



UNIVERSITA' DEL SALENTO
Corso di Laurea in Ingegneria dell'Informazione

*Tesi di Laurea in Principi di Progettazione del
Software*

Progettazione e Codifica con Angular di una GUI

Designing and Coding with Angular of a GUI

RELATORE:

Ch.mo Prof Luca Mainetti

STUDENTE: *Alessio Carachino*

Matricola n°20034650

Sommario

<i>Introduzione</i>	6
Il progetto	6
Cos'è SafeCity	6
<i>Analisi Funzionale</i>	7
Requisiti della GUI.....	7
Studio dei casi d'uso.....	8
Casi d'uso individuati	9
Casi d'uso – Dashboard	9
Casi d'uso – Map	10
Casi d'uso – Search.....	11
Altri casi d'uso.....	12
<i>Progettazione</i>	13
Design Pattern – Model View Controller	13
Design Pattern – Dependency Injection.....	13
Design Pattern – Singleton	14
Design Pattern – Observer.....	14
Programming Paradigm – Reactive Programming.....	14
Diagramma delle sequenze	15
Diagramma delle sequenze – Notifica	15
Diagramma delle sequenze – Dashboard.....	16
Diagramma delle sequenze – Map.....	17
Diagramma delle sequenze – Search	18
Albero delle dipendenze	19
<i>Codifica</i>	20
Perchè Angular.....	20
Typescript	20
NPM.....	21
Angular Material	21
OpenLayers	21
IntelliJ IDEA.....	22
Tabella delle funzionalità e relativo peso	23
Sezione Dashboard.....	24
Sezione Map	26

Sezione Search	27
Schermata di esempio – Notifica	29
Test dell'applicazione.....	30
<i>Criticità</i>	31
<i>Glossario</i>	33
<i>Elenco delle figure</i>	34
<i>Bibliografia e sitografia.....</i>	35

Introduzione

Il progetto

L’obiettivo della tesi è la realizzazione di una GUI (Graphic User Interface), ossia la produzione dell’interfaccia grafica di un’applicazione web. In questo caso, l’applicazione è già esistente ed ha un nome: SafeCity.

La GUI sarà utilizzata da un utente Admin del servizio, per consultare e salvare nella maniera più agile possibile statistiche e informazioni legate agli eventi segnalati dagli utenti che utilizzano SafeCity.

Cos’è SafeCity

Il progetto “SafeCity” è stato sviluppato con lo scopo di creare un software che aiuti chi governa un paese a gestire situazioni di pericolo e/o disagio che i cittadini si trovano ad affrontare tutti i giorni.

Nello specifico attraverso un’applicazione Mobile un cittadino può segnalare un evento scegliendo tra quelli predefiniti dal sistema (ed eventualmente aggiunti e modificati da un Responsabile). Tali segnalazioni sono gestite da una serie di Responsabili che, attraverso un’interfaccia Web possono non solo inoltrare degli agenti per risolvere la segnalazione stessa, ma anche prendere visione di alcuni dati relativi all’inquinamento atmosferico raccolti da dei dispositivi IOT posti in punti strategici della città.

Le boards sono anche in grado di inviare una segnalazione in maniera autonoma nel momento in cui i valori dei GAS (quali ad esempio monossido di carbonio, idrogeno ecc...) superano un valore limite ritenuto pericoloso per la salute dell’uomo. Gli agenti precedentemente citati utilizzano un’applicazione Mobile per ricevere gli incarichi accettandoli o meno secondo le loro esigenze.

Analisi Funzionale

Requisiti della GUI

Contrariamente a quanto si potrebbe pensare, per la realizzazione di una GUI non è indispensabile conoscere tutte le specifiche del progetto in cui l'interfaccia dovrà poi essere integrata.

È però necessario distinguere i requisiti che questa deve soddisfare:

Requisiti obbligatori:

- L'Admin deve poter visualizzare un grafico rappresentante il numero delle segnalazioni giornaliere dell'ultima settimana;
- L'Admin deve poter visualizzare un grafico rappresentativo dell'andamento delle segnalazioni anonime rispetto a quello delle segnalazioni di utenti non anonimi.
- L'Admin deve poter visualizzare un grafico rappresentativo dei giorni settimanali più attivi in numero di segnalazioni.
- L'Admin deve poter visualizzare direttamente su una mappa la posizione delle segnalazioni e filtrarle in base a *gravità* e *tipo di segnalazione*.
- L'Admin deve poter visualizzare le segnalazioni, in base a *zona* e *range di date*.
- L'Admin deve poter essere avvertito nel momento in cui arriva una nuova segnalazione, in modo che possa aggiornare la pagina e visualizzare i dati rinnovati.
- La GUI deve essere facilmente implementabile all'interno delle piattaforme mobile.

Requisiti desiderabili:

- L'Admin deve poter visualizzare il numero di segnalazioni massime raggiunte in un giorno e la data corrispondente.
- L'Admin deve poter visualizzare il numero di segnalazioni totali e un'informazione che espliciti quando è stata ricevuta l'ultima segnalazione.
- L'Admin deve poter salvare i dati filtrati della tabella in un file PDF.
- L'Admin deve poter cercare per ID una segnalazione.

Requisiti facoltativi:

- L'Admin deve poter visualizzare la posizione corrente sulla propria mappa, in modo da avere una visione migliore sulle segnalazioni nelle sue vicinanze.

Studio dei casi d'uso

L'analisi dei casi d'uso, nell'ambito dell'ingegneria del software, è una tecnica che ha come obiettivo non solo quello di modellare le interazioni tra gli utenti, fisici e non, che utilizzeranno il sistema che si vuole realizzare ma è anche fondamentale per esplicitare le azioni che questi potranno compiere senza ambiguità. Un caso d'uso non è altro che un insieme di azioni che, se compiuti nell'ordine descritto, porteranno al raggiungimento di uno scopo voluto.

Infine, tutti i casi d'uso vengono rappresentati all'interno di un diagramma, detto Diagramma dei Casi d'Uso che permette di avere a colpo d'occhio una panoramica generale sulle potenzialità del sistema. Nel diagramma menzionato, gli attori sono associati con i casi d'uso, e questo indica il fatto che essi partecipano allo specifico caso d'uso, mentre i casi d'uso sono messi in relazione tra di loro mediante due differenti tipologie di relazione: *inclusione*, se quello indicato dalla punta della freccia è incluso all'interno di quello alla base e *estensione*, se quello indicato dalla punta della freccia è esteso da quello presente alla base della stessa.

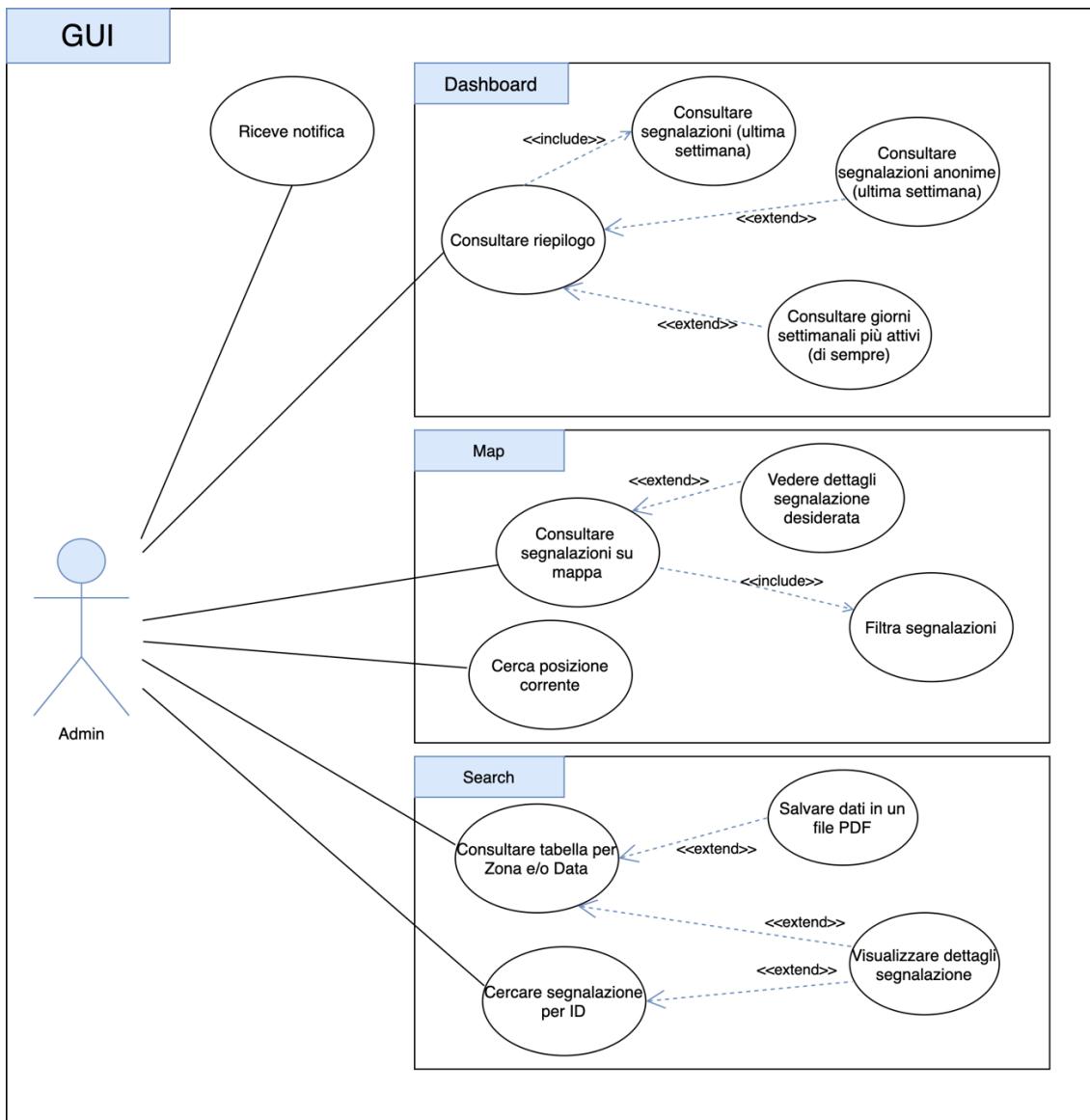


Figura 1 - Diagramma dei casi d'uso

Casi d'uso individuati

#	Nome del caso d'uso
1	Consultare riepilogo
2	Consultare segnalazioni (ultima settimana)
3	Consultare segnalazioni anonime (ultima settimana)
4	Consultare giorni settimanali più attivi (di sempre)
5	Consultare segnalazioni su mappa
6	Filtrare segnalazioni
7	Vedere dettagli segnalazione desiderata
8	Cerca posizione corrente
9	Consultare tabella per Zona e/o Data
10	Salvare dati in un file PDF
11	Cercare segnalazione per ID
12	Visualizzare dettagli spedizione
13	Riceve notifica

Casi d'uso – Dashboard

Nome: **Consultare riepilogo**

Precondizione: Admin effettua il Login

Descrizione: Consente all'Admin di consultare informazioni di base come, ad esempio, il numero totale di segnalazioni effettuate, il massimo numero di segnalazioni giornaliere mai registrate, ecc.

