Sommario

[3 Design 1](#_Toc105962081)

[3.1 Scenari d’uso 1](#_Toc105962082)

[3.2 Requisiti 4](#_Toc105962083)

[3.3 Scelte progettuali 7](#_Toc105962084)

[3.3.1 Tecnologie 7](#_Toc105962085)

[3.3.2 Motivazioni 10](#_Toc105962086)

[3.3.3 Architettura e sviluppo 12](#_Toc105962087)

# Design

## Scenari d’uso

Per capire in che modo strutturare la creazione del sistema a seconda del tipo di utilizzo che ne potrebbero fare gli utenti, sono stati introdotti gli scenari.

Gli scenari d’uso possibili sono:

Scenario 1 – Visualizzazione e interazione della mappa corematica in città da parte di un cittadino

In questo scenario d’uso, il cittadino si trova in città e interagisce con uno dei totem touchscreen disponibili e vuole vedere l’indice di felicità del distretto in cui abita. Sul totem è in esecuzione la web-app che mostra una mappa interattiva.

Il cittadino, clicca sul distretto in cui vive che è presente sulla mappa, rappresentato da un corema con un’icona colorata di una casa, e gli vengono mostrati i fenomeni urbani, sotto forma di coremi, che caratterizzano il distretto in cui vive.

Il cittadino, poi, per capire meglio ciò che sta visualizzando, clicca sul pulsante che mostra la legenda dei coremi. La legenda mostra il significato dei colori, con il relativo valore numerico e letterale, e tutti i coremi.

Il cittadino clicca su un corema su cui vuole avere più informazioni e gli viene mostrato il nome, il significato e il tipo del corema cliccato.

Scenario 2 – Visualizzazione e modifica della mappa corematica da parte di un amministratore cittadino

In questo scenario d’uso, l’amministratore cittadino si trova in ufficio, utilizza un tablet touchscreen e vuole modificare l’indice di felicità dei fenomeni urbani.

L’amministratore cittadino digita l’indirizzo web della web-app e clicca invio. Gli viene mostrata una pagina web con la mappa corematica con i distretti cittadini rappresentati da coremi con un icona colorata di una casa; clicca su un distretto e gli vengono mostrati i fenomeni urbani, sotto forma di coremi, che caratterizzano il distretto cliccato. Successivamente clicca sul pulsante menù e gli mostrati tutti i fenomeni urbani presenti nel distretto cliccato, con il relativo indice di felicità, rappresentato sotto forma di valore numerico, modificabile tramite uno slider e un pulsante che permette il salvataggio delle modifiche effettuate. L’amministratore modifica l’indice di felicità di alcuni fenomeni urbani e clicca sul bottone per salvare le modifiche effettuate. Il sistema mostra un messaggio che le modifiche sono state effettuate con successo, dopodiché il sistema elabora i cambiamenti che sono stati effettuati e aggiorna l’indice di felicità dei distretti e dei fenomeni urbani coinvolti.

Scenario 3 – Filtraggio dei distretti sulla mappa corematica da parte di un cittadino

In questo scenario d’uso, il cittadino si trova a casa, utilizza un tablet touchscreen e vuole filtrare i distretti cittadini in base al valore dell’indice di felicità

Il cittadino digita l’indirizzo web della web-app e clicca invio. Gli viene mostrata una pagina web con la mappa corematica con i distretti cittadini rappresentati da coremi con un’icona colorata di una casa. Il cittadino, poi, preme il pulsante per filtrare i distretti, il sistema gli mostra un menù con uno slider che rappresenta l’indice di felicità e un pulsante per confermare l’operazione di filtraggio. Il cittadino cambia il valore dell’indice di felicità attraverso lo slider e clicca sul pulsante per confermare.

Il sistema elabora l’operazione che è stata effettuata e mostra solamente i distretti con l’indice di felicità richiesto dal cittadino.

## Requisiti

I requisiti sono divisi in 5 categorie

1. requisiti funzionali
2. requisiti ambientali e contestuali
3. requisiti dei dati
4. profili utente
5. requisiti di usabilità
6. Le funzionalità principali offerte dal sistema, consentono all’utente di muoversi sulla mappa corematica, visualizzare la legenda, espandere un distretto, modificare l’indice di felicità dei singoli fenomeni urbani in un distretto e filtrare i distretti.

