Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Triennale in Informatica

OTTIMIZZAZIONE SPAZIALE E OPERAZIONI DI AGGIORNAMENTO ALL'INTERNO DI UNA PIATTAFORMA LEGACY DI SERVIZI IOT

Relatore:
Dr.
Federico Montori

Presentata da:
Massimo Monacchi

Sessione III
Anno Accademico 2020/2021

Sommario

In un mondo sempre più mobile, social e multicanale cambiano le esigenze ed i comportamenti degli utenti, che hanno acquisito il potere d'influenzare il mercato condividendo sul Web la loro esperienza nell'utilizzo del prodotto proposto. La Consumer Experience (CX) è indubbiamente oggi un elemento cardine del business, dal quale dipende il successo dell'azienda attraverso l'aumento della soddisfazione del cliente, creando nuove opportunità di interazione e relazione con i brand.
Ciò richiede piattaforme e tecnologie digitali sempre più innovative, di facile utilizzo e in grado di garantire risultati tangibili in breve tempo che aumentano la soddisfazione e di conseguenza la fedeltà del cliente. L'Internet of Things (IoT) è una delle tecnologie chiavi volte al miglioramento della CX: permettono di raccogliere dati da dispositivi interconnessi tramite internet per risolvere esigenze specifiche. Per rendere più agevole la creazione di applicazioni, anche da parte di utenti non esperti nel campo informatico, è stato introdotto il paradigma dell'End-User Service Composition (EUSC).
Un'applicazione che coniuga IoT con EUSC è SenSquare: piattaforma utile per la creazione di servizi personalizzabili a partire da dati ambientali. Il contributo della tesi verte sul miglioramento della consumer experience nell'utilizzo dell'applicazione SenSquare. In particolare si è migliorata l'usabilità della piattaforma, rinnovando l'interfaccia grafica aggiornando il framework Angular all'ultima versione disponibile e sostituendo il servizio a pagamento Google Maps con mappe Open Source. Inoltre, per velocizzare il caricamento dei datastreams sulle mappe, si è implementato un nuovo algoritmo di caching, evitando di effettuare chiamate ricorsive al server.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Nell'ultimo ventennio è avvenuta una vera rivoluzione digitale, grazie all'introduzione di oggetti in ambienti lavorativi, pubblici e privati dotati di sensori interconnessi tramite internet; questo ecosistema è definito "Internet of Things" (o con l'acronimo IoT) in grado di raccogliere costantemente dati per la creazione di servizi e/o risolvere diverse esigenze.
Un buon esempio di applicazione di sistemi IoT sono le "Smart City"[8]. Grazie a telecamere, parcheggi intelligenti, stazioni di monitoraggio ambientale e molto altro è possibile ottimizzare, migliorare infrastrutture e servizi aiutando i cittadini nella loro quotidianità, con specifiche applicazioni. Il lavoro di tesi volge al miglioramento della consumer experience di SenSquare, aggiornando il framework Angular all'ultima versione disponibile e implementando un nuovo algoritmo di caching per velocizzare il caricamento dei datastreams sulle mappe.
Questa tesi si articola in sei capitoli. Nel Capitolo 1 è descritta la situazione attuale del settore delle Progressive Web App e IoT End-User Service Composition. Nel Capitolo 2 è descritta l'architettura della piattaforma SenSquare ove si sottolineando i problemi risolti da questo elaborato. Il Capitolo 3 descrive le scelte implementative dell'aggiornamento del framework Front-End della piattaforma SenSquare. Nel Capitolo 4 è illustrato il funzionamento dell'algoritmo di caching per caricare e visualizzare più velocemente i datastreams sulle mappe.
Il Capitolo 5 analizza i risultati ottenuti, confrontando i tempi di caricamento dei datasreams nel sito originale, con quelli ottenuti applicando il nuovo algoritmo di ottimizzazione. Infine, il Capitolo 6 chiude il documento analizzando i risultati del lavoro svolto e proponendo gli sviluppi futuri di arricchimento e miglioramento della piattaforma SenSquare.

Capitolo 1

Stato dell'arte

1.1 Progressive Web Apps

Nel 2015 la designer Frances Berriman e l'ingegnere Alex Russell, coniarono il termine Progressive Web Apps[11]. Per PWA si intendono tutte le applicazioni web che vengono caricate come pagine normali di un sito internet, ma che si comportano come delle applicazioni native quando si utilizzano su un dispositivo mobile. Il principale vantaggio è quello di occupare minore memoria sui devices e di essere mobile responsive (contenuto dinamico che si adatta allo schermo del dispositivo). Attraverso il Service Worker, le PWA possono eseguire operazioni in background. Questo modulo, inoltre, permette la memorizzazione dei dati necessari per rendere l'app offline. Angular rappresenta uno dei framework di riferimento per lo sviluppo di moderne Web App. La prima versione fu rilasciata da Google nell'ottobre del 2010 e fu chiamata AngularJS[1]. La sua caratteristica principale è quella di estendere le capacità del HTML, come il binding o le dependency injection, facilitando la scrittura del codice. Con il passare degli anni sono state pubblicate nuove versioni, fino ad Angular 12. I miglioramenti principali apportati sono qui sotto elencati.

Prestazioni

Lo scopo delle nuove versione di Angular è il continuo miglioramento di performance del framework, principalmente sotto questi due aspetti:

1. Dimensione del codice compilato
2. Caricamento dei moduli

Il primo aspetto è stato migliorato in Angular5. Il nuovo sistema di build riduce notevolmente il codice generato, grazie alle seguenti tre operazioni:

- Eliminazione dei decorators Angular in fase di compilazione perché non necessari nella fase di run-time
- Introduzione della operazione di tree shaking, eseguito dal compilatore, che elimina dalla build il codice non necessario in quanto mai invocato dall'applicazione
- Eliminazione degli spazi inutili, opzione attivabile dal programmatore, utile per grandi progetti

Il secondo aspetto riguarda la nuova modalità d'importazione dei moduli. In Angular 2, i moduli che venivano utilizzati all'interno del progetto erano definiti in un unico file. Ogni qual volta un client richiede una pagina web, il server ricerca all'interno del file dei moduli, che sia necessari al corretto funzionamento. Questa operazione di ricerca viene ripetuta ad ogni richiesta del browser, rallentando il caricamento della pagina web.
Al contrario, in Angular 12, ad ogni pagina web viene associato un file contenente i moduli necessari, eliminando così la ricerca degli stessi.

Nuovo linguaggio di programmazione di Angular

Si è passati da JavaScript a Typescript: superset di JavaScript che permette di scrivere codice più robusto.

Le caratteristiche principali di TypeScript sono la possibilità di poterlo usare per creare programmi client-side e server-side e anche permette di creare oggetti basati su classi.
JavaScript è considerato un linguaggio semplice (a volte definito di scripting) che non ha una tipizzazione statica, ma soltanto dinamica. Inoltre, TypeScript permette al programmatore di assegnare un tipo alle variabili e alle funzioni. Questa caratteristica rende il codice più facile da leggere e prevenire bug. Infine, un documento scritto in TypeScript, può importare file e moduli all'interno di esso, generando un codice pulito e di facile comprensione.

Metodo di progettazione

La prima versione di Angular è basata sul paradigma Model-View-Controller(MVC). Come si può intuire dal nome, questo pattern si basa su tre componenti logici:

- Model-responsabile della gestione dei dati dell'applicazione
- View-responsabile della visualizzazione dei dati all'utente
- Controller-definisce le funzionalità dell'applicazione, facendo da tramite tra Model e View

Il modello ha l'obiettivo di disaccoppiare l'interfaccia utente dal modello dei dati, in modo da ottenere un'architettura più flessibile favorendo lo sviluppo, il test e la manutenzione di ciascuna parte indipendentemente dall'altra. Quando l'utente interagisce con l'applicazione, il controller acquisisce la richiesta dell'utente e specifica al model quali dati processare per soddisfare la domanda. Il risultato, viene trasferito all'utente tramite la componente View (la Figura 1.1 mostra il funzionamento del modello MVC).
Al contrario, l'ultima versione di Angular adotta il pattern Model-View-ViewModel (MVVM), variante del MVC. Rispetto al modello precedente, la View ingloba alcune funzionalità del Controller, avendo così un ruolo più attivo nel gestire gli eventi e il data-binding. Il ViewModel, invece, prende la gestione del resto delle funzionalità del Controller, facendo da tramite

tra la View e il Model. Quando l'utente interagisce con l'interfaccia grafica dell'applicazione, gli input vengono inoltrati al ViewModel il quale effettua un cambio di stato accedendo, se necessario, al Model. Mediante l'utilizzo di Observer object, la View "osserva" il comportamento della ViewModel. Quando quest'ultima cambia di stato, automaticamente la View riflette il cambiamento mostrando i nuovi dati all'utente.

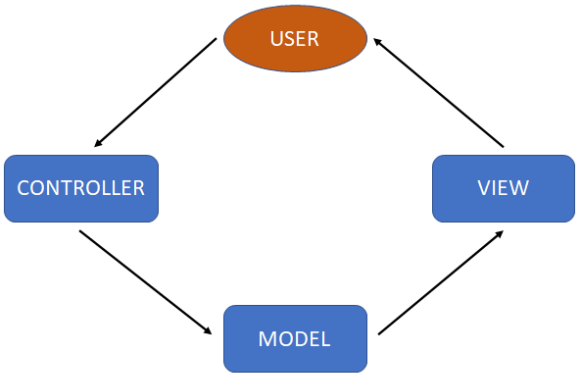

Figura 1.1: Funzionamento modello Model-View-Controller

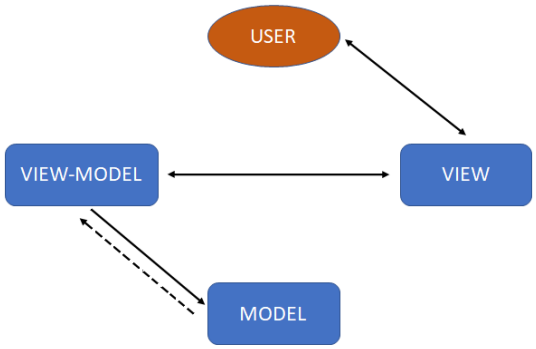

Figura 1.2: Funzionamento modello Model-ModelView-Controller

1.2 IoT End-User Service Composition

L'internet of Things ha cambiato il mondo in cui viviamo negli ultimi decenni: il numero di dispositivi connessi ha superato di gran lunga il numero di essere umani. Questa diffusione è dovuta anche agli innumerevoli sviluppi in questo ambito. I dati raccolti, però, sono enormi ed eterogenei, rendendo difficile il loro utilizzo. Da qui nasce la necessità di creare infrastrutture che facilitano l'iterazione tra gli utenti e i dati grezzi rilevati dalla rete di sensori: IoT End-User Services Composition (IoT-EUSC) è un ambiente di sviluppo che permette anche agli utenti meno esperti, di comporre autonomamente servizi personalizzabili e condividerli con la società. La caratteristica principale dell'IoT-EUSC è dovuta al fatto che gli utenti non solo interagiscono con il software, ma sono anche coinvolti nello sviluppo del prodotto stesso [12]. I programmatori, sulla base dei feedback degli utenti, modificano i programmi IoT-EUSC per renderli più intuitivi e di facile utilizzo.