Flusso base:

1. Admin accede alla Dashboard
2. Admin attende che il Sistema riceva ed elabori le segnalazioni registrate
3. Admin **Consulta le segnalazioni (ultima settimana)**

Descrizione: Consente all'Admin di consultare su un grafico l'andamento del numero delle segnalazioni della settimana corrente

Flusso alternativo 1: Consultare segnalazioni anonime (ultima settimana)

Descrizione: Consente all'Admin di consultare su un grafico l'andamento delle segnalazioni non anonime e di quelle inviate da utenti non anonimi.

4. Admin contrae il pannello espandibile che non vuole più visualizzare
5. Admin scorre la pagina e trova il pannello espandibile relativo alle informazioni sugli utenti
6. Admin espande il pannello relativo alle informazioni sugli utenti

Flusso alternativo 2: Consultare giorni settimanali più attivi (di sempre)

Descrizione: Consente all'Admin di consultare, per ogni giorno settimanale, la media del numero di segnalazioni registrate nel corrispettivo giorno della settimana.

4. Admin contrae il pannello espandibile che non vuole più visualizzare
5. Admin scorre la pagina e trova il pannello espandibile relativo alle segnalazioni durante la settimana
6. Admin espande il pannello relativo alle segnalazioni durante la settimana

Casi d'uso – Map

Nome: Consulta segnalazioni su mappa

Precondizione: I dati relativi alle segnalazioni vengono elaborati

Descrizione: Consente all'Admin di vedere su una mappa OpenLayers la posizione delle segnalazioni effettuate dagli utenti

Flusso base:

1. Admin accede alla sezione Map
2. Admin imposta il livello di zoom ottimale per la ricerca
3. Admin **Filtrà segnalazioni**

Descrizione: consente all'Admin di filtrare i risultati di ricerca per Gravità e Tipo di Segnalazione/Categoria

4. Admin seleziona valori di Gravità che desidera ('leggera', 'media', 'grave', 'tutte'). È possibile selezionare più di un valore mentre il valore predefinito è 'tutte'
5. Admin seleziona i Tipi di Segnalazione/Categorie che desidera. I valori possibili tra cui può scegliere dipenderanno dai dati ricevuti dal sistema; in particolare tutte le categorie saranno selezionabili. È possibile selezionare più di un valore mentre il valore predefinito è 'tutte'
6. Admin clicca sul bottone di conferma se ha impostato valori diversi da 'tutte' sia nel campo Gravità sia nel campo Tipi di Segnalazione/Categorie

Flusso alternativo 1: Vedere dettagli segnalazione desiderata

Descrizione: Admin consulta i dettagli (data della segnalazione, titolo, ecc...) relativi ad una segnalazione scelta

7. Admin cerca sulla mappa la segnalazione di cui vuole vedere i dettagli
8. Admin clicca sul marker relativo alla segnalazione desiderata
9. Admin espande il popup contenente i dettagli della segnalazione

Nome: **Cerca posizione corrente**

Precondizione: I dati relativi alle segnalazioni vengono elaborati

Descrizione: Consente all'Admin di vedere la sua posizione su una mappa OpenLayers. In questo modo, potrebbe vedere, ad esempio, le segnalazioni effettuate dagli utenti nelle sue vicinanze

Flusso base:

1. Admin accede alla sezione Map
2. Admin clicca il bottone relativo a 'Cerca la mia posizione'
3. Admin consente al browser di accedere al servizio di geolocalizzazione

Casi d'uso – Search

Nome: **Consultare tabella per Zona e/o Data segnalazioni su mappa**

Precondizione: I dati relativi alle segnalazioni vengono elaborati

Descrizione: Consente all'Admin di visualizzare le segnalazioni effettuate in specifiche zone e/o in un preciso range di date

Flusso base:

1. Admin accede alla sezione Search
2. Admin imposta il filtro che desidera applicare alla ricerca
3. Admin clicca sul bottone di conferma

Flusso alternativo 1:

4. Admin sceglie di **Salvare i risultati in un file PDF**, quindi clicca sul bottone apposito

Descrizione: Consente all'Admin di salvare le segnalazioni visualizzabili nella tabella in un file di formato PDF

Flusso alternativo 2:

4. Admin sceglie un risultato presente nella tabella
5. Admin clicca sul risultato scelto
6. Admin **Visualizza dettagli della segnalazione**

Descrizione: Consente all'Admin di avere accesso ai dettagli di una particolare segnalazione tramite l'interazione con un Dialog

Nome: Cercare segnalazione per ID

Precondizione: I dati relativi alle segnalazioni vengono elaborati

Descrizione: Consente all'Admin di cercare una particolare segnalazione per ID, ed eventualmente visualizzarne i dettagli

Flusso base:

1. Admin accede alla sezione Search
2. Admin imposta la ricerca per ID
3. Admin scrive l'ID della segnalazione da cercare

Flusso alternativo 1:

4. La ricerca per ID non ha prodotto il risultato, quindi, verrà visualizzato un Dialog di avvertenza

Flusso alternativo 2:

4. La ricerca per ID ha prodotto il risultato quindi l'Admin **Visualizza dettagli della segnalazione**

Altri casi d'uso

Nome: Riceve notifica

Iniziatore: Sistema rileva che i dati sulle segnalazioni della sessione corrente non sono aggiornati

Flusso base:

1. Admin nota la presenza del badge di notifica sopra l'icona a forma di campana
2. Admin clicca sull'icona a forma di campana per visualizzare i dettagli sulla notifica

Flusso alternativo 1:

3. Admin ignora la notifica, quindi non visualizzerà i dati aggiornati

Flusso alternativo 2:

3. Admin clicca sul contenuto della notifica
4. Il Sistema ricarica la pagina
5. Admin viene reindirizzato alla sezione Dashboard

Progettazione

Design Pattern – Model View Controller

Il modello di progettazione Model – View – Controller specifica che un'applicazione consiste in un modello di dati, una presentazione delle informazioni ed un controllo dei dati. Il pattern richiede che queste tre componenti siano separate ed encapsulate in differenti oggetti.

Nello specifico:

- il Model contiene solo i dati e fornisce i metodi per accedere agli stessi: per questo progetto non ci si è occupati della realizzazione del Model perché già esiste per SafeCity;
- La View presenta i dati all'utente, quindi, conosce come accedere ai dati e si occupa dell'interazione tra il Sistema e l'Utente;
- Il Controller rappresenta il punto di congiunzione tra il Model e la View.

Quindi, viene a crearsi un sistema ciclico di quattro entità dove il Controller riceve i comandi dell'utente che a sua volta manipola i dati del Model in modo che esso aggiorni la View che a sua volta sarà vista dall'Utente. Uno schema rappresentativo è quello in figura 6:

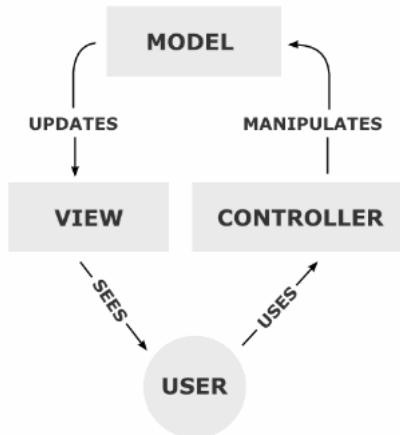


Figura 2 - Schema MVC

Design Pattern – Dependency Injection

Le dipendenze sono Services o Objects di cui una Classe ha bisogno per svolgere la sua funzione. La dependency injection è un design pattern per cui una Classe richiede le dipendenze da fonti esterne, piuttosto che crearle da s.

Il client, quindi, delega a un componente separato (definito come injector) la responsabilità di approvvigionare le dipendenze di cui esso necessita; il client non chiama in modo esplicito l'injector, bensì è l'injector che si occupa della costruzione degli oggetti.

La Dependency Injection di Angular fornisce le dipendenze alla classe nella fase di inizializzazione della stessa cioè, in pratica, il Service che ha le dipendenze di cui la classe ha bisogno

viene iniettato all'interno del costruttore della classe Typescript. Ciò permette di aumentare la flessibilità e la modularità dell'applicazione.

Design Pattern – Singleton

Il Design Pattern Singleton consente ad una Classe o ad un Service dichiarato Singleton di essere istanziato un'unica volta, cioè non possono essere presenti multiple istanze della Classe o del Service in esecuzione all'interno dell'applicazione.

Poiché i Services di Angular hanno come obiettivo principale quello di permettere la condivisione di dati tra i componenti dell'applicazione, fornire ad essi gli stessi dati è di vitale importanza per evitare bug causati dall'inconsistenza degli stessi; ciò è quindi facilitato dalla natura del Singleton: poiché i Services Singleton non possono essere istanziati molteplici volte, iniettare nell'applicazione o in alcune Classi il Service vuol dire fornire ai componenti gli stessi dati; tuttavia questo DI non è esente da svantaggi: usare un Service Singleton significa infatti che l'applicazione potrebbe occupare più memoria per poter utilizzare tutti i dati di cui ha bisogno.

Design Pattern – Observer

L'observer è il design pattern utilizzato tramite l'utilizzo di "Observable" in Angular. Gli oggetti Observable sono oggetti che desideriamo osservare, captando ogni loro variazione e agendo sugli stessi.

Gli oggetti osservabili vengono registrati e altri oggetti osservano, usando il metodo di iscrizione. L'utilizzo di questi oggetti è molto consigliato per la gestione degli eventi attraverso un approccio asincrono che è l'ideale quando l'applicazione ha ad esempio bisogno di usare dei dati provenienti da una fonte esterna.