Più nello specifico, i requisiti funzionali sono:

* zoom geografico
  + consente di effettuare l’operazione di zoom-in e zoom-out sulla mappa corematica
* zoom semantico
  + può essere di 2 tipi:
    - zoom-in: disgrega il corema cliccato mostrando più dettagli
    - zoom-out: aggrega il corema mostrando meno dettagli
* filtro semantico
  + filtra i distretti in base all’indice di felicità
* modifica indice di felicità dei fenomeni urbani
  + aggiorna l’indice di felicità del distretto, attraverso la modifica degli indici di felicità dei fenomeni urbani che caratterizzano l’area urbana di un distretto
* visualizzazione legenda
  + consente all’utente di visualizzare una legenda che fornisce la spiegazione dei coremi e fenomeni urbani

1. L’interazione con il sistema può avvenire sia all’aperto, in diverse aree cittadine in cui sono presenti dei totem touchscreen, sia al chiuso su tablet touchscreen.
2. Il sistema deve essere in grado di gestire ed elaborare dati provenienti da fonti esterne. Infatti i dati di distretti, area urbana e fenomeni urbani, provengono da un file esterno, di tipo geojson, il quale sarà elaborato dal sistema che salverà i valori in un database. Successivamente alcuni di questi dati, come ad esempio il valore dell’indice di felicità dei fenomeni urbani, potranno essere modificati dagli utenti e salvati nel database.
3. Il sistema è rivolto sia a semplici cittadini che potranno visualizzare e comprendere con facilità la qualità della vita nei distretti cittadini, che ad amministratori della città che potranno prendere delle decisioni per migliorare la qualità della vita nei distretti cittadini
4. Il sistema dovrà avere un’interfaccia minimale che sia più intuitiva possibile in modo che gli utenti debbano avere solo un’esperienza base con l’utilizzo di dispositivi touchscreen

## Scelte progettuali

### 3.3.1 Tecnologie

Per sviluppare il sistema si è scelto di usare tecnologie innovative, performanti e multipiattaforma.

Angular

Il framework utilizzato è Angular, un framework open source per lo sviluppo di applicazioni web ed è stato progettato per fornire uno strumento facile e veloce per sviluppare applicazioni che girano su qualunque piattaforma. (citazione wikipedia)

E’ un framework basato su componenti per la creazione di applicazioni web scalabili ed ha un una raccolta di librerie integrate che coprono un’ampia varietà di funzionalità tra cui routing, gestione dei forms e comunicazione client-server.

(citazioni angular.io)

Node Package Manager

Npm, abbreviazione di Node Package Manager, è un gestore di pacchetti per il linguaggio di programmazione Javascript che permette di organizzare, installare, riutilizzare e condividere del codice. Consiste in un client da linea di comando, chiamato npm, e un database online di pacchetti pubblici e privati, chiamato npm registry. Il registry è accessibile via client e i pacchetti disponibili sono consultabili sul sito web di npm. Il gestore di pacchetti e il registry sono gestiti da npm, Inc. (citazione wikipedia)

Attraverso l’uso del comando npm potremo cercare, utilizzare o pubblicare pacchetti che sono mantenuti all’interno del registry.

Angular – cli

Angular mette a disposizione Angular CLI, un’interfaccia a riga di comando che viene utilizzato per inizializzare, sviluppare, strutturare e mantenere le applicazioni Angular direttamente da una shell di comando. (citazione angular.io/cli)

(inserire lista di comandi più utilizzati?)

TypeScript

TypeScript è un linguaggio di programmazione open source sviluppato da Microsoft. Si tratta di un’estensione di JavaScript che basa le sue caratteristiche su ECMAScript 6.