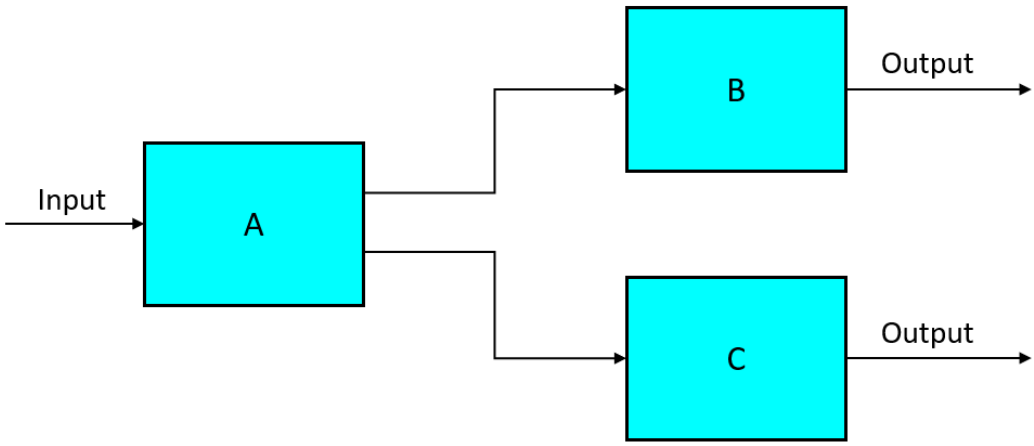

Figura 1.3: Esempio diagramma Flow-Based Programming

Un approccio IoT-EUSC, per la creazione di programmi, è il Flow-Based Programming (FBP)[9]. Questo paradigma stabilisce che un'applicazione sia

costituita da una serie di blocchi denominati black box che eseguono operazioni specifiche. Ogni blocco utilizza una o più porte (sia in ingresso che in uscita) tra loro collegabili in modo da costituire una rete, simile ad un diagramma di flusso. I blocchi, quindi, si scambiano flussi di dati denominati Information Packets. Questi ultimi trasportano informazioni in forma di dati o comandi che innescano l'avvio di un processo relativo al blocco di destinazione. Un altro approccio IoT-EUSC, per la creazione di programmi, è il Block-Based Programming [10]. Questo paradigma è una forma evoluta del Flow-Base Programming, in quanto le applicazioni sono costituite sempre da una serie di blocchi interconnessi tra di loro con operazioni drag-and-drop che rendono più facile la creazione di servizi.
SenSquare è una piattaforma middleware IoT per il monitoraggio ambientale, che raccoglie grandi fonti di dati eterogenei rilevati da sensori presenti sul territorio pubblico e privato, mettendoli a disposizione agli utenti che, tramite un client, generano servizi per migliorare la loro quotidianità.
Questi servizi sono creati dagli utenti attraverso IoT-EUSC con approccio Block-Based Programming. In SenSquare il flusso di dati è composto da una serie ordinata d'informazioni, ognuna costituita da un timestamp, da coordinate geografiche e dal valore misurato. Un singolo flusso di dati, che viene associato ad un sensore di rilevazione, non può costituire una base per EUSC poiché se il sensore viene spento senza preavviso, oppure viene spostato, l'istanza non è più utilizzabile. Per ovviare a questo inconveniente, si crea un "Servizio di Base", cioè un'entità astratta rappresentata dall'unione di un singolo tipo di dato (temperatura, qualità dell'aria, pressione, ecc.) e da una funzione di aggregazione (massima, minima, media, ecc.). Il "Servizio di Base" ritorna un singolo valore applicando la funzione di aggregazione sul tutto il flusso di dati rilevati in una specifica area geografica [7]. Un esempio di Servizio di Base è la Temperatura massima misurata a 100 metri dal duomo di Milano. Il beneficio principale di questo approccio è la creazione di servizi complessi a partire da quelli di base.
SenSquare utilizza Blockly[5], un plug-in creato da Google per creare templa-

te basati sulla metodologia Block-Based Programming. I blocchi che compongono il template vengono tradotti in codice python3, per poi essere eseguiti periodicamente con i flussi di dati richiesti.

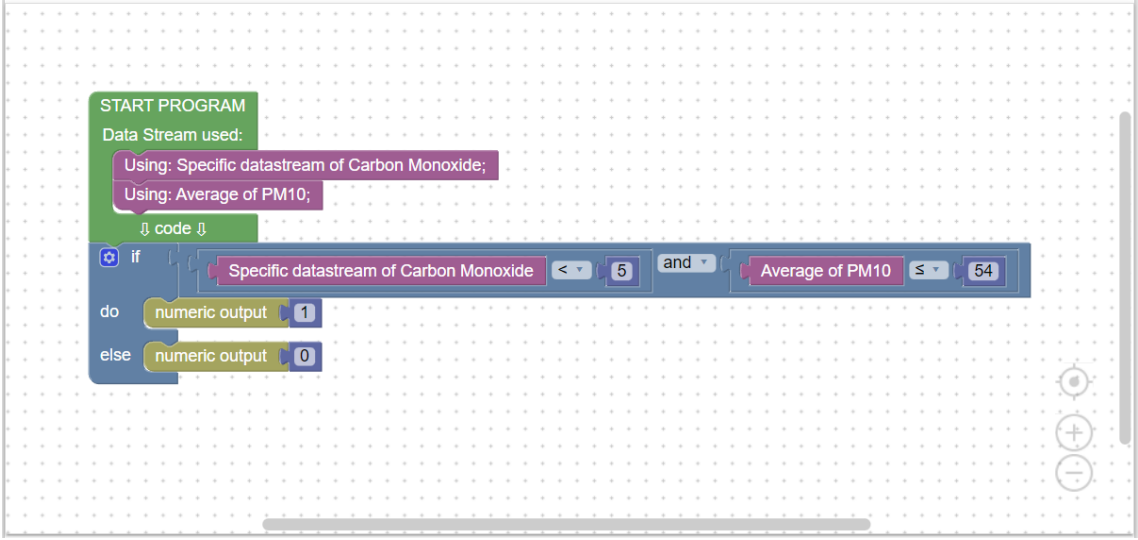

Figura 1.4: Esempio creazione di un servizio con Blockly in SenSquare

La figura 1.4 rappresenta l'utilizzo di Blockly all'interno di SenSquare. Il programma prende come input il datastream specifico del Carbon Monoxide e la media del PM10. Se il primo valore è inferiore a 5 e se la media del PM10 è inferiore o uguale a 54 restituisce il valore uno, zero in caso contrario.

1.3 Contributi

L'obiettivo di questa tesi è di modernizzare e migliorare la piattaforma SenSquare con nuovi standard tecnologici. La scelta di questo elaborato è stata dettata dalla obsolescenza di Angular, ormai quasi inutilizzata dalla maggioranza dei programmatori. Ciò si è rilevato un ottimo punto di partenza per valutare le diverse funzionalità dell'ultima versione del framework rispetto a quello iniziale.
Il contributo di questo lavoro di tesi è di aggiornare Angular per i motivi citati nei paragrafi precedenti. Dall'estate del 2018, Google ha reso disponibile l'utilizzo delle sue mappe solo a pagamento. Quindi, lo sforzo iniziale è stato quello di apprendere il funzionamento della prima release di Angular, cambiando integralmente le mappe preesistenti sul sito sviluppate da Google Maps con mappe gratuite open source. Per realizzare questo obiettivo, è stato necessario utilizzare una tecnologia fruibile gratuitamente agli utenti. La scelta si è indirizzata verso Leaflet (libreria JavaScript che permette di aggiungere al proprio sito web mappe geografiche interattive) per le sue molteplici funzionalità e per la sua ampia documentazione accessibile online.
Terminata questa fase preliminare, si è passati alla modernizzazione del sito generando una Responsive Web App.
Ciò è stato possibile grazie all'utilizzo di Ionic,un framework HTML5 open source, usato per creare applicazioni mobile ibride in grado di realizzare un design responsivo che si adatta graficamente in modo automatico al dispositivo con il quali viene visualizzato.
Dopo la modernizzazione di SenSquare, si è passati alla creazione di un algoritmo di caching che acceleri la visualizzazione dei datastreams all'interno delle mappe.

Capitolo 2

SenSquare

In questo capitolo verrà illustrata la piattaforma SenSquare originale, analizzando l'architettura e le funzionalità dell'applicazione ed evidenziando le problematiche.
Come accennato nel capitolo precedente, SenSquare è una piattaforma nata come un'efficiente soluzione dell'IoT Collaborativo, dove gli utenti partecipanti condividono dati. Tramite un'interfaccia grafica, la piattaforma permette la creazione e l'istanziazione di servizi personalizzabili. I due componenti principali di SenSquare sono i template e l'istanze. I primi sono il codice sorgente di un servizio e le successive sono l'istanziamento del template in un'area specifica. I proprietari dei vari template e istanze, posso decidere di condividerli con altri utenti della piattaforma.

2.1 Architettura

Come si può osservare dalla figura 2.1, l'architettura di SenSquare è divisa in tre parti:

- il Database
- il Server
- il Client

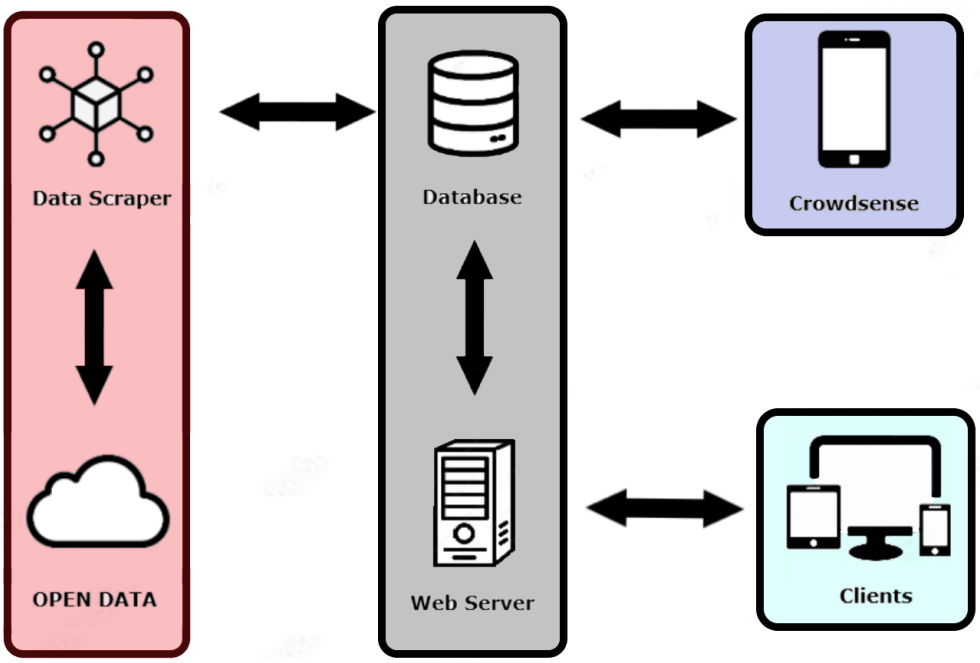

Figura 2.1: Architettura SenSquare

2.1.1 Database

Il database è il componente che immagazzina tutti i dati della piattaforma sviluppato con MySql composto dalle seguenti tabelle:

- **Devices**

  - `Device`: contiene le informazioni base dei dispositivi di rilevazione, specificando se sono delle stazioni di monitoraggio oppure altri tipi di sensori.
  - `DataStream`: contiene quali tipologie di misurazioni emette un device. Ad esempio se rileva la temperatura atmosferica o la qualità dell'aria.
  - `Measurement`: contiene tutti le misurazioni effettuati dai devices.
  - `DataClass`: contiene le possibili informazioni da assegnare ad una misurazione (tipo ed unità di misura).
  - `DataStreamOption`: contiene informazioni per specificare nella fase di creazione del template se l'utente vuole usare tutti i DataStream oppure uno solo nell'area a disposizione.