Un altro esempio di uso più comune nel progetto realizzato è quello rappresentato dai click su alcuni buttoni che saranno osservati dal corrispettivo observer che scatenerà azioni o metodi.

Programming Paradigm – Reactive Programming

Il paradigma Reactive agevola l'utilizzo di oggetti le cui proprietà sono dipendenti da altri oggetti che a loro volta cambieranno per via di stream di dati e che dovranno in qualche modo aggiornare in cascata, in maniera asincrona, gli oggetti che dipendono da quelle specifiche proprietà. Questo paradigma è quindi nato appositamente per semplificare la creazione di user interfaces interattive.

Un'applicazione Angular è di per sé un sistema reactive. Quando un utente clicca un bottone l'applicazione reagisce a questo evento e aggiorna il modello. All'aggiornamento del modello l'applicazione propaga i cambiamenti attraverso l'albero dei componenti.

Diagramma delle sequenze

Al fine di poter esplicitare l'ordine delle interazioni tra gli elementi del sistema occorre che i vari comportamenti siano modellati separatamente, ad esempio col diagramma delle sequenze, che è una tipologia di diagramma di interazione, descritta dall'UML.

In questo tipo di diagramma non è tanto importante quantificare il tempo che intercorre tra un evento ed un altro quanto descrivere attraverso una notazione precisa la loro successione esatta. Come specificato dallo standard dell'OMG, tra i costituenti base della notazione si ritrovano:

- le linee di vita: rappresentano la cronologia di un processo. La sua testa consta di un rettangolo che in genere include il nome dell'oggetto e il nome della classe. Le linee di vita sono stereotipate in tre modi: entità, confine e elemento di controllo;
- le linee tratteggiate: sono quelle che scendono dalle teste e rappresentano linearmente muovendosi verso il basso, il passare del tempo.

Inoltre, per rappresentare tutte le diverse tipologie di messaggi, si utilizzeranno:

- messaggi asincroni con una freccia con linea continua e con punta vuota;
- messaggi sincroni con una freccia con linea continua e punta piena;
- messaggi trovati con un cerchio pieno concatenato ad una freccia continua e punta piena;
- messaggi di risposta con una freccia con linea tratteggiata e punta piena.