Il linguaggio estende la sintassi di JavaScript in modo che qualunque programma scritto in JavaScript sia in grado di funzionare con TypeScript senza nessuna modifica. E’ stato progettato per lo sviluppo di grandi applicazioni ed è destinato ad essere compilato in JavaScript per poter essere interpretato da qualunque web browser o app. Il suo punto di forza è la tipizzazione. (citazione wikipedia)

Bootstrap

Bootstrap è un framework frontend open source, per uno sviluppo web semplice e veloce. Include modelli di progettazione basati su HTML e CSS per tipografia, forms, pulsanti, tabelle, navigazione, modali, caroselli di immagini e molto altro. Inoltre offre anche la possibilità di creare facilmente design responsive. (citazione bootstrap 5 w3school)

Firebase

Firebase è una piattaforma serverless per lo sviluppo di applicazioni mobili e web. Open source ma supportata da Google, Firebase sfrutta l’infrastruttura di Google e il suo cloud per fornire una suite di strumenti, per scrivere, analizzare e mantenere applicazioni cross-platform. Firebase infatti offre funzionalità come analisi, database (usando strutture noSQL), messaggistica e segnalazioni di arresti anomali per la gestione di applicazioni web, IOS, e Android

(citazione <https://www.geekandjob.com/wiki/firebase>)

OpenLayers

OpenLayers è una libreria JavaScript open source per visualizzare mappa interattive nel browser web.

(citazione wikipedia)

### 3.3.2 Motivazioni

Le conoscenze di programmazione acquisite durante il corso di studi sarebbero state sufficienti per sviluppare la web-app. Tuttavia uno dei motivi per cui si è scelto di utilizzare le tecnologie sopra elencate è non solo perché sono più recenti, versatili e performanti, ma anche perché sono più utilizzate in ambito lavorativo.

Basti pensare che la prima versione di Angular 2+ sia stata rilasciata nell’anno 2016 e la versione utilizzata per lo sviluppo del sistema è Angular 13.3.0, rilasciata nell’anno 2021. Uno dei vantaggi dell’utilizzo di Angular è la sua struttura a componenti che consente di suddividere l’applicazione in più componenti, solitamente uno per ogni funzionalità, permettendo il riutilizzo degli stessi contenendo cosi la mole di codice da scrivere.

La versione di Bootstrap utilizzata è la 5.1.3 e la sua peculiarità è di creare un design responsive e accattivante, attraverso l’utilizzo di classi CSS proprie, in modo molto semplice e intuitivo.

Typescript è il linguaggio di programmazione usato per sviluppare applicazioni in Angular e la versione utilizzata è la 4.6.2. A differenza di JavaScript, il cui codice viene eseguito direttamente nel browser, è un linguaggio compilato. Il codice typescript viene prima tradotto in codice JavaScript attraverso un traspilatore (è un tipo di compilatore che traspila il codice sorgente di un linguaggio ad alto livello in un output leggibile da un linguaggio a basso livello) e poi eseguito nel browser. Essendo un linguaggio tipizzato, gli errori sono individuati a tempo di compilazione.

Firebase consente di integrare rapidamente e facilmente un database noSQL all’interno della web-app. La versione utilizzata è la 9.8.1.

Infine, OpenLayers consente di inserire all’interno della web-app mappe opensource. La versione utilizzata è la 5.3.0. E’ una libreria JavaScript che è stata scelta rispetto a un’altra libreria, Leaflet, per una documentazione più completa e per avere già tutte le funzionalità incluse senza bisogno di integrare plugin esterni.

