL: <https://gdal.org>.
- GitHub, Inc. *100 million developers and counting (Online)*. Ultima consultazione: 19 aprile 2024. 2023. URL: <https://github.blog/2023-01-25-100-million-developers-and-counting/>.
- GitLab B.V. *GitLab (Online)*. Ultima consultazione: 19 aprile 2024. URL: <https://about.gitlab.com/why-gitlab/>.
- Google. *Flutter - Build for any screen (Online)*. Ultima consultazione: 23 marzo 2024.
URL: <https://flutter.dev/>.
- *Google meet: video chiamate e riunioni per tutti (Online)*. Ultima consultazione: 17 aprile 2024. URL: <https://meet.google.com/>.
- Google for Developers. *Google Books APIs (Online)*. Ultima consultazione: 1 maggio 2024. URL: <https://developers.google.com/books>.
- GraphQL Contributors. *GraphQL & the GraphQL LSP Reference Ecosystem for building browser & IDE tools (Online)*. Ultima consultazione: 22 aprile 2024. URL: <https://github.com/graphql/graphiql?tab=readme-ov-file#graphiql>.
- Holovaty, A. e J. Kaplan-Moss. *The Django Book*. Ultima consultazione: 21 aprile 2024. 2013. URL: <https://readthedocs.org/projects/djangobook/downloads/pdf/latest/>.
- Howe, D. *Transport Layer Security protocol (Online)*. Ultima consultazione: 29 maggio 2024. URL: https://foldoc.org/Transport_Layer_Security_protocol.
- ISMAR-CNR. *dataset-oc-med-chl-multi-l3-chl_1km_daily-rt-v02 (Dataset)*. Ultima consultazione: 4 aprile 2024. URL: http://ritmare.artov.ismar.cnr.it/thredds/ritmare/SatelliteOS/OC/catalog.html?dataset=CHL_CASE12_A.
- *dataset-oc-med-chl-multi-l3-chl_1km_daily-rt-v02 (Immagine)*. URL: <http://ritmare.artov.ismar.cnr.it/thredds/godiva2/godiva2.html?menu=&layer=CHL&elevation=0&time=2023-05-22T00:00:00.000Z&scale=0.01,10&bbox=-238.354357,-338.673427,481.645643,223.826573&server=http://ritmare.artov.ismar.cnr.it/thredds/wms/xchlcase12>.
- *HFR-TirLig_catalog (Dataset)*. Ultima consultazione: 4 aprile 2024. URL: https://thredds.hfrnode.eu:8443/thredds/NRTcurrent/HFR-TirLig/HFR-TirLig_catalog.html.

- ISMAR-CNR. *Istituto di scienze marine (Online)*. Ultima consultazione: 27 marzo 2024. URL: <https://www.ismar.cnr.it/>.
- *Mediterranean SST Analysis, L4, 1km daily (Immagine)*. URL: http://ritmare.artov.ismar.cnr.it/thredds/godiva2/godiva2.html?menu=&layer=analysed_sst&elevation=0&time=2023-05-22T00:00:00.000Z&scale=289.2,294&bbox=66.889391,84.287338,78.139391,93.0764&server=http://ritmare.artov.ismar.cnr.it/thredds/wms/sstuhr.
 - *Near Real Time Surface Ocean Velocity by HFR-TirLig network*. Ultima consultazione: 4 aprile 2024. DOI: <https://doi.org/10.57762/35S0-EE87>. URL: <https://www.hfrnode.eu/networks/hfr-tirlig/>.
 - *Rete radar HF (Immagine)*. 2023. URL: <https://www.ismar.cnr.it/wp-content/uploads/2023/06/mappa-Radar-HF.jpg>.
- Jelvix. *Waterfall vs. Agile (Immagine)*. URL: <https://jelvix.com/wp-content/uploads/2020/07/waterfall-vs-agile.jpg>.
- JGraph Ltd. *Flowchart Maker & Online Diagram Software - Draw.io (Online)*. Ultima consultazione: 22 aprile 2024. URL: <https://www.drawio.com/>.
- Jones, M. B. e D. Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. RFC 6750. Ott. 2012. DOI: [10.17487/RFC6750](https://doi.org/10.17487/RFC6750). URL: <https://www.rfc-editor.org/info/rfc6750>.
- Maintenance Team. *Arch Wiki (Online)*. Ultima consultazione: 6 maggio 2024. URL: <https://wiki.archlinux.org/>.
- Mbuguah, S., V. Mony e G. Nyabuto. «Architectural Review of Client-Server Models». In: *International Journal of Scientific Research and Engineering Trends* 10 (2024), pp. 139–143.
- Merder, J., G. Zhao, N. Pahlevan, R. A. Rigby et al. «A novel algorithm for ocean chlorophyll-a concentration using MODIS Aqua data». In: *ISPRS Journal of Photogrammetry and Remote Sensing* 210 (2024), pp. 198–211. ISSN: 0924-2716. DOI: <https://doi.org/10.1016/j.isprsjprs.2024.03.014>. URL: <https://www.sciencedirect.com/science/article/pii/S0924271624000868>.
- Microsoft. *Visual Studio Code. Code editing. Redefined. (Online)*. Ultima consultazione: 19 aprile 2024. URL: <https://code.visualstudio.com/>.
- Miller, R. B. «Response time in man-computer conversational transactions». *American Federation of Information Processing Societies: Proceedings of the AFIPS '68 Fall Joint Computer Conference, December 9-11, 1968, San Francisco, California, USA - Part I*. Vol. 33. AFIPS Conference Proceedings. AFIPS / ACM / Thomson Book Company, Washington D.C., 1968, pp. 267–277. DOI: [10.1145/1476589.1476628](https://doi.acm.org/10.1145/1476589.1476628). URL: <http://doi.acm.org/10.1145/1476589.1476628>.