Diagramma delle sequenze – Notifica

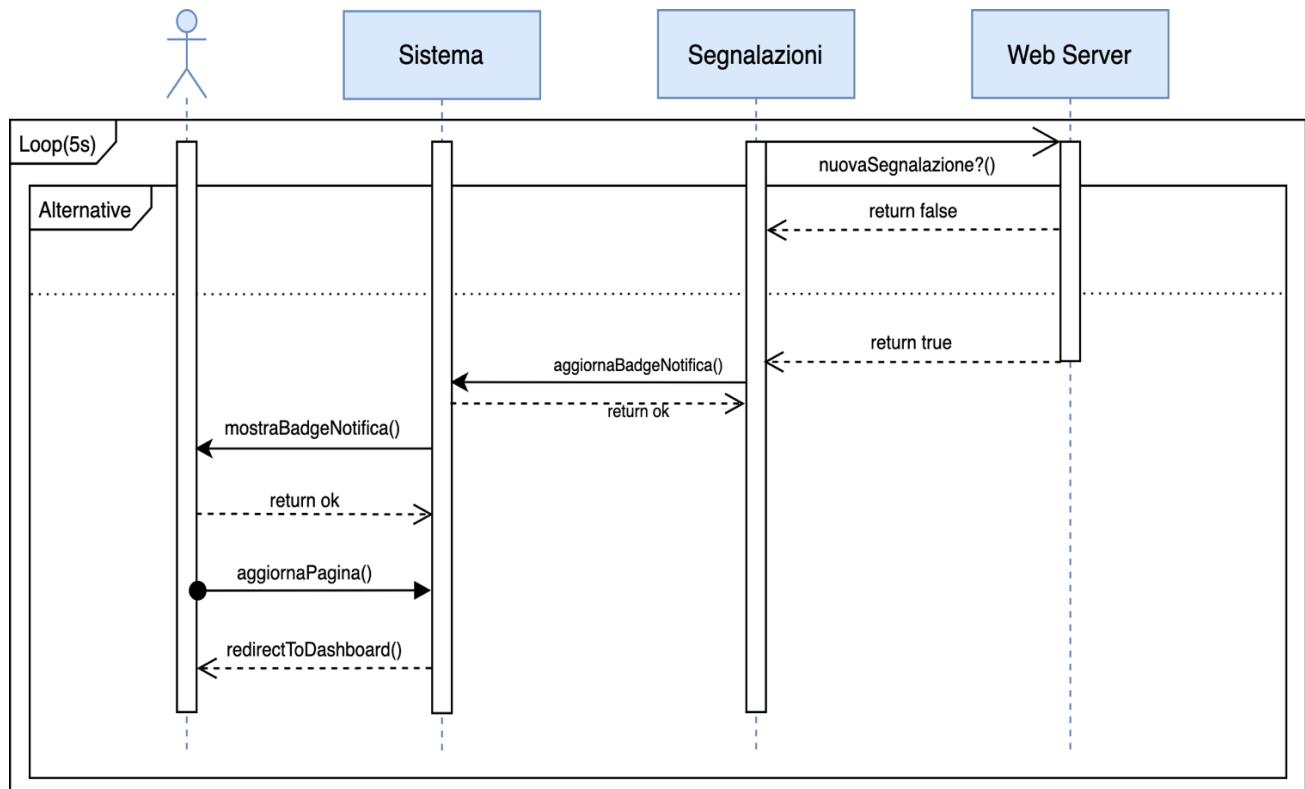


Figura 3 - Diagramma delle sequenze - Notifica

Diagramma delle sequenze – Dashboard

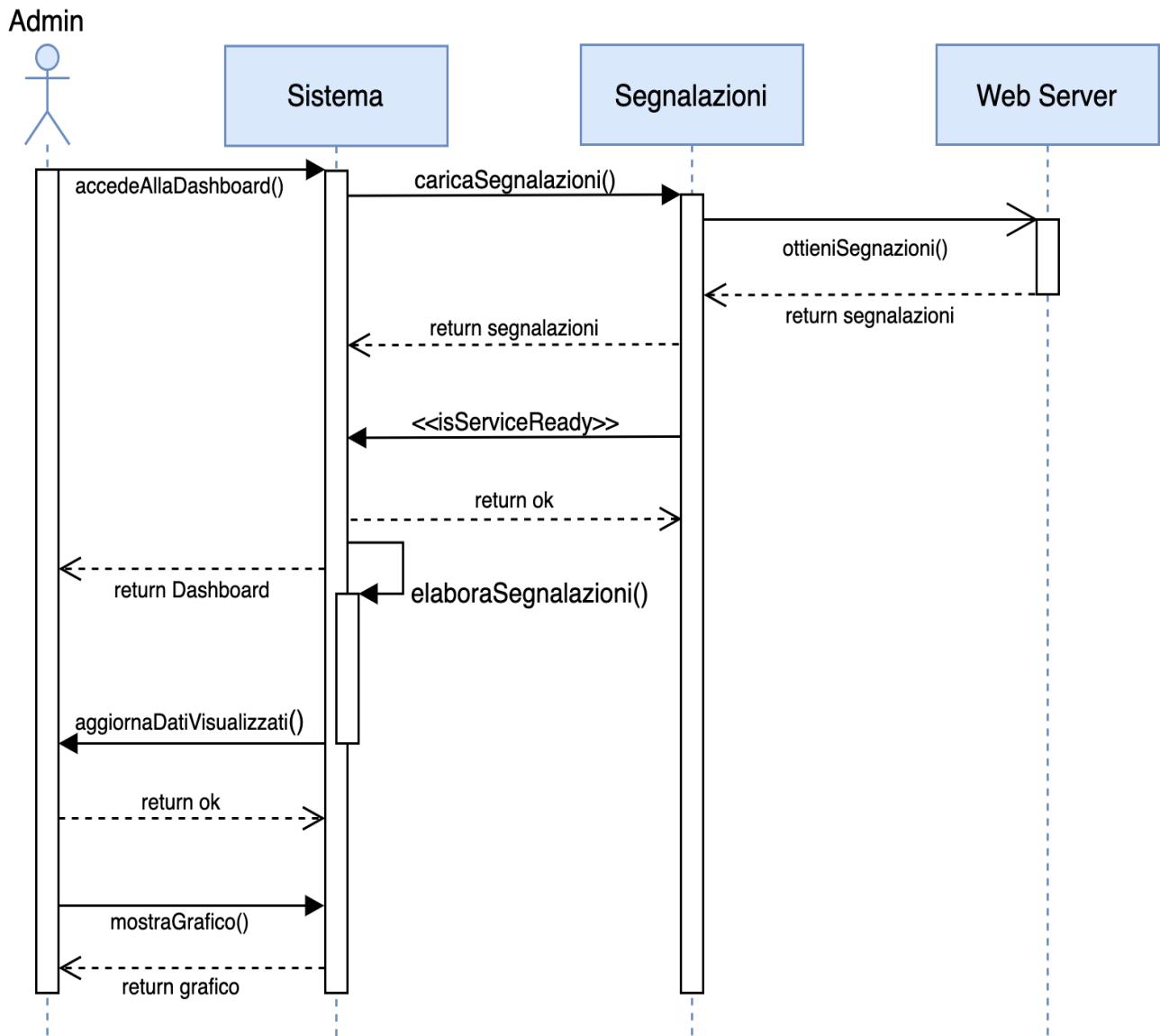


Figura 4 - Diagramma delle sequenze - Dashboard

Diagramma delle sequenze – Map

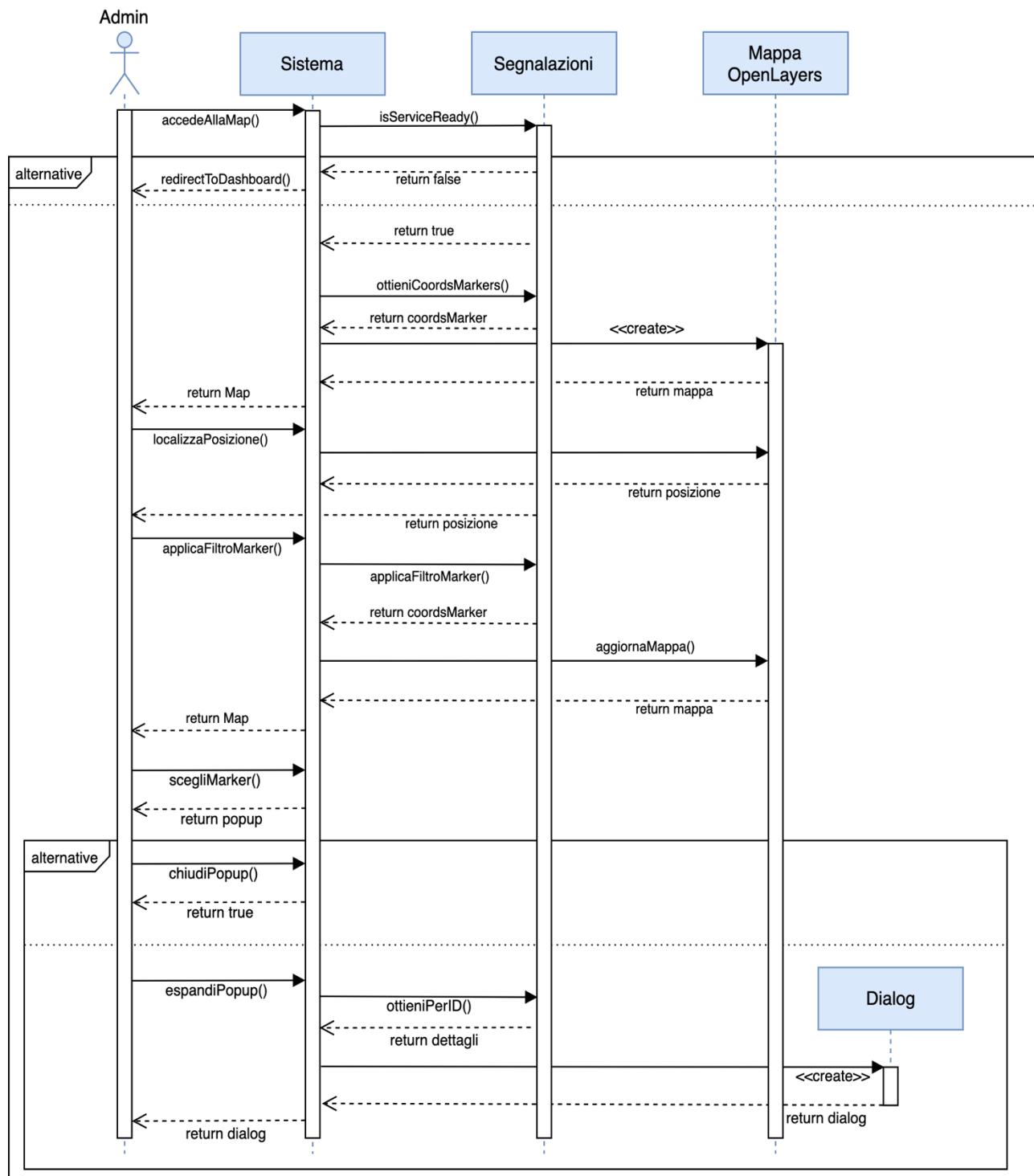


Figura 5 - Diagramma delle sequenze - Map

Diagramma delle sequenze – Search

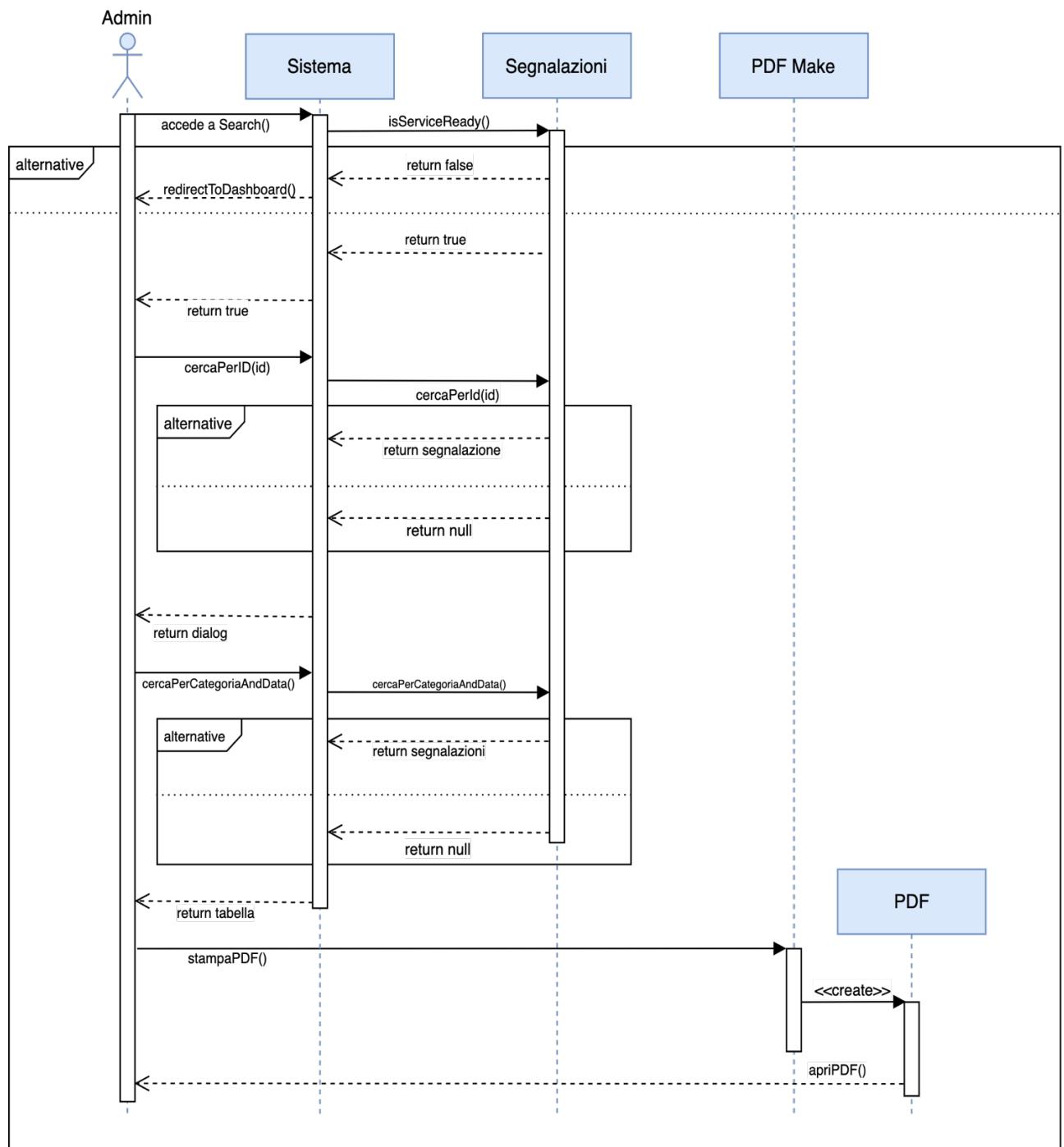


Figura 6 - Diagramma delle sequenze - Search

Albero delle dipendenze

Affinché vengano soddisfatti i requisiti di qualità che deve avere un buon software, individuati dall'ingegneria del software, occorre che questo venga strutturato già in fase di progettazione. In questo caso, è stato realizzato l'albero delle dipendenze, che consente di avere una panoramica strutturale delle dipendenze tra i vari componenti, moduli e servizi implementati nel progetto Angular.

In particolare, se l'albero viene costruito in maniera ottimale, il progetto che sfrutta tale struttura potrà godere di vantaggi come:

- la modularità, quindi viene facilitato il procedimento di modifica, eliminazione, sostituzione e implementazione di moduli;
- la riusabilità, infatti più le varie funzionalità sono incapsulate e accorpate all'interno di specifici moduli e più queste possono essere riutilizzabili per altri progetti;
- chiarezza strutturale, per agevolare il lavoro alle persone che entrano a far parte del progetto anche a distanza di mesi dall'inizio della fase di codifica.