### 3.3.3 Architettura e sviluppo

Diagramma delle classi

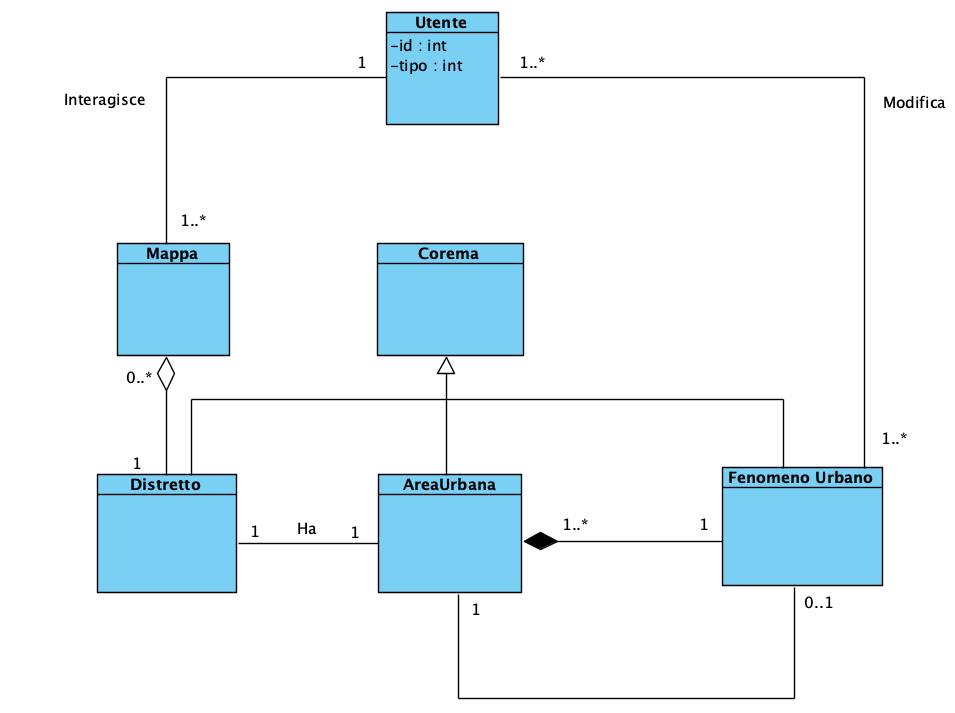
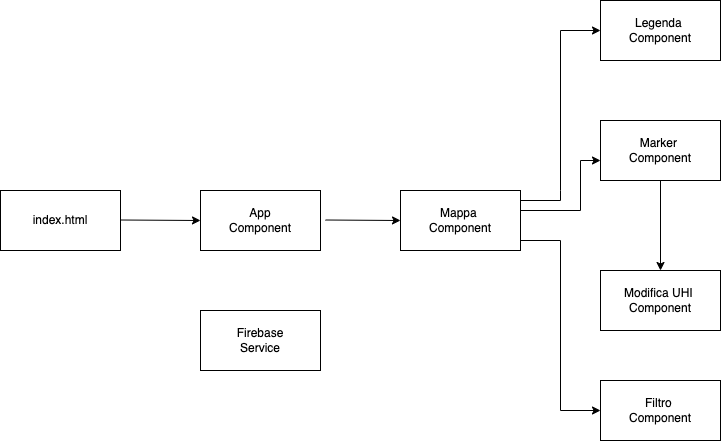


Diagramma architettura web-app

L’architettura della web-app, grazie ad Angular, è suddivisa in componenti. Un componente è composto da:

* Template HTML: definisce la view per l’app e può essere visto come l’UI dell’app
* Classe: è associata al template HTML, può essere vista come la logica dietro l’UI e contiene:
  + Proprietà e dati da passare al template per essere visualizzate dall’UI
  + Metodi che eseguono azioni per il template e definiscono la logica e le operazioni
* Metadati: forniscono ad Angular informazioni aggiuntive sul componente

La struttura della web-app è descritta dalla seguente figura:



Adesso si andrà a descrivere ogni elemento o componente che forma la struttura della web-app

Index.html

Come da figura, il file index.html rappresenta la view che sarà mostrata a display al caricamento dell’app.

Il file index.html contiene il componente App Component.

App Component

L’App Component a sua volta contiene il componente principale, Mappa Component.

Mappa Component

Tutto ciò che riguarda la mappa è stato creato grazie alla libreria JavaScript OpenLayers.

La mappa è il componente principale di OpenLayers e per eseguire il rendering della mappa sono necessari:

* Una view
* Uno o più layer
* Un target di destinazione

La mappa utilizzata è OpenStreetMap, un’insieme di mappe open source gratuite, mentre la proiezione utilizzata per creare la mappa è EPSG:4326.

La proiezione è un passaggio necessario nella creazione di una mappa bidimensionale ed è uno degli elementi essenziali della cartografia

In cartografia il termine “proiezione di una mappa” è il termine usato per descrivere un ampio insieme di trasformazioni impiegate per rappresentate la superficie curva bidimensionale di un globo su un piano. Nella proiezione di una mappa, le coordinate, spesso espresse come latitudine e longitudine, delle posizioni dalla superficie del globo, vengono trasformate in coordinate su un piano.