- Ministero del Lavoro e delle Politiche Sociali. *Smart working (Online)*. Ultima consultazione: 17 aprile 2024. 2023. URL: <https://www.lavoro.gov.it/strumenti-e-servizi/smart-working/Pagine/default>.
- Mozilla Corporation. *Resources for Developers, by Developers_ (Online)*. Ultima consultazione: 1 maggio 2024. URL: <https://developer.mozilla.org/en-US/>.
- Open Geospatial Consortium Inc. *OGC Network Common Data Form (NetCDF)*. A cura di B. Domenico. Ver. 1.0. 2011. URL: <http://www.opengis.net/doc/1S/netcdf/1.0>.
- *OpenGIS®Web Map Server Implementation Specification*. A cura di J. de la Beaujardiere. Ver. 1.3.0. 2006. URL: https://portal.ogc.org/files/?artifact_id=14416.
- Open Source Org. *The Open Source Definition (Online)*. Ultima consultazione: 20 aprile 2024. 2007. URL: <https://web.archive.org/web/20070611152544/https://opensource.org/docs/osd>.
- Oreopoulos, L., L. Boisvert e P. Przyborski. *About Aqua / Aqua Project Science (Online)*. Ultima consultazione: 1 aprile 2024. 2024. URL: <https://aqua.nasa.gov/content/about-aqua>.
- PostGIS PSC & OSGeo. *PostGIS (Online)*. Ultima consultazione: 1 maggio 2024. URL: <https://postgis.net/>.
- Postman. *allmeteoAPI (Online)*. Ultima consultazione: 29 maggio 2024. URL: <https://documenter.getpostman.com/view/11878734/TVYAf1Fn>.
- Postman, Inc. *What is an API? (Online)*. Ultima consultazione: 1 maggio 2024. URL: <https://www.postman.com/what-is-an-api/>.
- Python Software Foundation. Ultima consultazione: 20 aprile 2024. URL: <https://wiki.python.org/moin/BeginnersGuide/Overview>.
- *Python Package Index - PyPI (Online)*. Ultima consultazione: 20 aprile 2024. URL: <https://pypi.org/>.
- Ravaioli, M., C. Bergami, F. Riminucci, S. Aracri et al. *La rete scientifica italiana di siti fissi per l'osservazione del mare – IFON Stato dell'arte e upgrades durante il Progetto RITMARE (2012 – 2016)*. A cura di M. Ravaioli, C. Bergami e F. Riminucci. Ultima consultazione: 28 aprile 2024. Roma, CNR Pubblicazioni 2017, 2017. ISBN: 978-88-80802-44-0. URL: <https://www.ismar.cnr.it/wp-content/uploads/2022/07/rete-scientifica-italiana-siti-fissi-osservazione-mare-ifon-ismar-cnr.pdf>.
- Renner, T., R. E. Cohen, T. Cvitas, J. G. Frey et al. *Quantities, units and symbols in physical chemistry*. The Royal Society of Chemistry, 2007. ISBN: 978-0-85404-433-7. DOI: 10.1039/9781847557889. URL: <https://doi.org/10.1039/9781847557889>.