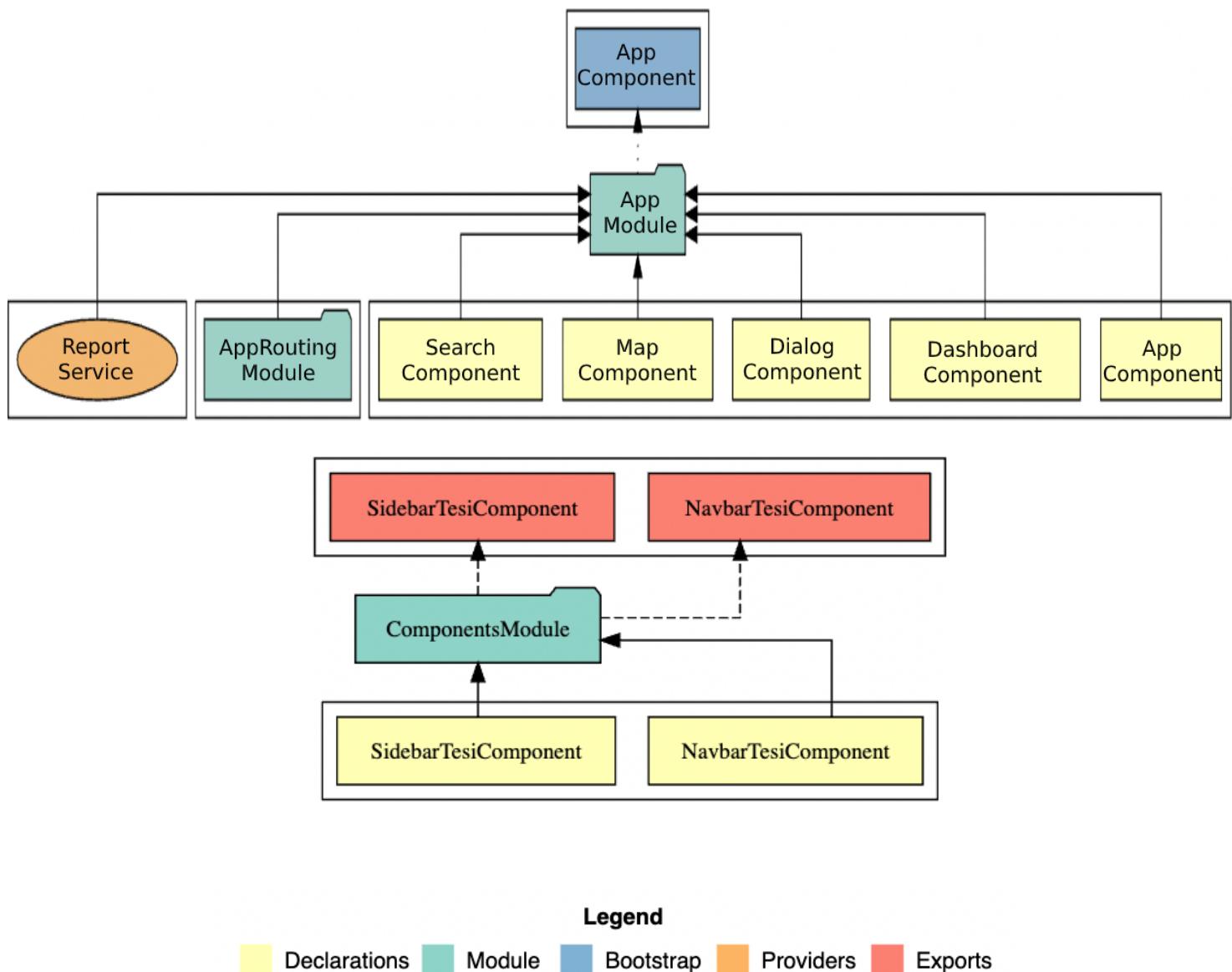


Figura 7 - Albero delle dipendenze

Codifica

Perchè Angular

Il software, che sia una web app o un'app mobile nativa/ibrida, necessita di requisiti precisi non solo in termini di funzionalità. Scegliere il framework più adatto è di fondamentale importanza per far sì che il software venga sviluppato nel modo migliore possibile, in termini di velocità e semplicità.

Un framework è in sostanza un'architettura logica di supporto sul quale un software può essere progettato e realizzato. Per la GUI di questo progetto sono state sfruttate le potenzialità di Angular, che è un framework molto utilizzato per la realizzazione di web app e app mobile native/ibride, ed è in continua crescita ormai praticamente da quando è stato reso disponibile per via della sua semplicità e versatilità.

Le caratteristiche principali di Angular fanno al caso della GUI dell'Admin per SafeCity, il cui frontend è anch'esso realizzato con Angular, perché:

- Permette di usare il linguaggio di programmazione Typescript;
- Quasi la totalità del codice scritto con Angular è riutilizzabile se si vuole portare la web app su altre piattaforme;
- Supporta il package manager Npm, che è una raccolta di pacchetti utilizzabili nel progetto: ciò si traduce in un minor tempo necessario per la realizzazione del progetto;
- È ideale per la realizzazione di una SPA (Single Page Application) grazie alle sue logiche di condivisione di dati, provenienti da un web server e non, tra componenti;
- Supporta nativamente gli eventi touch e gesture tipiche degli schermi dei dispositivi mobile, il che è ottimo in prospettiva di future implementazioni della GUI all'interno di app mobile.



Figura 8 - logo di Angular

Typescript

Per poter sfruttare appieno le potenzialità di Angular, i suoi sviluppatori consigliano come linguaggio di programmazione Typescript.

Typescript, infatti, è un linguaggio open source che “transpila” il codice scritto in linguaggio Javascript, quindi possiede le medesime qualità di quest’ultimo e ne aggiunge delle nuove, come ad esempio la tipizzazione statica degli oggetti, oltre che avere una curva d’apprendimento ridotta.

Angular in particolare, riesce a trarre il massimo beneficio da questo linguaggio, mantenuto da Microsoft, perché permette di usare delle API scritte in Javascript, fornite anche da terzi, rendendo il lavoro dello sviluppatore molto più agile.

NPM

NPM è un package manager cioè una raccolta di pacchetti gratuiti, o a pagamento, utilizzabili all'interno del progetto Angular. Utilizzabile solo dopo aver installato Node.js, NPM viene messo a disposizione come strumento da linea di comando con il quale vengono scaricati i moduli JavaScript che si vogliono integrare. In particolare, sono tre i moduli maggiormente sfruttati per questo progetto, ossia Angular Material, per fornire una veste grafica leggera e accattivante, OpenLayers, che è un modulo open source utilizzato per visualizzare su una mappa geografica interattiva informazioni di ogni genere e infine Chartist, il cui scopo è quello di permettere la realizzazione di diagrammi graficamente leggeri e ben animati in maniera semplice.

Tra i moduli secondari importati con NPM è presente Pdfmake che permette la generazione di file in formato PDF i cui dati sono contenuti all'interno di tabelle.



Figura 9 - logo di NPM

Angular Material

Tutti i componenti del progetto Angular realizzato hanno una forte integrazione di Angular Material che è una libreria di componenti per la User Interface ispirata dal Material Design di Google. Ciò consente alla web app di regalare all'end user un'esperienza gratificante a livello visivo: non si incontreranno mai lunghi testi ma è tutte le informazioni sono raccolte in Cards, Expansion Panels comprimibili e estendibili, Icons, Dialogs, Tabs, ecc...

Infine, l'insieme dei componenti sono perfettamente coerenti l'uno con l'altro, dato che Angular Material consente di applicare un tema ai propri componenti di tipo sia predefinito, cioè incluso nel package, sia personalizzato.

OpenLayers

OpenLayers è una libreria Javascript open source che permette l'integrazione di una mappa geografica interattiva all'interno del progetto. La mappa, in particolare, è utilizzabile con tutte le funzionalità disponibili gratuitamente, il che rende OpenLayers una valida alternativa alla concorrenza che in genere permette di accedere gratuitamente solo ad una parte delle funzionalità.

Integrabile nel progetto grazie ad NPM, la mappa visualizzabile nella web app è strutturata in più livelli chiave come ad esempio la View, che determina alcune proprietà che deve avere la mappa

renderizzata, come le coordinate del centro o la risoluzione, i Layers che sono gli elementi sovrapponibili alla mappa e le Features che rappresentano gli oggetti da renderizzare con differenti geometries (come punti, linee) o con immagini (come i markers).



Figura 10 - logo di OpenLayers

IntelliJ IDEA

IntelliJ IDEA è un Integrated Development Environment (IDE) della Suite offerta da JetBrains concepito per realizzare software Java ma supporta anche Angular, dalla creazione del progetto e dalla realizzazione dei componenti fino al debugging della stessa. Ciò significa che riesce a gestire intelligentemente anche file scritti in linguaggio HTML, CSS/SCSS, e Typescript.

Esso è uno degli IDE più potenti disponibili per gli sviluppatori, grazie alle sue molteplici caratteristiche, come la coding assistance che permette un raffinato auto-completamento, e refactoring delle linee di codice in maniera adeguata al contesto. Infine, il suo ecosistema di plug-in rende di fatto potenzialmente estendibile le funzionalità dell'IDE stesso.



Figura 11 - logo di IntelliJ

Tabella delle funzionalità e relativo peso

Al fine di pianificare meglio la fase di codifica, è opportuno stilare una tabella che specifichi tutte le funzionalità che devono essere implementate, facendo riferimento ai casi d'uso e ai requisiti emersi dalla fase di analisi; tale tabella deve essere facilmente leggibile infatti ha per ciascuna riga due colonne: la prima indica il nome della funzionalità, e la seconda ha all'interno un valore numerico espresso in percentuale per rappresentare approssimativamente il peso che ha la corrispettiva funzionalità all'interno della fase di codifica, soprattutto in termini di tempo necessario per lo sviluppo.

Nome della funzionalità	Peso (%)
Riepilogo informazioni sugli eventi segnalati	5
Grafico dell'andamento delle segnalazioni	10
Grafico dell'andamento delle segnalazioni anonime	10
Grafico indicativo dei giorni settimanali più attivi	10
Mappa geografica delle segnalazioni	15
Operazioni di filtraggio per la mappa	10
Visualizzare informazioni dettagliate della segnalazione scelta sulla mappa	5
Servizio di geolocalizzazione	2
Visualizzare tabella delle segnalazioni	5
Operazioni di filtraggio per la tabella	10
Salvare i risultati della tabella su un file in formato PDF	2
Cercare una segnalazione per ID	3
Visualizzare i dettagli dell'eventuale segnalazione trovata per ID	3
Sistema di notifica dell'admin in seguito alla rilevazione di una nuova segnalazione	3
Realizzazione della Route Guard per la gestione delle rotte accessibili in relazioni allo stato del servizio	5
Realizzazione della barra di navigazione laterale e barra della navigazione superiore	2

Sezione Dashboard

La Dashboard ha un ruolo cruciale all'interno del progetto: sarà infatti questo componente a segnalare al Report Service, che è quindi iniettato nel costruttore del componente, che dovrà richiedere al web server i dati sugli eventi segnalati.

Il Report Service, tramite l'Http Client Service, quindi, inoltra una chiamata GET al web server. Quando avrà ricevuto le informazioni, il Report Service segnalerà alla Dashboard che i dati sono stati ricevuti; in questo modo verranno elaborate le informazioni specifiche richieste per quest'ultima.

Nella figura seguente, è rappresentato uno screenshot della prima schermata a cui l'admin ha accesso quando i dati saranno ricevuti dalla Dashboard.