- **Template**
  - `CustomServiceTemplate`: contiene i dati dei template creati dagli utenti.
  - `CustomServiceTemplateRating`: contiene le valutazioni che gli utenti assegnano ad uno specifico template.
  - `CustomService`: contiene le informazioni relative alle istanze associate ai template.

- **User**
  - `Account`: contiene le informazioni di base degli utenti che si registrano alla piattaforma.
  - `Participant`: contiene le entità che utilizzano la piattaforma. Può essere un ente governativo creato dalla piattaforma per aggiungere servizi/misurazioni oppure un utente normale iscritto.

2.1.2 Server

Il server, scritto in python, nell'archiettura di SenSquare ricopre un ruolo fondamentale offrendo molti servizi. I compiti principali sono di far interagire, attraverso le sua API, il client con il database e di recuperare e organizzare dati IoT da risorse pubbliche.
Come un normale WebServer, è costituito da un'interfaccia conforme ai vincoli dello stile architetturale REST, che consente l'interazione del client per ricevere i dati per il suo corretto funzionamento. Inoltre, il server esegue giornalmente due metodi fondamentali al fine di tenere sempre aggiornato il suo database: il Data Gathering e il Data Annotation.

2.1.2.1 Data Gathering

Per Data Gathering si intende l'azione di recupero dalla rete internet i dati ambientali. Esso avviene in due diverse modalità: Open Data e Mobile CrowdSensing.

Open Data

Per Open Data si intendono tutte le informazioni accessibili liberamente e gratuitamente. I dati posso essere raccolti e immagazzinati in repository da sensori statici certificati (macchine di rilevazione ambientali fisse) o da un qualunque individuo. Pertanto, I repository si differenziano in "Affidabili" e "Inaffidabili".
Per "Affidabili" si intendono un set di dati gestiti da organizzazioni o governi, che certificano la validità. Purtroppo, questi dati non so reperibili in modo facile, ma bisogna eseguire una richiesta HTTP per estrarli dalle pagine web. In particolare, la maggior parte dei dati vengono estratti dal sito dell'Agenzia regionale per la protezione dell'ambiente della regione Emilia-Romagna (ARPA).
I dati "Inaffidabili" non garantiscono la loro veridicità poiché sono creati tramite il crowdsourcing, cioè da utenti che effettuano misurazioni con i loro dispositivi IoT. Questi dati sono in genere privi di etichetta, annotati male o

incompleti che richiedono una fase di elaborazione per classificarli e decidere quali usare. Tramite un algoritmo, si è riuscito ad estrarre da ThingSpeak, repository "inaffidabile", un set di dati validi per SenSquare [2].

Mobile CrowdSensing

La piattaforma sfrutta anche il Mobile CrowdSensing (MCS), ovvero paradigma utile alla raccolta di una grossa mole di dati con il minimo costo economico. E' stata creato un'applicazione android [4] che carica periodicamente i dati raccolti dai sensori del dispositivo. Nell'app vi è un algoritmo in grado di preservare la privacy degli utenti e di regolare la quantità di dati raccolti. In particolare, quando i dati sono scarsi in una determinata area, l'algoritmo richiede ai dispositivi mobile un contributo maggiore di dati e viceversa, limita il flusso di dati quando ve ne sono in abbondanza.

2.1.2.2 Data Annotation

Per Data Annotation si intende l'azione di identificare e classificare i dati eterogenei reperiti dal Data Gathering. In particolare, si occupa di dare un senso ai valori, assegnando ad ogni rilevazione un'unità di misura e un Data Type (come temperatura, qualità dell'aria, ecc.).

2.1.3 Client

Il client è stato sviluppato in Angular2, framework che permette di creare siti internet usando TypeScript, HTML5 e SCSS. Il client è il mezzo principale con cui le persone utilizzano la piattaforma SenSquare. Il sito è disponibile all'indirizzo `http://sensquare.disi.unibo.it/` ed è formato da 6 pagine principali, navigabili attraverso un menu a scomparsa o bottoni. Per descrivere e presentare al meglio il client della piattaforma, per ogni pagina web vi sarà una breve descrizione delle sue funzionalità, accompagnata da un immagine e dall'indirizzo della risorsa.

- **About SenSquare**
  - Url: <`Domain`>/about
  - è la prima pagina che si incontra entrando del sito. Tramite due bottoni, si aprono due *Dialog* che permettono all'utente di Registrarsi o effettuare il Login. In seguito, l'utente verra indirizzato nella schermata contente tutti i servizi.

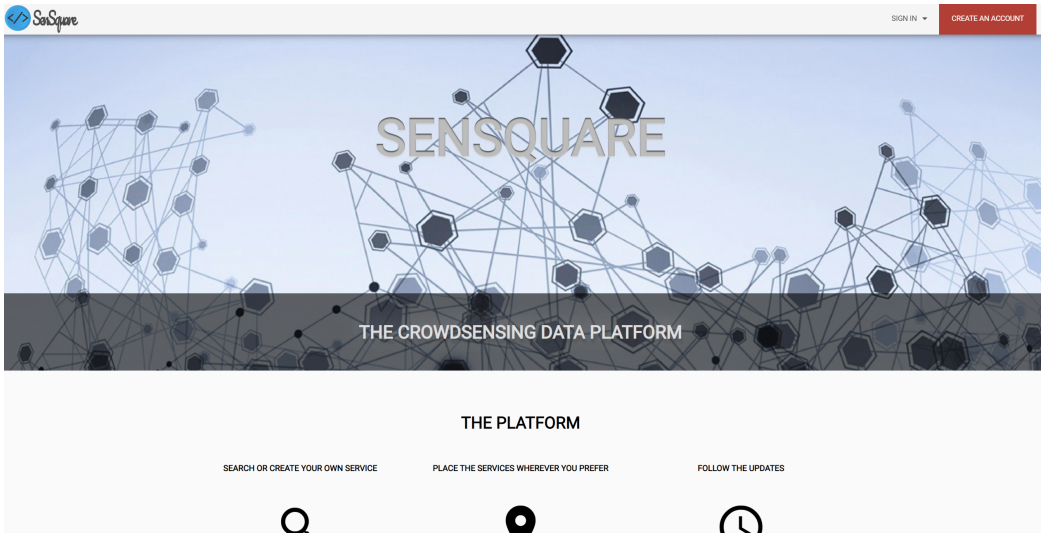

Figura 2.2: Pagina About SenSquare

- **Instances**
  - Url: <Domain>/instances
  - In questa pagina, l'utente può osservare tutti i servizi creati dalla community. L'utilizzatore può selezionare una card per visualizzare le informazioni del servizio scelto, oppure può andare nella pagina relative ai template.

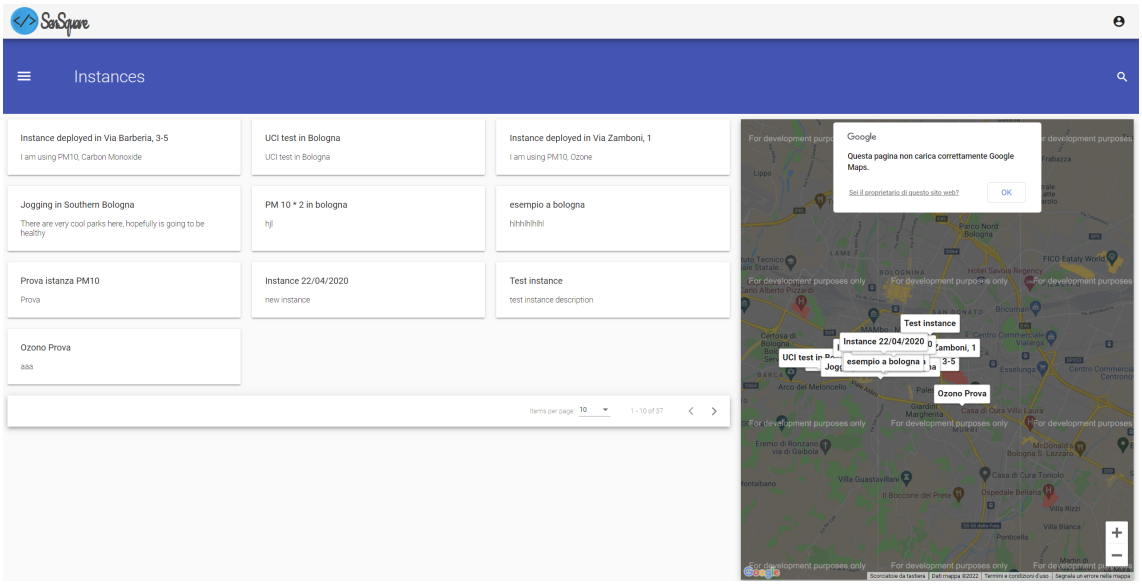

Figura 2.3: Pagina web con tutte i servizi attivi

- **Instance Detail**
  - Url: <Domain>/instances/<pk>
  - All'interno di questa pagina, l'utente potrà visualizzare i dettagli del servizio selezionato. In particolare, oltre alle informazioni base (titolo e descrizione), vi trova un grafico con i risultati calcolati dal servizio in un determinato tempo e una mappa contenente l'area in cui adopera il servizio. Tramite due bottoni, se l'utente è proprietario del servizio, potrà modificare o eliminare l'instanza. Premendo sul bottone *Edit*, si aprira un alert-dialog dove il proprietario del servizio può cambiare le coordinate geografiche di operatività dello stesso.

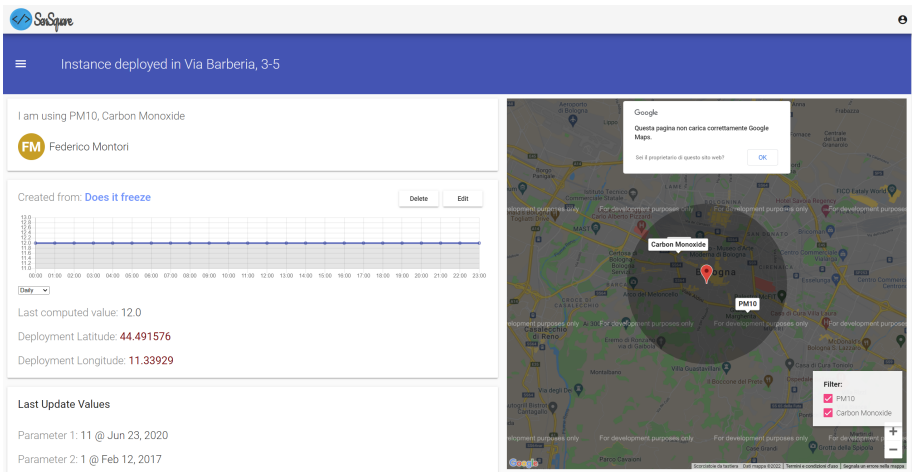

Figura 2.4: Pagina Web realtiva ad un Servizio

- **Edit Instance**
  - Url: <Domain>/instances/<pk>
  - Premendo il bottone *Edit*, si apre un Dialog come nella Figura 2.5 dove l'utente può cambiare le coordinate geografiche in cui far operare il servizio selezionando i nuovi datastream.