- Resinex Trading S.r.l. *ELASTIC BEACONS. The widest range of elastic beacons in the world*. Ultima consultazione: 23 maggio 2024. URL: <https://www.resinextrad.com/it/wp-content/uploads/2016/02/ElasticBeacons.pdf>.
- Rubio, A., J. Mader, L. Cognati, C. Mantovani et al. «HF Radar Activity in European Coastal Seas: Next Steps toward a Pan-European HF Radar Network». In: *Frontiers in Marine Science* 4 (2017). ISSN: 2296-7745. DOI: 10.3389/fmars.2017.00008. URL: <https://www.frontiersin.org/articles/10.3389/fmars.2017.00008>.
- Sazid Mohammad, S. e R. Ramakrishnan. «GeoTIFF - A standard image file format for GIS applications». In: *Map India - Image Processing & Interpretation* (2003). URL: <https://www.geospatialworld.net/wp-content/uploads/images/pdf/117.pdf>.
- Schwaber, K. e J. Sutherland. *La guida Scrum. La Guida Definitiva a Scrum: Le Regole del Gioco*. Ultima consultazione: 10 aprile 2024. 2020. URL: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Italian.pdf>.
- Slack Technologies, LLC. *Slack: la tua piattaforma di produttività (Online)*. Ultima consultazione: 17 aprile 2024. URL: <https://slack.com/intl/it-it/>.
- Stöckli, R. *Satellite Aqua della NASA (Immagine)*. 2009. URL: https://upload.wikimedia.org/wikipedia/commons/f/fb/The_Aqua_Satellite.jpg.
- The GraphQL Foundation. *GraphQL. A query language for your API (Online)*. Ultima consultazione: 22 aprile 2024. URL: <https://graphql.org/>.
- The PostgreSQL Global Development Group. *PostgreSQL: The World's Most Advanced Open Source Relational Database (Online)*. Ultima consultazione: 25 marzo 2024. URL: <https://www.postgresql.org/>.
- Tivix Inc. *Django-cron's documentation (Online)*. Ultima consultazione: 6 maggio 2024. URL: <https://django-cron.readthedocs.io/en/latest/>.
- Treccani S.P.A. *Clonare. In Vocabolario Treccani online*. Ultima consultazione: 19 aprile 2024. URL: <https://www.treccani.it/vocabolario/clonare/>.
- *Demo. In Vocabolario Treccani online*. Ultima consultazione: 9 aprile 2024. URL: https://www.treccani.it/vocabolario/demo_res-eceee31b-0017-11de-9d89-0016357eee51/.
 - *Eutrofizzazione. In Vocabolario Treccani online*. Ultima consultazione: 1 aprile 2024. URL: <https://www.treccani.it/vocabolario/eutrofizzazione/>.
 - *Fitoplàncton. In Vocabolario Treccani online*. Ultima consultazione: 1 aprile 2024. URL: <https://www.treccani.it/vocabolario/fitoplancton/>.
 - *Interpolazione. Dizionario delle Scienze Fisiche (1996)*. Ultima consultazione: 23 maggio 2024. URL: [https://www.treccani.it/enciclopedia/interpolazione_\(Dizionario-delle-Scienze-Fisiche\)/](https://www.treccani.it/enciclopedia/interpolazione_(Dizionario-delle-Scienze-Fisiche)/).

- Méda. In *Vocabolario Treccani online*. Ultima consultazione: 23 maggio 2024.
URL: <https://www.treccani.it/vocabolario/meda/>.
 - Plottare. In *Vocabolario Treccani online*. Ultima consultazione: 4 maggio 2024.
URL: <https://www.treccani.it/vocabolario/plottare/?search=plottare%2F>.
 - Repository. In *Enciclopedia Treccani online: lessico del XXI Secolo* (2013).
Ultima consultazione: 19 aprile 2024. URL: [https://www.treccani.it/encyclopedia/repository_\(Lessico-del-XXI-Secolo\)/](https://www.treccani.it/encyclopedia/repository_(Lessico-del-XXI-Secolo)/).
- Unidata/UCAR Community Programs (UCP). *NSF Unidata Software Documentation (Online)*. Ultima consultazione: 4 maggio 2024 (Online). URL: <https://docs.unidata.ucar.edu/index.html>.
- Volpe, G., S. Colella, V. E. Brando, V. Forneris et al. «Mediterranean ocean colour Level 3 operational multi-sensor processing». In: *Ocean Science* 15.1 (2019), pp. 127–146. DOI: 10.5194/os-15-127-2019. URL: <https://os.copernicus.org/articles/15/127/2019/>.
- Waters, K. *All about Agile: Agile Management Made Easy!* allaboutagile.com, 2012.
ISBN: 9781469915517. URL: <https://books.google.it/books?id=jsACuwAACAAJ>.

Ringraziamenti

*A chi non si è sentito all'altezza,
a chi non crede più in se stesso,
a me, che alla fine ce l'ho fatta.*

Al termine di questo lavoro, che completa il mio percorso universitario, vorrei ringraziare le persone che sono rimaste al mio fianco, sostenendomi ed incoraggiandomi, in questo periodo di vita.

Innanzitutto ci terrei a ringraziare la mia relatrice, la prof.ssa Alessandra Raffaetà, per aver accettato di accompagnarmi nella stesura di questa tesi. Ha avuto una pazienza immensa e mi ha fatta fare un buon lavoro, nonostante le circostanze. Ma soprattutto ha creduto in me!

Poi desidero ringraziare i miei genitori e i miei fratelli per avermi sopportata e supportata; un grazie speciale va a Fabio che, nonostante i 727 km di distanza, mi è rimasto accanto e mi ha sempre strappata un sorriso! Sono inoltre grata ai miei amici, vicini e lontani, Roberta C., Nicola D.A., Marika M., Marco C. e Diego P., per il sostegno nei momenti belli e anche in quelli un po' più complicati.

Infine, un grazie immenso va a Mirtolino, che mi ha aiutata tantissimo senza saperlo!