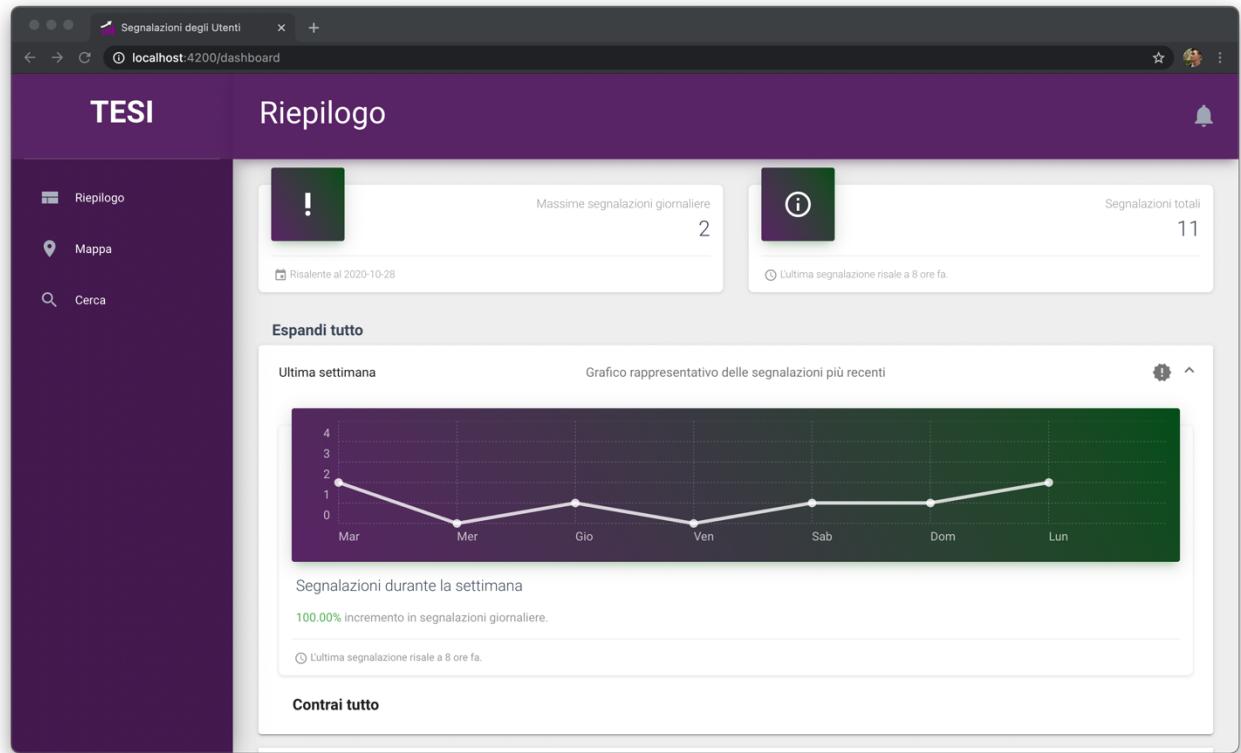


Figura 12 - Schermata Dashboard 1

Come si era prefissato nel capito dell'Analisi Funzionale, inizialmente, il pannello relativo al grafico rappresentativo delle segnalazioni più recenti sarà esteso.

Tramite il bottone “Contri tutto” o “Espandi tutto”, l'admin avrà modo di controllare i pannelli espandibili; dopo aver cliccato su “Contri tutto”, la schermata che vedrà sarà nella pagina seguente.

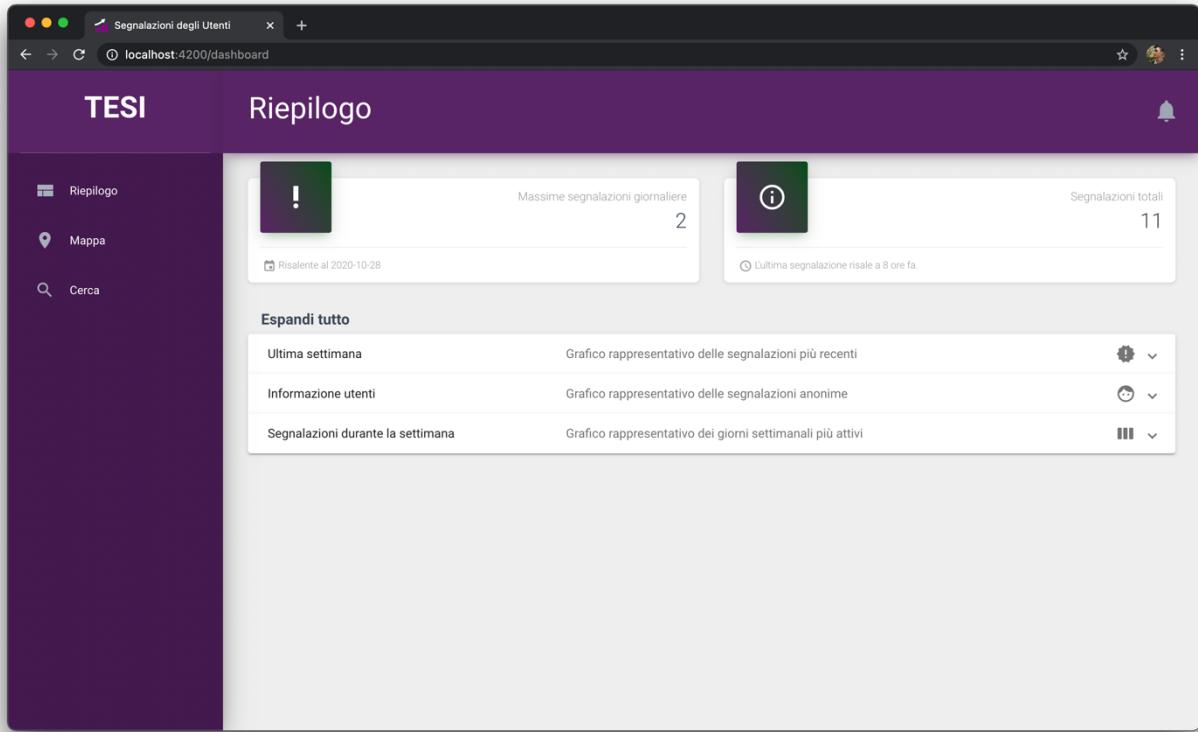


Figura 13 - Schermata Dashboard 2

In questo modo, sarà agevole per l'Admin scegliere il pannello che vuole visualizzare. Sempre a scopo dimostrativo, basterà che clicchi sul pannello desiderato ed esso si espanderà come in Figura 13.

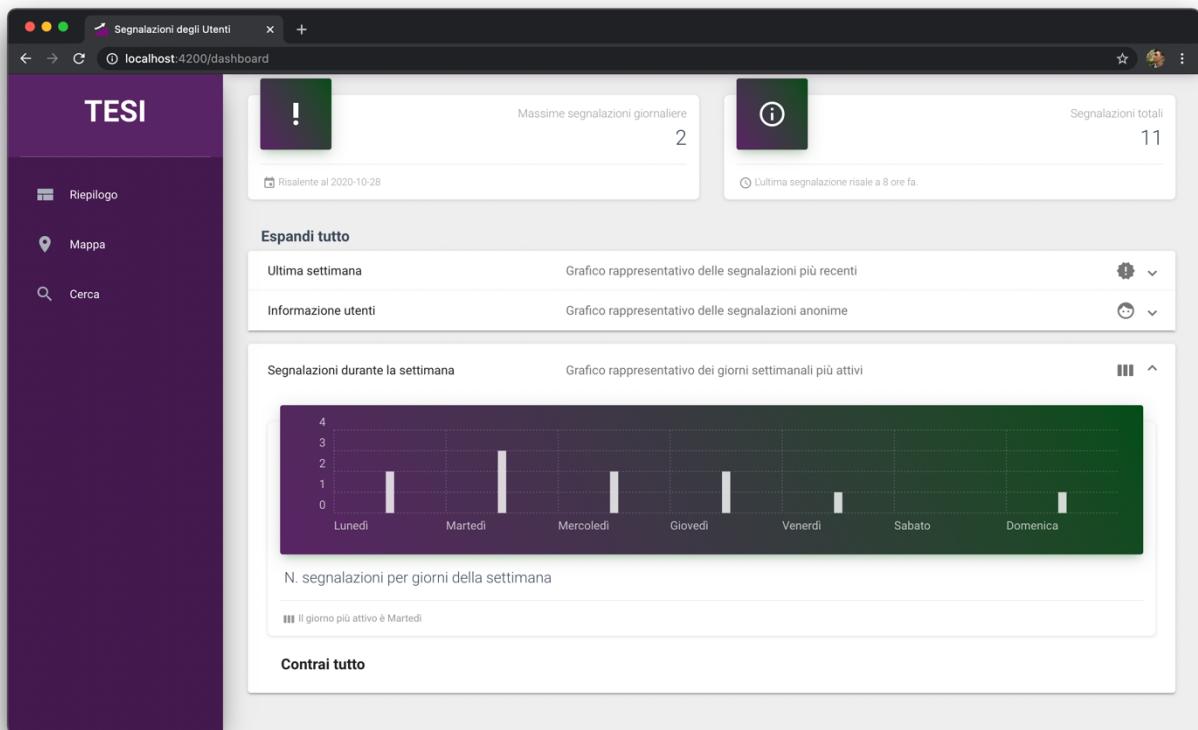


Figura 14 - Schermata Dashboard 3

Sezione Map

Dopo che il Report Service avrà ricevuto dal web server i dati sulle segnalazioni, la Route Guard permetterà all'Admin di visualizzare anche le sezioni Map e Search, raggiungibili dalla barra di navigazione laterale, altrimenti impedirà l'accesso e reindirizzerà l'admin alla Dashboard, in modo che il Report Service possa riprovare a contattare il web server.

Quando la sezione Mappa sarà renderizzata, l'admin, come previsto nel capitolo dell'Analisi Funzionale, dovrà poter filtrare, ad esempio, le segnalazioni in base a Gravità e Tipo di Segnalazione/Categoria delle stesse.

Nella Figura 14, l'Admin avrà dovuto selezionare “Leggera” e “Grave” tra le opzioni del campo associato alla Gravità, avrà cliccato sul bottone di conferma e infine avrà interagito con la segnalazione indicata dal Marker di colore verde.