(cit wiki <https://en.wikipedia.org/wiki/Map_projection>)

Invece durante la creazione del layer per la mappa, si è specificato la sorgente da cui provengono i dati, un file di tipo geojson, e il formato dei dati, geojson.

Geojson è un formato aperto, utilizzato per archiviare una collezione di geometrie spaziali i cui attributi sono descritti attraverso il formato JSON. Le geometrie possibili sono punti, linee spezzate, poligoni e collezioni multiple di queste tipologie.

(citazione wiki <https://it.wikipedia.org/wiki/GeoJSON>)

Il componente Mappa viene eseguito al caricamento della web app e fa essenzialmente 2 cose:

* Crea la mappa
* Crea un layer che sarà aggiunto sulla mappa

Nella creazione della mappa viene usato un layer di base per caricare le mappe con la sorgente che fa riferimento, come detto in precedenza, a OpenStreetMap.

Successivamente viene aggiunto un layer che ha come sorgente di dati un file geojson locale che ha tutte le informazioni necessarie per disegnare i distretti come punti sulla mappa in base a latitudine e longitudine. Durante la lettura del file geojson, viene costruita un’associazione tra distretti, area urbana, fenomeni urbani creando gli oggetti delle classi corrispondenti e vengono elaborati i valori associati a questi elementi per calcolare l’indice di felicità. Tutti questi valori vengono salvati, poi, sul database.

Al caricamento della web app, in base al valore dell’indice di felicità, viene associato il colore corrispondente per distretti e fenomeni urbani

Il componente Mappa appena descritto, è il padre di tutti gli altri componenti che si andranno a descrivere.

Infatti all’interno del componente Mappa ci sono i seguenti componenti:

* Marker
* Modifica UHI
* Filtro
* Legenda

Marker Component

Il componente Marker viene caricato quando l’utente clicca su un distretto. Crea, attorno al distretto cliccato, un’area urbana con all’interno i coremi dei fenomeni urbani con il colore associato in base al valore dell’indice di felicità.

Modifica UHI Component

Il componente Modifica UHI è associato al pulsante menù. Se è stato cliccato un distretto, caricando il componente Marker, il click sul tasto menù mostra l’indice di felicità, per ogni fenomeno urbano associato al distretto cliccato che può essere modificato tramite uno slider e un pulsante che consente il salvataggio delle modifiche effettuate. Se viene cliccato il pulsante di salvataggio, i valori modificati vengono aggiornati nel database e vengono ricalcolati i colori dei fenomeni urbani e del distretto associato.

Filtro component

Il componente filtro è associato al pulsante menù. Se viene cliccato il tasto menù e non è stato cliccato su nessun distretto, viene caricato il componente Filtro che mostra uno slider che consente di filtrare i distretti, presenti sulla mappa, in base all’indice di felicità

Legenda Component

Il componente Legenda è associato al pulsante info e viene scatenato quando il pulsante info viene cliccato. Mostra tutte le informazioni utili alla comprensioni dei coremi recuperando i dati utili nelle classi associate ai coremi.

Service

Oltre ai componenti, Angular fornisce i servizi. A differenza di un componente, un servizio è una classe particolare usato per rappresentare dati che non sono associati ad alcuna view e che deve essere condiviso tra componenti. Per far si che un servizio sia visto da tutti i componenti deve essere registrato con Angular che crea una singola istanza della classe del servizio, detto singleton. Tramite il design pattern Dependency Injection, l’injector crea e gestisce il singleton di ogni servizio registrato e se un componente necessita di un servizio, la classe del componente definisce il servizio come una dipendenza, solitamente, all’interno del costruttore del componente; l’injector inietta l’istanza della classe del servizio quando viene istanziata la classe del componente.

Firebase Service

Il servizio Firebase è utilizzato dai componenti che si interfacciano con il database. Questo servizio contiene metodi per salvare e aggiornare i dati dei distretti nel database e restituire i dati dei distretti contenuti nel database

Feature Handler Service

Il servizio Feature Handler, contiene metodi che elaborano il file geojson, creando gli oggetti relativi ai valori contenuti nel file geojson e richiamando i metodi del servizio Firebase per salvare e aggiornare i dati.

Diagramma di Gantt