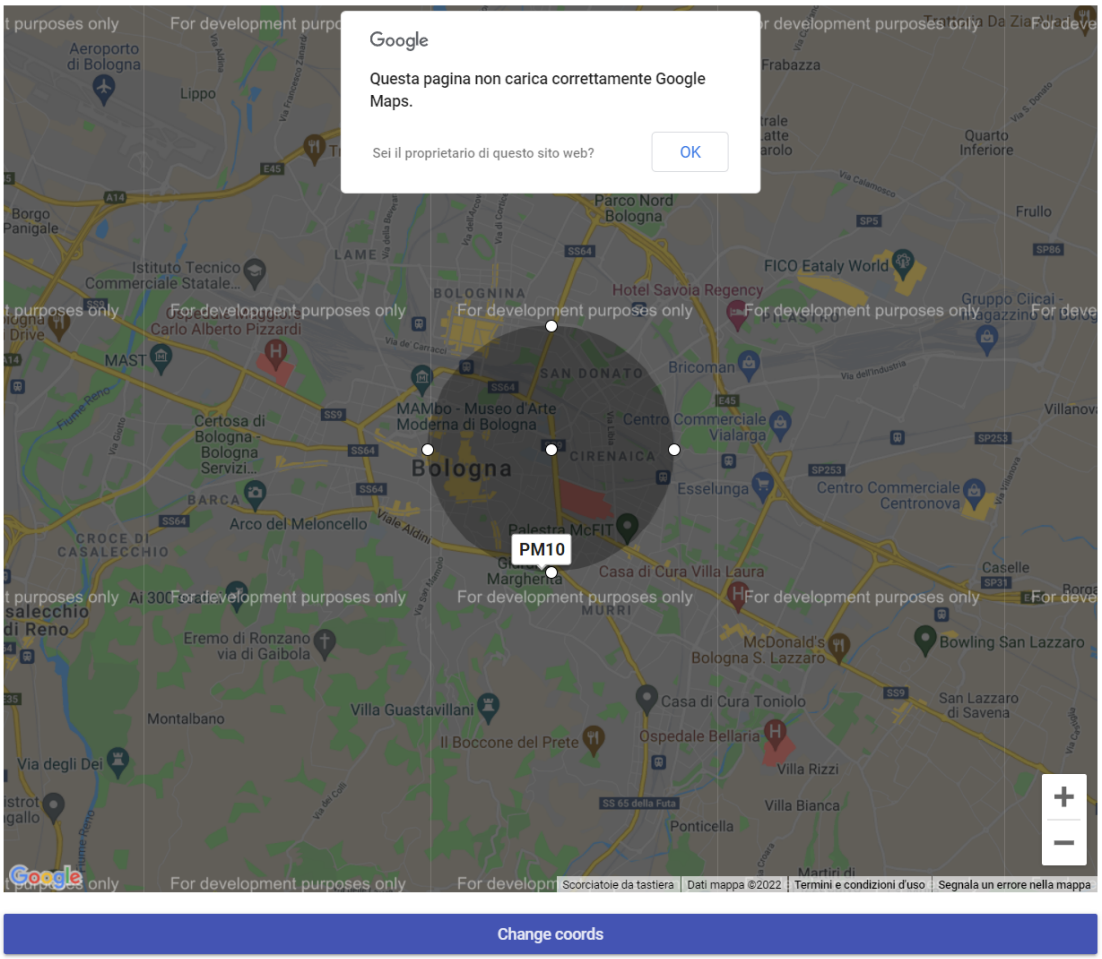

Figura 2.5: Pagina Web realtiva ad un Servizio

- **Templates**
  - Url: <Domain>/templates/
  - L'utente, mediante l'ausilio del menu a scomparsa, può visualizzare un elenco di tutti i template pubblici presenti su SenSquare. L'utilizzatore in questa pagina può creare un template o selezionare un template esistente per visualizzarne i dettagli.

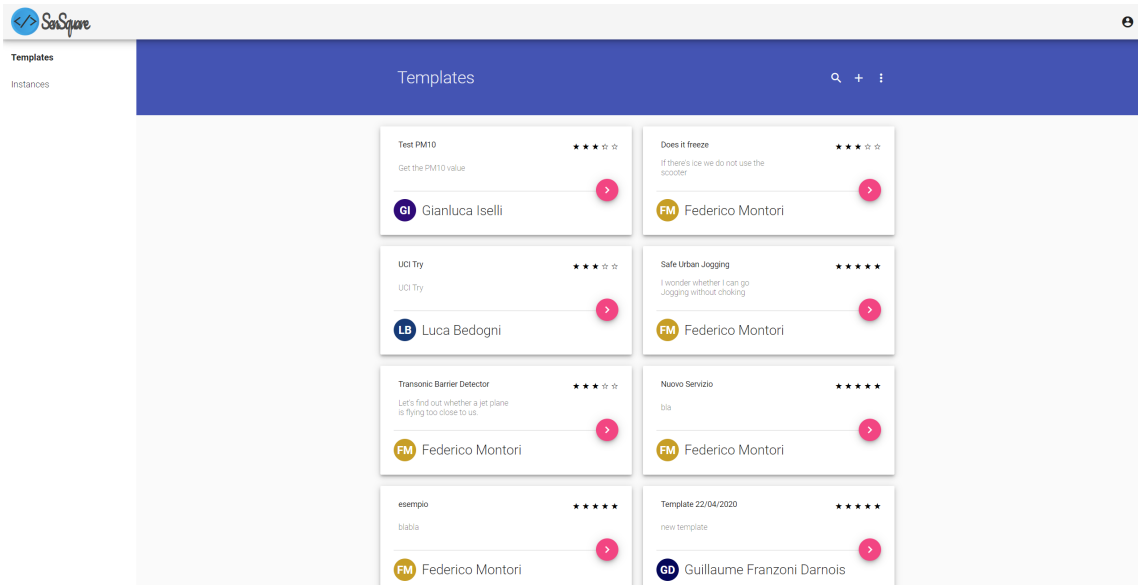

Figura 2.6: Pagina Web con tutti i Template disponibili nella piattaforma

- **Template Detail**
  - Url: <Domain>/templates/<pk>
  - All'interno di questa pagina, l'utente potrà visualizzare il codice del template attraverso Blockly o se è il proprietario del template, cliccando sugli opportuni pulsanti, potrà editare o eliminare lo stesso.

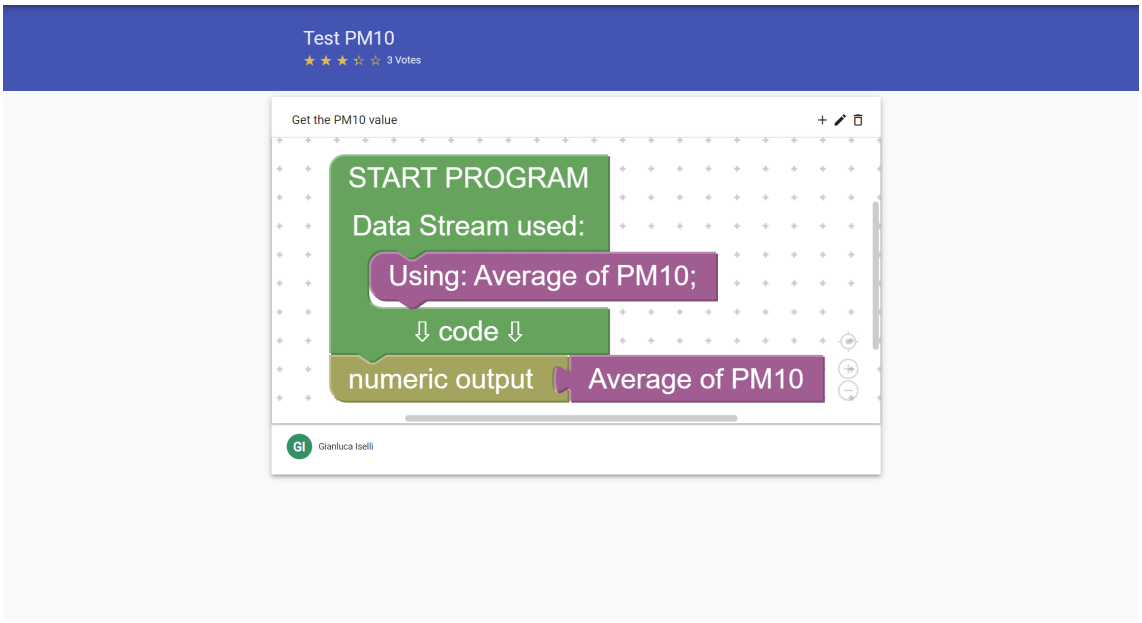

Figura 2.7: Pagina Web realtiva ad un Template

- **Create Template**
  - Url: <`Domain`>/templates/add
  - Con questa pagina, l'utente, attraverso l'editor Blockly, può creare un nuovo template da zero oppure a partire da un template esistente. Oltre al codice, deve inserire il titolo e una breve descrizione associata al template.

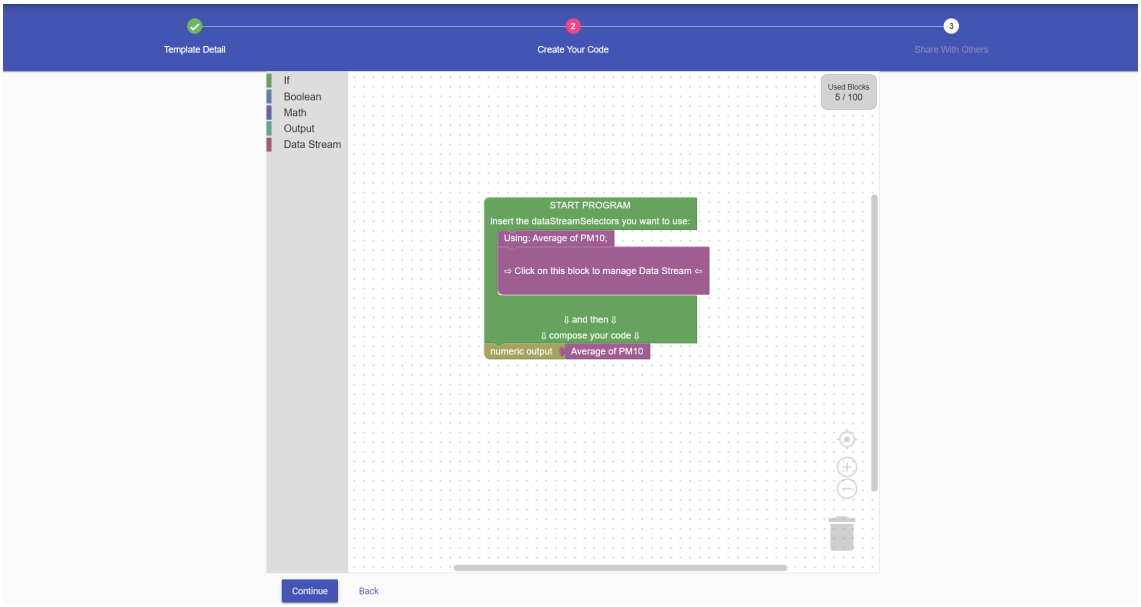

Figura 2.8: Pagina Web per creare un Template

- **Create Instance**
  - Url: <`Domain`>/instances/<`pk`>/add
  - In questa schermata l'utente può creare un nuovo servizio in due fasi: nella prima fase deve selezionare, attraverso la mappa, le coordinate dell'area in cui deve operare il servizio e i datastream necessari; nella seconda fase deve inserire il nome e una piccola descrizione dell'instanza.