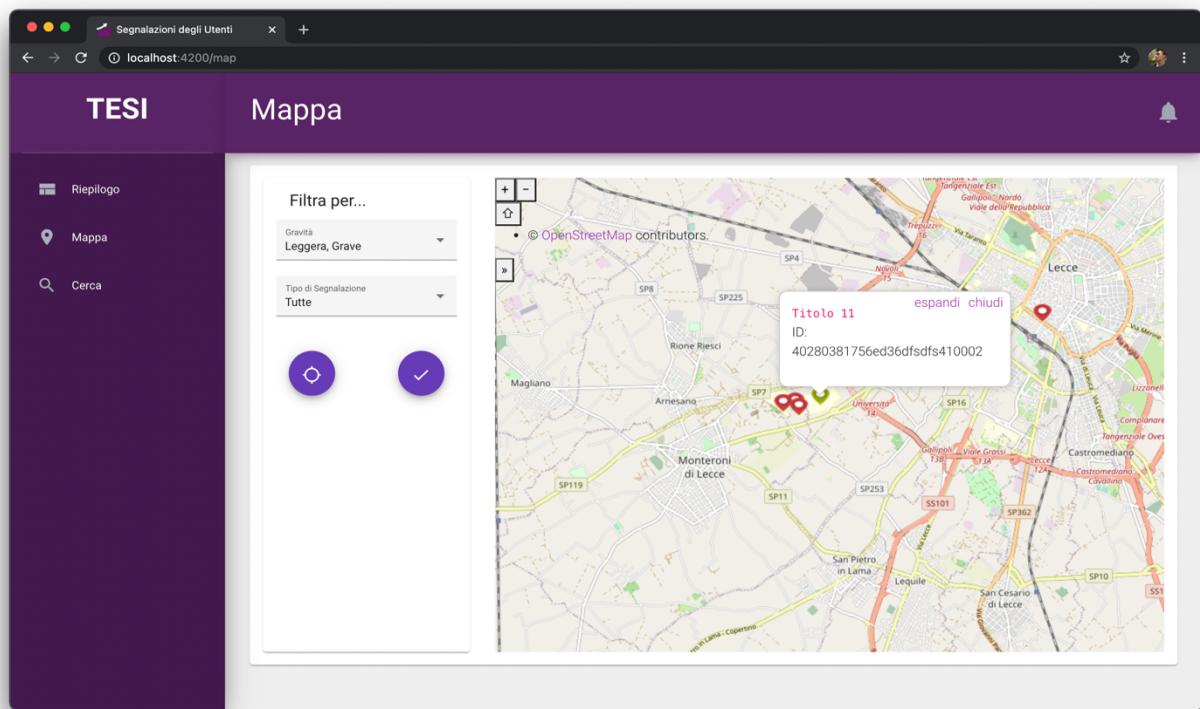


Figura 15 - Schermata Map 1

A questo punto, l'Admin potrà cliccare sulla voce “espandi” del popup per visualizzare tutti i dettagli legati alla segnalazione: si aprirà un dialog del genere:



Figura 16 - Schermata Map 2

Sezione Search

Anche per la Sezione Search vale lo stesso ragionamento della sezione Map: sarà accessibile tramite la barra di navigazione laterale solo se la Route Guard riconosce che il Report Service ha effettivamente ricevuto i dati sulle segnalazioni dal web server.

Non appena l'admin accederà alla sezione Search, visualizzerà una schermata come quella nella seguente figura:

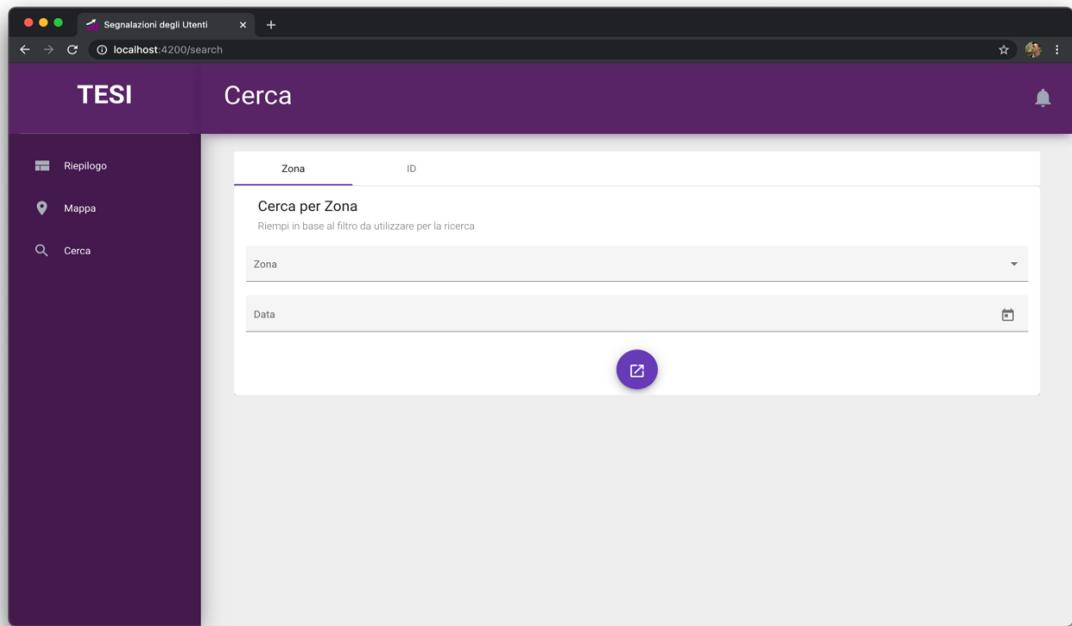


Figura 17 - Schermata Search 1

Da questa schermata, l'Admin potrà scegliere se effettuare una ricerca per Zona e/o Data e per ID, selezionando il Tab apposito. Se ad esempio sceglierà di cercare per Zona e/o Data, e il sistema avrà trovato dei risultati, l'Admin vedrà una tabella comparire:

A screenshot of the same web application window as Figure 17. The "Cerca" tab is now selected in the sidebar. The search form has been populated with "Ingegneria" in the "Zona" field and "10/1/2020 – 11/30/2020" in the "Data" field. Below the search form is a section titled "Risultati" with the sub-instruction "Sono stati trovati 2 risultati". A table displays two results: "Titolo 2" (Date: 2020-10-28 11:49:52.0, Categories: Evento Atmosferico, Segnalazione Città, Segnalazione Ambiente, Zona: Ingegneria) and "Titolo 3" (Date: 2020-10-28 11:51:20.0, Categories: Segnalazione Ambiente, Zona: Ingegneria). At the bottom of the table are pagination controls: "Items per page: 5", "1 – 2 of 2", and navigation arrows. A blue circular button with a downward arrow is located at the bottom right of the table.

Figura 18 - Schermata Search 2

Quando visualizza la tabella, l'admin può scegliere di salvare i risultati ottenuti in un file in formato PDF cliccando sul relativo bottone.

Qualora dovesse invece scegliere di effettuare una ricerca per ID, dovrà cliccare sul Tab ID. Quello che si troverà di fronte sarà una schermata del tipo:

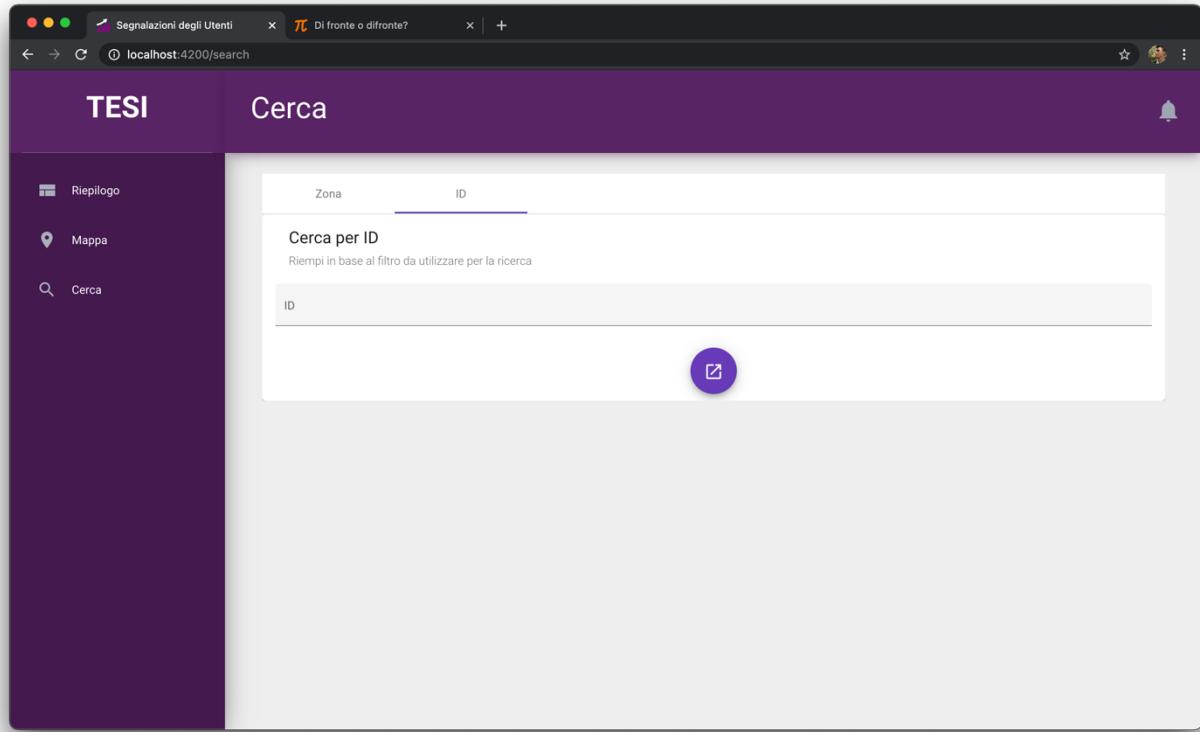


Figura 19 - Schermata Search 3

Infine, riempiendo il campo ID e cliccando sul bottone di invio, se la ricerca avrà esito positivo, l'Admin visualizzerà i dettagli del risultato nella seguente modalità:



Figura 20 - Schermata Search 4

Schermata di esempio – Notifica

Nel momento in cui il Report Service rileverà che i dati sulle segnalazioni in memoria non sono aggiornati, avviserà l'admin, in qualsiasi schermata egli si trovi, tramite un badge rosso sull'icona a forma di campana presente sulla barra di navigazione superiore.

La schermata esemplificativa è la seguente:

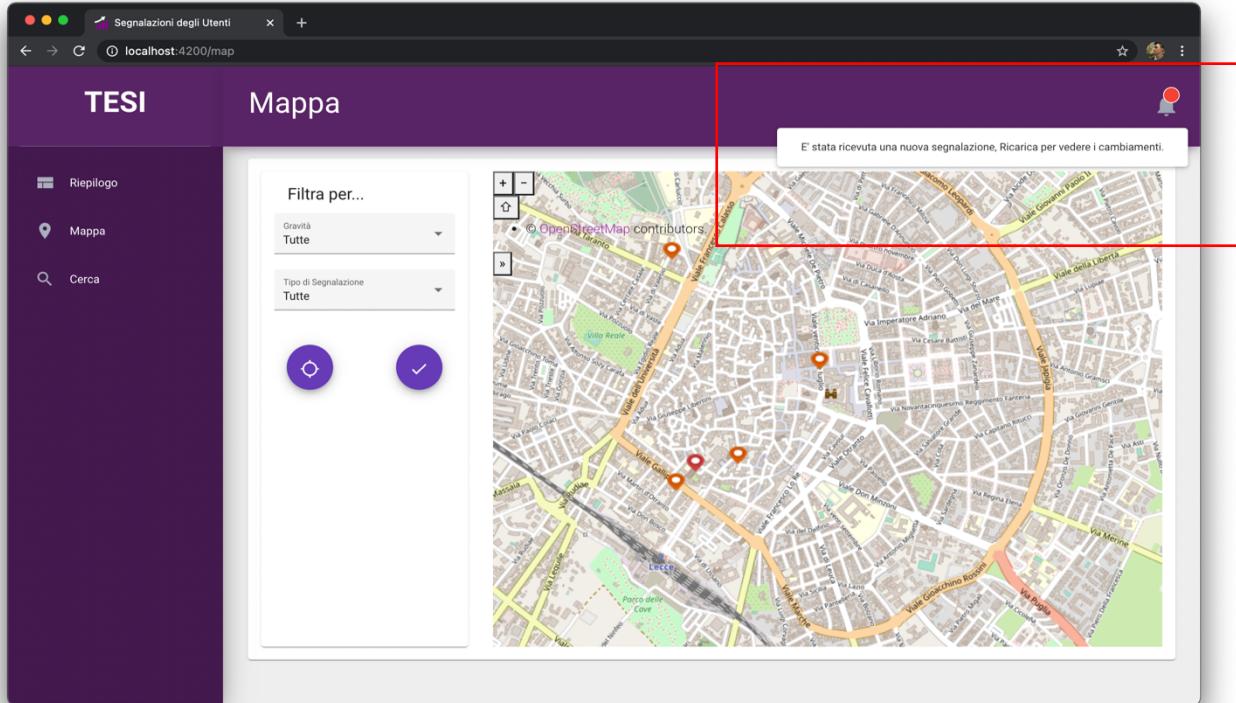


Figura 21 - Schermata Notifica

Cliccando sul messaggio di avviso, visualizzabile ogni volta che si clicca sull'icona ed è disponibile almeno una nuova segnalazione, l'admin vedrà la pagina ricaricarsi e sarà reindirizzato alla sezione Dashboard.

Test dell'applicazione

Per la cattura delle schermate dei paragrafi precedenti sono state utilizzate undici segnalazioni di test, presenti su un file di testo, scritte in formato JSON; ciò significa che la chiamata GET, a cui si *abbona* il ReportService, non ha come indirizzo il reale web server ma ha, come indirizzo, il file JSON di test che simula il formato in cui verranno ricevute le segnalazioni quando la chiamata GET attuale verrà sostituita con quella definitiva, cioè quella in cui viene indicato l'indirizzo del web server reale come destinazione.

Ad esempio, la segnalazione il cui *titolo* dell'evento è “Titolo 11” ha il seguente codice JSON associato, le cui informazioni sono visibili in maniera ridotta anche precedentemente in Figura 20:

```
"idsegnalazione": "40280381756ed36dfsdfs410002",
"dataSegnalazione": "2020-11-09 11:49:52.0",
"gpx": {
    "idgpx": "40280381756ed36a01756ed40d3c0000",
    "lat": 40.335997,
    "longt": 18.121587
},
"evento": {
    "idevento": "40280381755b07e101755b0cd7610000",
    "titolo": "Titolo 11",
    "descrizione": "Incidente stradale.",
    "data_attivazione": null,
    "categorias": [
        {
            "idcategoria": "qdbsnxj8274hgncksmcnh627djcnfg24",
            "nome": "Segnalazione Città"
        }
    ],
    "immagineUrl": null,
    "request_login": 0
},
"iduser": null,
"immagini": [],
"zona": {
    "idzona": null,
    "nome": "Ingresso Ecotekne",
    "descrizione": null
}
```

Figura 22 - Esempio segnalazione in formato JSON

Come si può notare in Figura 22, le coordinate espresse in termini di latitudine e longitudine fanno riferimento all'ingresso dell'Ecotekne, infatti tra le schermate relative alla sezione Map viene visualizzato un Marker in corrispondenza di tali coordinate.

Per quanto riguarda, invece, i test effettuati per verificare il funzionamento del sistema di notifica, non potendo fare una chiamata GET ad un web server reale, si è sfruttato un artificio del ReportService che emette la presenza di una nuova segnalazione, e quindi di una nuova notifica, dopo aver controllato che la dimensione della lista delle segnalazioni ricevute dal file di testo non sia cambiata, quindi essa viene emessa, in realtà, quando non c'è. Invertita la condizione di emissione della notifica, quando la chiamata GET sarà indirizzata al web server di SafeCity, il sistema di notifica, quindi, rileverà che la dimensione della lista delle segnalazioni potrà eventualmente aumentare, a distanza di 5 secondi, e, di conseguenza, attiverà il badge di notifica visualizzabile nella barra di navigazione superiore.

Criticità

Poiché il progetto realizzato dovrà essere integrato e ampliato all'interno di SafeCity, ho ritenuto opportuno raccogliere in una tabella le criticità emerse sia in fase di codifica sia precedentemente. La tabella ha 3 colonne: Nome della Criticità, Descrizione e Livello di Attenzione; quest'ultima è descritta da un valore numerico che va da 1 (poca attenzione richiesta) a 5 (molta attenzione richiesta).

Nome della Criticità	Descrizione	Livello di Attenzione
Mappa OpenLayers	<p>La mappa implementata grazie a OpenLayers ha lo svantaggio, rispetto alla concorrenza, di essere meno intuitiva da utilizzare per integrare le funzionalità ideate.</p> <p>Uno dei possibili problemi è legato ai Markers, che attiveranno il popup con l'anteprima della segnalazione cliccata: per come è implementata la funzionalità attualmente, il popup si aprirà se il click avviene all'interno di un raggio di 75 metri dalla segnalazione. Se ci sono più segnalazioni nel raggio di 75 metri verrà comunque aperto il popup dalla segnalazione più vicina.</p>	1
Report Service	<p>Il Report Service di Angular, allo stato attuale ha un possibile problema per quanto riguarda l'integrazione con SafeCity.</p> <p>Poiché, come già visto, la GUI è basata su un'unica chiamata GET, ossia quella che preleva tutte le segnalazioni che arrivano dal web server in ascolto, potrebbe accadere che se vengono trasmette troppe segnalazioni l'applicazione potrebbe subire dei rallentamenti dovuti alla permanenza in memoria di un numero elevato di segnalazioni.</p>	3

Notifica	<p>Il sistema di notifica implementato ha alla base un principio di funzionamento semplice: il Sistema non è in grado di ricevere una notifica Push, bensì ogni 5 secondi esso controlla che non ci sia una nuova segnalazione, effettuando la stessa chiamata GET al web server.</p> <p>Ciò significa che al momento dell'integrazione con SafeCity, sarebbe meglio modificare la chiamata (presente nel Report Service), in modo che venga effettuata una diversa per aumentare l'efficienza.</p>	2
----------	---	---

Glossario

Nome	Descrizione
Angular	Angular (o Angular 2) è un framework open source per lo sviluppo di applicazioni web con licenza MIT.
Design pattern	In informatica si definisce design pattern una soluzione progettuale generale ad un problema ricorrente.
Dialog	I Dialog, (o Alert Box) sono spesso usati per informare l'utente con un messaggio popup, o chiedere l'autorizzazione per compiere una specifica azione.
Framework	È un'architettura logica di supporto sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore.
Frontend	Parte dell'applicazione con cui l'utente interagisce direttamente
Open Source	Con open source, in informatica, si indica un tipo di software. In particolare, un software è reso open source per mezzo di una licenza per cui i detentori dei diritti favoriscono la modifica, lo studio e l'utilizzo del codice sorgente.
Plug-in	È un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie
UML	In ingegneria del software, UML (Unified Modeling Language, “linguaggio di modellizzazione unificato”) è un linguaggio di modellazione e di specifica basato sul paradigma object-oriented. L’UML svolge la funzione di “lingua franca” nella comunità della progettazione e programmazione a oggetti.
Web Server	In informatica un web server è un'applicazione software che, in esecuzione su un server, è in grado di gestire le richieste di trasferimento di pagine web di un client, tipicamente un web browser.

Elenco delle figure

FIGURA 1 - DIAGRAMMA DEI CASI D'USO	8
FIGURA 2 - SCHEMA MVC	13
FIGURA 3 - DIAGRAMMA DELLE SEQUENZE - NOTIFICA	15
FIGURA 4 - DIAGRAMMA DELLE SEQUENZE - DASHBOARD	16
FIGURA 5 - DIAGRAMMA DELLE SEQUENZE - MAP	17
FIGURA 6 - DIAGRAMMA DELLE SEQUENZE - SEARCH.....	18
FIGURA 7 - ALBERO DELLE DIPENDENZE	19
FIGURA 8 - LOGO DI ANGULAR	20
FIGURA 9 - LOGO DI NPM	21
FIGURA 10 - LOGO DI OPENLAYERS	22
FIGURA 11 - LOGO DI INTELLIJ	22
FIGURA 12 - SCHERMATA DASHBOARD 1	24
FIGURA 13 - SCHERMATA DASHBOARD 2	25
FIGURA 14 - SCHERMATA DASHBOARD 3	25
FIGURA 15 - SCHERMATA MAP 1	26
FIGURA 16 - SCHERMATA MAP 2	26
FIGURA 17 - SCHERMATA SEARCH 1	27
FIGURA 18 - SCHERMATA SEARCH 2	27
FIGURA 19 - SCHERMATA SEARCH 3	28
FIGURA 20 - SCHERMATA SEARCH 4	28
FIGURA 21 - SCHERMATA NOTIFICA	29
FIGURA 22 - ESEMPIO SEGNALAZIONE IN FORMATO JSON	30

Bibliografia e sitografia

Tipologia	Nome
Libro	UML e Unified Process di Jim Arlow e Ila Neustadt
Libro	Progettazione del Software e Design Pattern in Java di C.S. Horstmann
Sito	angular.io/docs
Sito	wikipedia.org
Sito	openlayers.org/en/latest/doc
Sito	ionos.it