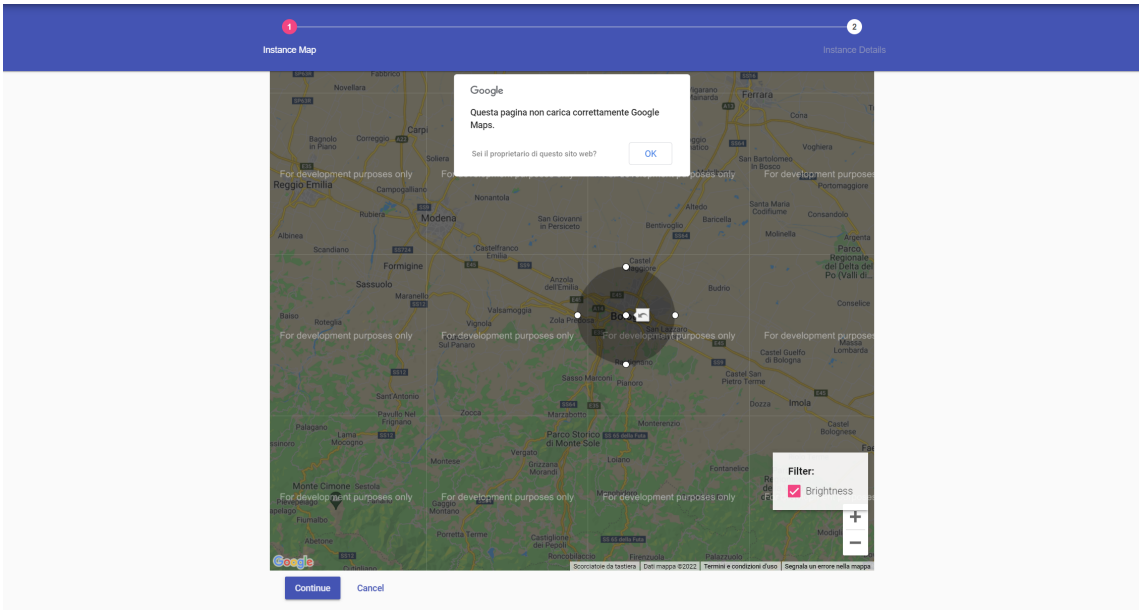

Figura 2.9: Pagina Web per creare un Servizio relativo ad uno specifico Template

2.2 Problematiche del sito originale

Il sito originale è il risultato di continui aggiornamenti non strutturali, mantenendo l'architettura di partenza ormai obsoleta. Il Front-end è risultata l'area che ha richiesto un immediato aggiornamento.
Come è evidente dalle immagini nella sezione precedente, le mappe complessive presentano la scritta "For development pourpose only", questo perché il servizio di Google Maps negli anni è diventato a pagamento rendendole poche visibili per non clienti di Google. Inoltre, il sito è stato sviluppato con una versione di Angular rilasciata nel 2017 e da allora sono uscite nuove versioni migliorative.
Infine, quando un utente seleziona l'area in cui far operare un servizio, il caricamento dei datastreams è lenta dando una sgradevole esperienza nell'utilizzo.

Capitolo 3

Implementazione

In questo capitolo non verranno trattati aspetti riguardanti l'implementazioni esistenti sulla piattaforma, ma solamente le scelte implementative delle funzionalità introdotte da questo elaborato. In dettaglio analizzeremo le modifiche all'aggiornamento del framework Angular e la sostituzione delle mappe Google Maps con Leafleat.

3.1 Aggiornamento Front-End: Angular v12

SenSquare è una piattaforma creata nel 2017 [6] con tecnologia dell'epoca. Per il Front-End è stato utilizzato Angular 2, software che tre anni fa era la punta di diamante nella creazione di Web App. Ad oggi, questa tecnologia risulta ormai obsoleta ed inefficiente e quindi c'è la necessita di aggiornare tutte le librerie del Front-End. La versione disponibile al tempo di questo elaborato è Angular 12. A causa dei numerosi aggiornamenti del framework, l'esecuzione del comando `ng update @angular/cli @angular/core` ha generato un elevato quantitativo di errori di sintassi nel codice. A causa di ciò è stato necessario creare un nuovo progetto per poi importare manualmente, con le dovute modifiche, tutte le funzionalità del progetto originario. Nelle sezioni successive descriverò passo per passo le azioni seguite per l'aggiornamento del sito.

3.1.1 Ionic

Il primo passo è stato la scelta della libreria da utilizzare. Dovendo ricreare da zero il progetto, si è scelto di utilizzare Ionic: framework per creare applicazioni ibride e Progressive Web Application, integrando i framework frontend più popolari, come ad esempio Angular. Ionic permette di fornire un'esperienza quanto più simile alle applicazioni native attraverso componenti che emulano i widget di sistema iOS e Android, o di tradurre l'applicazione web per creare app native per i dispositivi mobile. La creazione di un'applicazione nativa implica l'utilizzo di ambienti di sviluppo Android Studio e X-code e i linguaggi Java e Swift. E' molto raro, solitamente, trovare un programmatore che sia in grado di sviluppare app in entrambe le piattaforme. Ionic risolve questo problema traducendo automaticamente la Web App in applicazioni per dispositivi mobili mediante il plug-in Cordova[3].

3.1.2 Ambiente di sviluppo

Il secondo passo è stato quello di creare l'ambiente di sviluppo scaricando e installando i nuovi software per l'ideazione del Front-End. La Tabella 3.1 rappresenta le nuove tecnologie utilizzate con le rispettive versioni.

Nome	Versione
Ionic	6.16.3
Nodejs	14.17.3
Angular	12.0.5
npm	6.14.13

Tabella 3.1: Tabella contente le nuove versioni del progetto

3.1.3 Importazioni delle vecchie funzionalità

Preparato l'ambiente di sviluppo, il passo successivo è stato quello di ricreare tutte le pagine web con le loro funzionalità del sito originario. In

questo paragrafo descriverò come ho eseguito l'aggiornamento del codice nei tre linguaggio di programmazione di Angular (HTML, TypeScript, SCSS).

Modifiche al codice HTML

Il codice HTML è quello che ha subito una maggiore modifica. Nel codice originario si è usato Angular Material per sviluppare la parte visiva delle pagine Web. L'utilizzo di questa libreria comporta la scrittura di un elevato quantitativo di codice HTML causando un lavoro maggiore di modifica, se non si aggiorna progressivamente il codice alle nuove versioni di Angular Material. Pertanto ho scelto di non utilizzare Angular Material e di usare i componenti di Ionic, mantenendo inalterato lo stile delle pagine Web. Inoltre, un altro motivo per il non utilizzo di Angular Material è perché i componenti di Ionic sono meglio predisposti per la traduzione in codice Java e Swift, comportando meno lavoro al programmatore.
I principali componenti di Ionic utilizzati nel progetto sono:

- `ion-menu`, componente per creare menu a scomparsa usato per navigare all'interno del sito
- `ion-button`, componente per creare un elemento selezionabile
- `ion-list`, componente per creare una lista per ospitare all'interno elementi come bottoni, icone, ecc.
- `ion-input`, componente per creare un input dove un utente può inserire delle informazioni;
- `ion-label`, componente per visualizzare un testo semplice

Modifiche al codice SCSS

Il codice SCSS è stato modificato per ricreare il vecchio stile per i componenti di Ionic. Poiché alcune funzionalità nel sito originale erano inutilizzabili sui dispositivi mobili, sono stati inseriti due selettori che rilevano la dimensione in pixel del dispositivo in uso, permettendo di caricare il codice css adatto. In Figura 3.1 sono rappresentati i due selettori utilizzati.

```
@media only screen and (max-width: 600px) {
...
}
```

```
@media only screen and (min-width: 600px) {
...
}
```

Figura 3.1: A sinistra, il selettore carica codice css per dispositivi Mobile, a destra per dispositivi Desktop

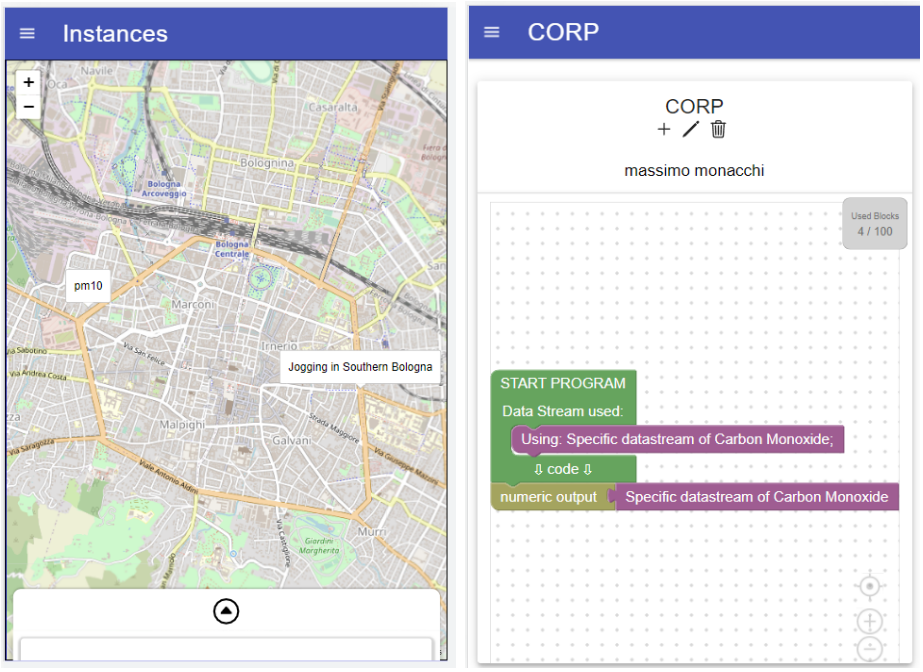

Figura 3.2: Schermate di esempio di due pagine web visualizzate da due dispositivi mobile

TypeScript

Nel sito originale si utilizza TypeScript 2.3. Angular 12, invece, richiede TypeScript 4.2 e versioni successive. Per cui si è reso necessario il refactoring del codice, in cui è stata rinnovata la sintassi del codice e dei moduli utilizzati.

3.2 OpenStreetMap

Come accennato nei capitoli precedenti, SenSquare utilizza il servizio di Google Maps per aggiungere mappe all'interno del sito, diventato a pagamento nel 2018. In alternativa, si è deciso di utilizzare OpenStreetMap: librerie open source che offrono servizi di mappe interattive. In particolare, la scelta è ricaduta sulla libreria Leaflet, la migliore fra tutte quelle disponibili per la sua ampia documentazione e facilità d'uso. Nelle sezioni successive illustrerò le modifiche apportate alla versione originale.

3.2.1 Installazione e importazione di Leaflet

Come primo passo occorre scaricare i moduli di Leaflet e le sue dipendenze via npm digitando nella console i seguenti comandi:

- `npm install Leaflet`
- `npm install @asymmetrik/ngx-Leaflet`

Una volta scaricate le librerie, occorre eseguire delle azioni preliminari per aggiungere Leaflet al progetto e per garantirne il corretto funzionamento:

- **Import Module**-permette l'utilizzo delle libreria nel programma. Grazie ad Angular 12, non vi è più la necessità di aggiungere l'importazione ad ogni componente del progetto che lo utilizza, ma è sufficiente inserire il codice mostrato dalla Figura 3.3 nel file modulo radice dell'app chiamato `app.module.ts`

```
import { LeafletModule } from '@asymmetrik/ngx-leaflet';

...
imports: [
        ...
        LeafletModule
]
...
```

Figura 3.3: Linea di codice per aggiungere il modulo Leaflet

- **Stylesheet**-la Figura 3.4 mostra come importare il Leaflet SCSS: il foglio di stile che permette di visualizzare la grafica di Leaflet. Esso si aggiunge alla matrice di stili contenuta nel file `angular.json`

```
{
        ...
        "styles": [
                "styles.css",
                "./node_modules/leaflet/dist/leaflet.css"
        ],
        ...
}
```

Figura 3.4: Linea di codice per aggiungere al progetto Leaflet SCSS

3.2.2 Creazione e Configurazione della mappa

In questa sezione descriverò il codice per aggiungere e configurare una mappa all'interno di una pagina Web (applicabile ad ogni mappa del sito). Nel progetto ogni mappa è un componente di Angular, cioè un elemento che

compone l'interfaccia grafica composto da un file HTML contenente ciò che viene visualizzato nella pagina, da un documento TypeScript che ne definisce il suo comportamento, da un file SCSS che definisce lo stile e, per ultimo, da un file contenente i moduli non importati dal componente padre.
Nel file .html del componente, è definito un tag HTML che funge da contenitore della mappa. La Figura 3.5 rappresenta il tag in questione

```
<div
     leaflet
     class="map"
     id="..."
     (leafletZoom)="zoom"
     (leafletMapReady)="onMapReady($event)"
     [leafletOptions]="options">
 </div>
```

Figura 3.5: Tag HTML Leaflet

che ha i seguenti attributi:

- `Leaflet`, per specificare al DOM che l'elemento conterrà la mappa Leaflet
- `class`, usato per attribuire all'elemento HTML uno stile (es width, height, ecc..)
- `LeafletZoom`, per settare lo zoom di partenza della mappa
- `leafltetMapReady`, per assegnare quale funzione chiamare una volta caricata la mappa (in questo caso `function onMapReady()`)
- `LeafletOptions`, opzioni per la creazione della mappa

Il codice TypeScript e SCSS è rimasto pressoché invariato richiedendo soltanto adattamenti alle nuove mappe, tranne per le seguenti funzionalità.

Marker e Pop-up

Al contrario di Google Maps, per creare un marker o un popup personalizzato in Leaflet, non basta solo modificare il codice SCSS e HTML, ma si deve creare dei componenti Angular assestanti per poi attribuirli alla mappa stessa. Nei file TypeScript associati alle mappe, vi sono due funzioni: **createMarker()** e **createPopup()**. Entrambi, servono per creare e aggiungere i marker e pop-up ad una mappa. Il modulo `ComponentFactoryResolver` converte i componenti marker e popup in variabili per poi assegnarli alla mappa tramite le due funzioni di Binding: `L.divIcon(..)` e `L.popup(..)`.

Cerchio

In alcune mappe, l'utente ha la possibilità di selezionare l'area operativa associata a un servizio mediante una circonferenza dimensionabile a piacere. In Google Maps, questa operazione si ottiene chiamando una funzione dedicata. Al contrario, in Leaflet si è dovuto usare un modulo specifico che permette di creare cerchi dimensionabili. Quando il cerchio viene spostato oppure cambia dimensione automaticamente, viene chiamata una funzione che carica i datastreams nella nuova area.

Capitolo 4

Ottimizzazione spaziale

4.1 Problema

Come anticipato nella Sezione 2.2, SenSquare presenta delle funzionalità lente offrendo un'esperienza poco gradevole all'utente. Quando un utente vuole creare un'istanza, deve selezionare l'area operativa di essa tramite una circonferenza presente sulla mappa. Ad ogni modifica del circonferenza, il client esegue una chiamata al server per recuperare tutti i datastreams presenti in quella determinata area. Se quest'ultima è grande, il server deve eseguire un calcolo computazionale oneroso per ricercare all'interno del database i datastreams necessari.

4.2 Soluzione

Per risolvere il problema appena descritto, è stato implementato un algoritmo di caching che salva l'aree e datastream già visitate dall'utente, velocizzando il caricamento dei devices sulla mappa. Prima di spiegare come funziona l'algoritmo, è doveroso introdurre le tre possibili azioni che può eseguire l'utente con la circonferenza:

- **AUMENTO**

  Come mostra la Figura 4.1, per *aumento* si intende l'azione d'espansione della superficie operativa dell'istanza.

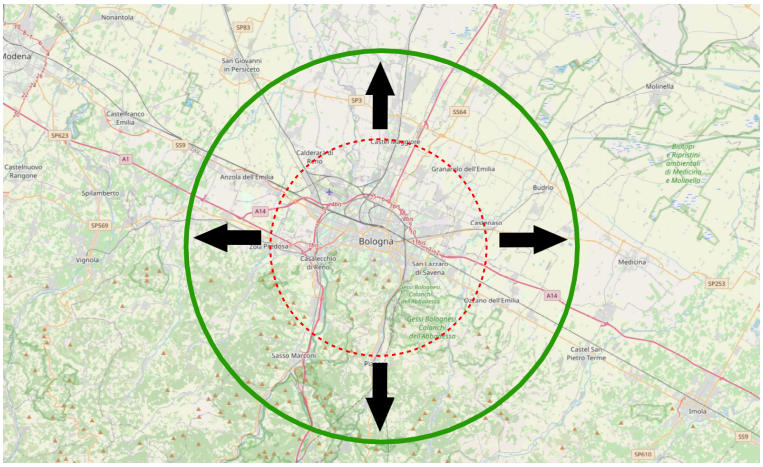

Figura 4.1: Esempio d'aumento area del cerchio

- **RIDUZIONE**

  Come mostra la Figura 4.2, per *riduzione* si intende l'azione di diminuzione della superficie operativa dell'istanza.

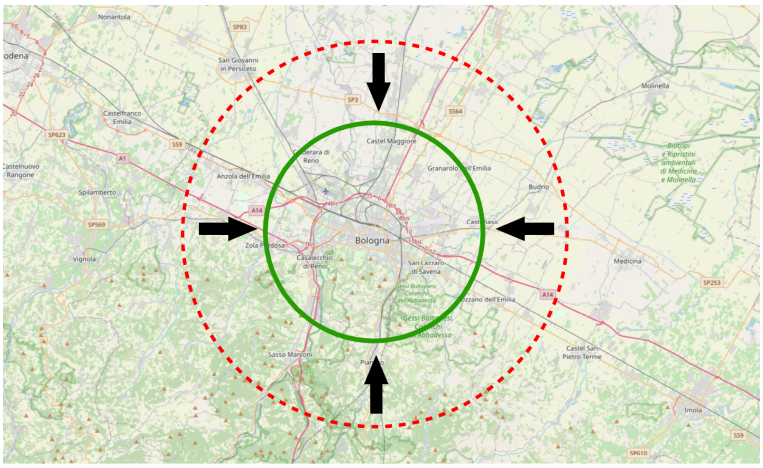

Figura 4.2: Esempio di riduzione area del cerchio

- **SHIFT**

  Come mostra la Figura 4.3, per *shift* si intende l'azione di traslazione della superficie operativa dell'istanza.

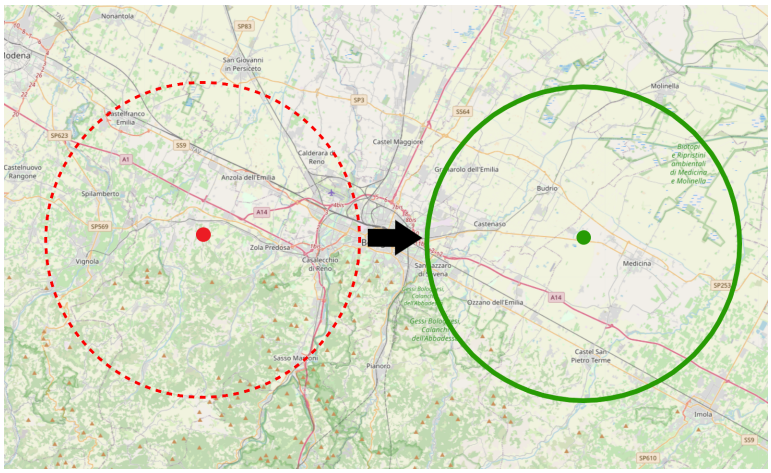

Figura 4.3: Esempio di shift del cerchio

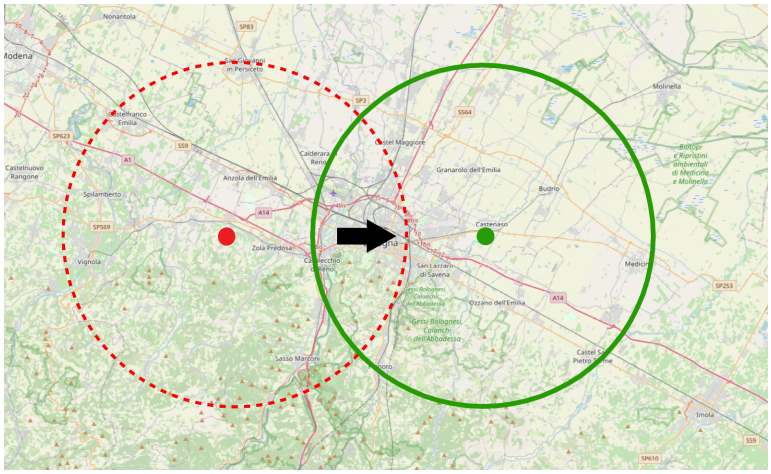

Figura 4.4: Esempio di shift del cerchio con intersezione

4.3 Algoritmo dell'ottimizzazione spaziale

L'idea di base dell'algoritmo di ottimizzazione è la memorizzazione dei datastreams nel web browser da utilizzare successivamente ad ogni cambio dell'area operativa, evitando così ulteriori chiamate al server.
La classe che gestisce il caching dei dati utilizza due dizionari: `listMap` che memorizza tutte le aree che l'utente ha già visitato e `listDataStream` che memorizza i datastream ricevuti dal server. I dizionari sono costruiti in modo tale da associare ad ogni chiave una lista di dati. Inoltre, la classe utilizza il modulo Turf composto da librerie che permettono operazioni tra aree geospaziali.
Supponiamo che entrambi i dizionari siano vuoti e che l'utente si trova a creare il suo primo servizio. All'apertura della pagina, viene caricato sulla mappa un cerchio di default e viene invocata la funzione `getData()` (appartenente alla classe di caching) che verifica se l'area generata dal cerchio sia già stata visitata in passato dall'utente.

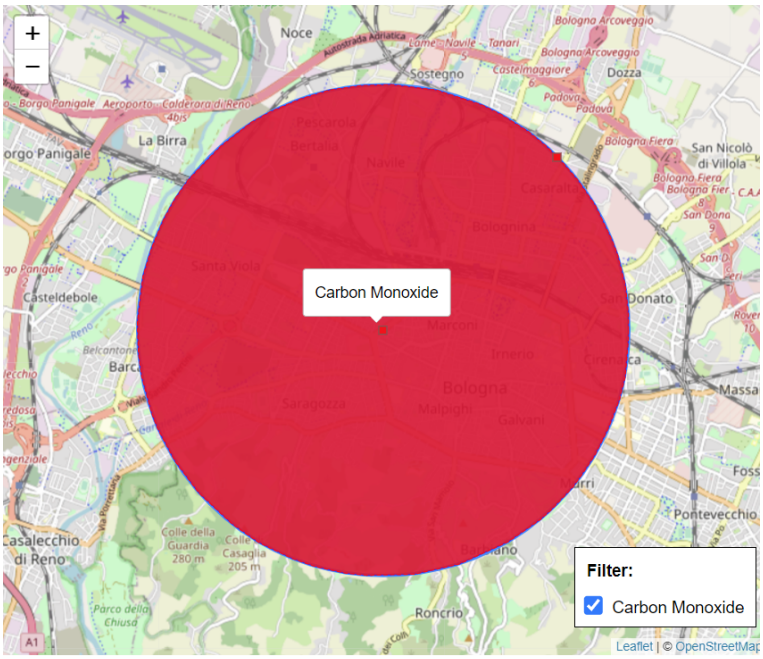

Figura 4.5: Situazione cerchio default con dizionari vuoti

Essendo i dizionari entrambi vuoti, l'area del cerchio è da considerasi nuova. Questa situazione è rappresentata in Figura 4.5 dove la superficie in rosso è la nuova area. Quando il client riceve i datastreams, quest'ultimi e la superficie vengono salvati nei rispettivi dizionari usando i *datatype* dei datastream come chiave per mappare i valori. Successivamente, se l'utente ridimensiona o trasla il cerchio, l'algoritmo effettua sequenzialmente queste operazioni:

1. Individuazione area comune
   Usando come chiave i *datatype* dell'istanza, si prendono tutte le mappe che si trovano nel dizionario e attraverso la funzione `intersect(..)` si individua se la superficie del cerchio ha delle parti in comune con tutte le mappe salvate precedentemente.(nella Figura 4.6, corrisponde all'area verde)

2. Individuazione area non in comune
   Se la nuova area non interseca con nessuna area precedentemente visitata, l'area non comune corrisponde alla tutta la superficie generata dal cerchio. Altrimenti, si usa la funzione `difference(...)` per individuare l'area non in comune.(nella Figura 4.6, corrisponde all'area rossa)

3. Ricerca dei datastreams
   La ricerca dei datastreams si esegue sequenzialmente in base alle due aree individuate nei punto 1 e 2. Se esiste un area "in comune" si ricerca nel dizionario `listDataStream` i devices che sono in quell'area. Se esiste l'area "non in comune", si effettua una chiamata al server per ricevere i nuovi dati. Se esistono entrambi le aree, si effettua prima la ricerca nel dizionario i datasreams che si trovano nell'area "comune" e successivamente si effettua la chiamata al server specificando quali datastreams sono stati trovati precedentemente in modo tale che il back-end li escluda dalla query.

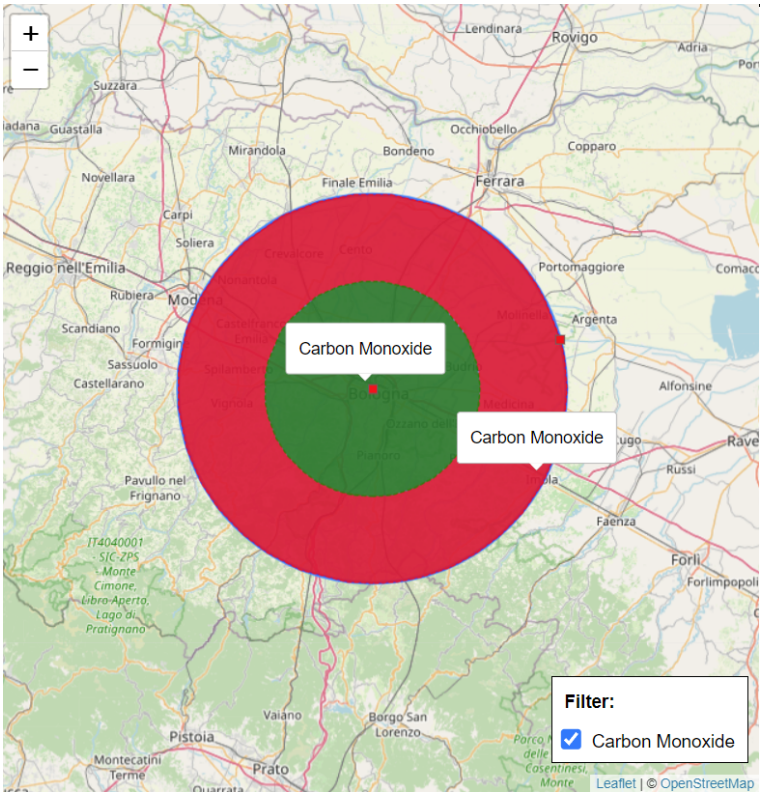

Figura 4.6: Esempio di identificazione da parte dell'algoritmo delle aree già esplorate (in verde) e delle aree inesplorate (in rosso)

4. Salvataggio delle nuove informazioni
   I dizionari si aggiornano solamente se è stata individuata un area "non comune": la nuova area generata dal cerchio viene unita a tutte le mappe che interseca, tramite la funzione `union(..)`. Invece, i datastream vengono salvati nell'array del dizionario a cui si riferisce la chiave *datatype*. Per non occupare troppa memoria nella macchina dell'utente, se quest'ultimo array è piu lungo di 50 unità, vengono eliminati dalla lista i datastreams che si trovano più lontano rispetto al centro dell'ultimo cerchio.

I metodi che gestiscono il caching dei dati si trovano all'interno di un servizio `Dependency Injection` che permette di condividere i dizionari tra tutte le pagine web dell'applicazione. Inoltre, i dizionari vengono salvati nella

cache del client tramite la funzione `sessionStorage`, che viene cancellata ad ogni nuova sessione del browser. Ciò richiede una nuova chiamata al server ad ogni nuova sessione del client garantendo aggiornata la visualizzazione dei datastreams.

Capitolo 5

Risultati

In questo capitolo verranno analizzati i risultati ottenuti grazie alle migliorie apportate da questo elaborato. Originariamente l'applicazione presentava diverse problematiche: il servizio di Google Maps non più utilizzabile, l'arretratezza delle librerie del Front-end e la lentezza di alcune funzionalità del sito.
L'aggiornamento di Angular e l'inserimento delle mappe Open-Street-Map ha reso il Front-end conforme agli standard più recenti migliorando la User Experience del sito. Il nuovo SenSquare è stato testato in tutte le sue funzionalità sui browser più utilizzati con esito positivo. La lentezza del caricamento dei datastreams è stato migliorata realizzando un nuovo algoritmo di ottimizzazione. La sua efficacia è stata verificata misurando i tempi di caricamento delle tre operazioni possibili che l'utente può fare sulla mappa, quando inserisce una nuova istanza (aumento, riduzione e shift). In particolare in ognuna di queste operazioni sono state eseguite le misure della velocità media di caricamento dei datastreams nelle seguenti condizioni: sito originale senza algoritmo, sito con il nuovo algoritmo di ottimizzazione con cache iniziale vuota e piena. Nelle seguenti tabelle e grafici, sono riportati i tempi di caricamento medi in secondi dei datastreams relative alle operazioni aumento, riduzione e shift nei tre casi precedenti descritti. **L'algoritmo ha ridotto notevolmente i tempi di caricamento dei datasreams fino al**

98% rispetto all'applicazione originale.

OPERAZIONE AUMENTO

Raggio dell'area (m)	Senza algoritmo	Con algoritmo, cache vuota	Con algoritmo, cache piena
1000	1,231	1,040	0,026
1500	1,510	0,992	0,0371
2250	1,152	0,743	0,031
3375	2,659	1,768	0,0499
5062	2,440	2,002	0,0501
7590	2,662	1,732	0,0501
11385	2,835	2,396	0,056
17078	3,050	2,118	0,0571

Tabella 5.1: Tabella contenente i tempi di caricamento dei datastreams in secondi dell'operazione Aumento. I raggi presi in considerazione sono stati calcolati in modo tale che il cerchio successivo abbia l'area doppia del precedente.

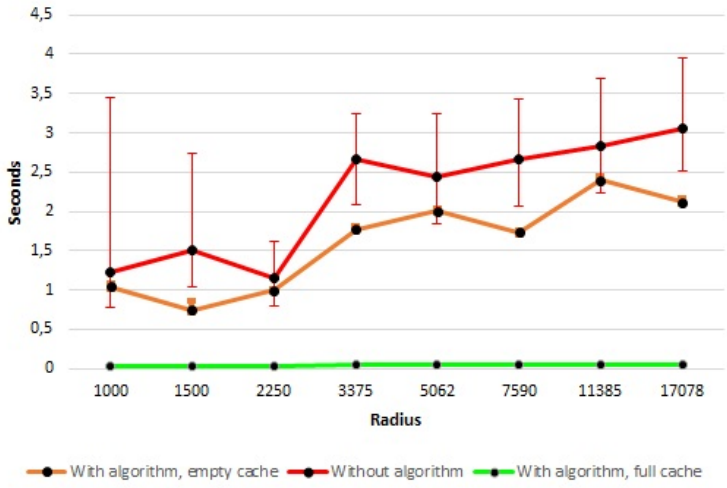

Figura 5.1: Rappresentazione grafica della Tabella 5.1

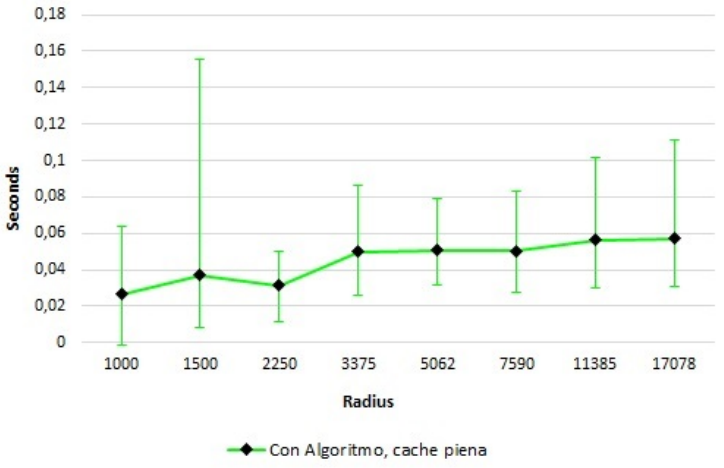

Figura 5.2: Grafico dei tempi di caricamento dei datastreams usando l'algoritmo con cache piena relativa alla Tabella 5.1

OPERAZIONE RIDUZIONE

N° Posizione Cerchio	Senza algoritmo	Con algoritmo, cache vuota	Con algoritmo, cache piena
17078	3,050	2,390	0,010
11385	2,835	0,011	0,010
7590	2,662	0,012	0,010
5062	2,440	0,0118	0,009
3375	2,659	0,010	0,009
2250	1,152	0,009	0,010
1500	1,510	0,010	0,0101
1000	1,231	0,012	0,010

Tabella 5.2: Tabella contenente i tempi di caricamento dei datastreams in secondi dell'operazione Shift spostando il centro della circonferenza del cerchio di un raggio in modo da intersecare l'area del cerchio precedente.

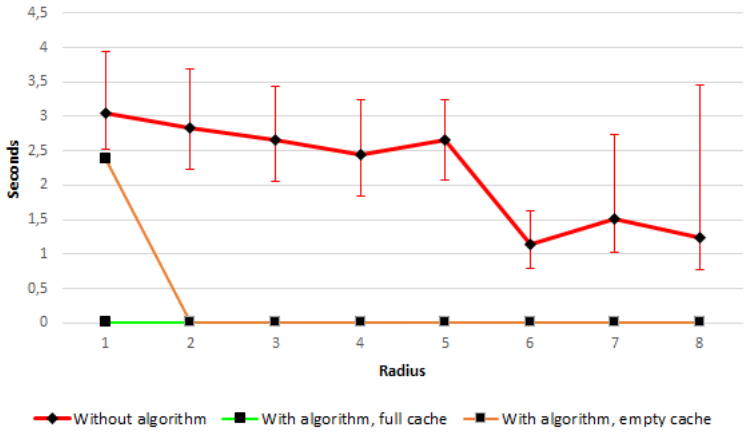

Figura 5.3: Grafico dei tempi di caricamento dei datastreams usando l'algoritmo con cache piena relativa alla Tabella 5.2

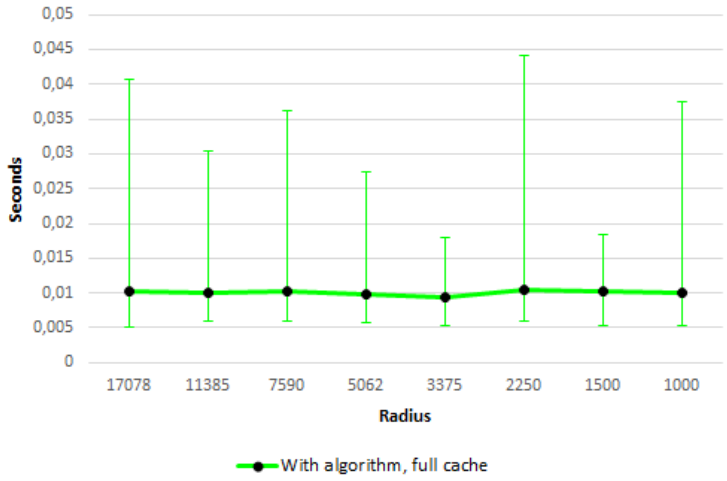

Figura 5.4: Grafico dei tempi di caricamento dei datastreams usando l'algoritmo con cache piena relativa alla Tabella 5.2

OPERAZIONE SHIFT

N° Posizione Cerchio	Senza algoritmo	Con algoritmo, cache vuota	Con algoritmo, cache piena
1	1,975	1,374	0,015
2	1,001	0,724	0,015
3	1,401	0,806	0,013
4	1,128	0,838	0,011
5	1,316	0,689	0,012
6	2,010	1,550	0,011
7	1,474	0,860	0,011
8	2,189	1,818	0,010
9	1,008	0,403	0,010
10	1,345	0,999	0,010
11	0,715	0,126	0,010
12	0,749	0,438	0,011
13	1,416	0,912	0,011
14	1,072	0,444	0,011
15	1,384	1,039	0,011

Tabella 5.3: Tabella contenente i tempi di caricamento dei datastreams in secondi dell'operazione Shift spostando il centro della circonferenza del cerchio di un raggio in modo da intersecare l'area del cerchio precedente.

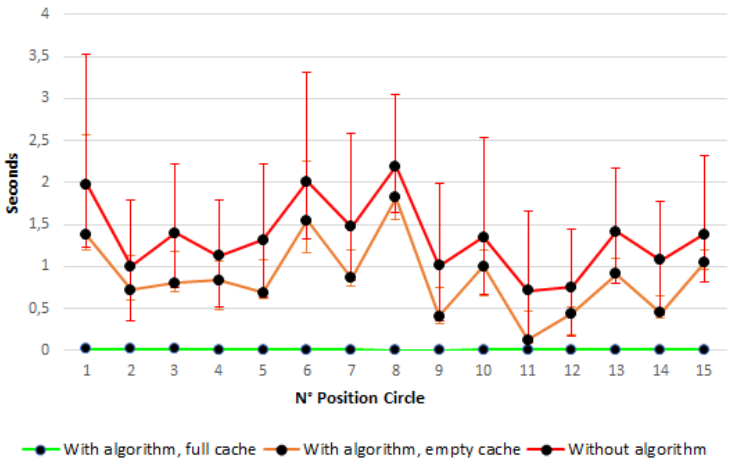

Figura 5.5: Rappresentazione grafica della Tabella 5.3

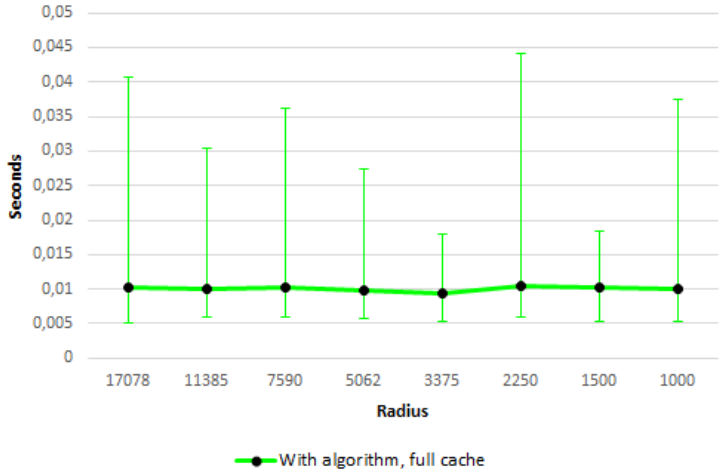

Figura 5.6: Grafico dei tempi di caricamento dei datastreams usando l'algoritmo con cache piena relativa alla Tabella 5.3

Capitolo 6

Conclusione

Lo scopo di questo elaborato è stato quello di migliorare la consumer experience dell'applicazione SenSquare, implementando un nuovo algoritmo di caching, per velocizzare il caricamento dei datastreams e anche di aggiornare l'intero Front-end dell'applicazione, secondo gli ultimi standard tecnologici.
Nei primi capitoli della tesi sono stati introdotti i concetti di Progressive Web Apps mediante Angular e di IoT End-User Service Composition (IoT-EUSC), un modello per facilitare l'utilizzo dei dati raccolti da dispositivi intelligenti da parte di utenti non esperti nel campo informatico. In seguito è stato presentato SenSquare descrivendo il suo modello architetturale, le funzionalità che offre agli utenti e le sue problematiche. Dopo questa introduzione , l'elaborato espone le principali soluzioni migliorative di SenSquare.
Il primo passo è stato quello di aggiornare Angular, di aggiungere il framework Ionic, di migliorare il Front-End creando un sito aperto a tutti i dispositivi (Desktop e Mobile) e di sostituire le Google Maps (proprietarie) con Open-Street-Map (Open Source).
Il secondo passo è stato quello di proporre e implementare un algoritmo di ottimizzazione per velocizzare il caricamento dei datastreams sulle mappe mediante il caching.
Grazie a questo algoritmo, la visualizzazione dei datasreams è istantanea, ottenendo una riduzione dei tempi di caricamento fino al 98% rispetto alla

soluzione precedente.

SenSquare è una piattaforma che offre innumerevoli scenari da approfondire e svariate migliorie da apportare ad essa. Uno di queste migliorie è l'aggiornamento delle librerie che compongono il Back-End, secondo gli ultimi standard tecnologici. Anche il database richiederebbe una riprogettazione per velocizzare le query. In conclusione possiamo affermare che tutti gli obiettivi prefissati sono stati raggiunti potenziando il progetto SenSquare e aumentando l'apprezzamento del consumatore nel suo utilizzo.

Bibliografia

[1] AngularJs. AngularJs site: `https://angularjs.org/`.

[2] JP. Calbimonte, Oscar Corcho, Zhixian Yan, H. Jeung, and K. Aberer. Deriving semantic sensor metadata from raw measurements. 904:33–48, 2012.

[3] Cordova. Cordova: Target multiple platforms with one code base: free and open source `https://cordova.apache.org/`.

[4] Guillaume Franzoni Darnois. *Progettazione e integrazione di un sistema di Mobile Crowdsensing con una piattaforma IoT orientata ai servizi.* PhD thesis.

[5] Google. Blockly - a library for building visual programming editors. [Online]. Available: `https://developers.google.com/blockly/guides/overview` (Accessed 2017-11-30).

[6] Gianluca Iselli. *SenSquare: una piattaforma IoT di crowdsensing e sviluppo collaborativo di servizi personalizzati.* PhD thesis.

[7] Federico Montori, Vincenzo Armandi, and Luca Bedogni. Iot end-user service composition via a visual programming interface. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 401–403, 2021.

[8] Federico Montori, Luca Bedogni, and Luciano Bononi. A collaborative internet of things architecture for smart cities and environmental monitoring. *IEEE Internet of Things Journal*, 5(2):592–605, 2017.

[9] Tomasz Szydlo, Robert Brzoza-Woch, Joanna Sendorek, Mateusz Windak, and Chris Gniady. Flow-based programming for iot leveraging fog computing. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 74–79, 2017.

[10] Tomasz Szydlo, Robert Brzoza-Woch, Joanna Sendorek, Mateusz Windak, and Chris Gniady. Flow-based programming for iot leveraging fog computing. In *2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 74–79, 2017.

[11] S Tandel and Abhishek Jamadar. Impact of progressive web apps on web app development. *International Journal of Innovative Research in Science, Engineering and Technology*, 7(9):9439–9444, 2018.

[12] Liping Zhao, Pericles Loucopoulos, Evangelia Kavakli, and Keletso J. Letsholo. User studies on end-user service composition: A literature review and a design framework. *ACM Trans. Web*, 13(3), jul 2019.