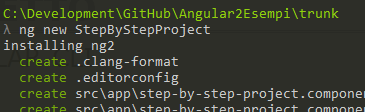
# Prerequisiti

* Installare/Aggiornare NodeJs
* Installare/Aggiornare npm
* Installare/Aggiornare angular CLI (npm install -g angular-cli)

Se si vuole scaricare il presente progetto, una volta fatto il checkout è necessario lanciare il comando npm install.

# STARTUP PROGETTO

* Creato il progetto con ANGULAR CLI



Questa operazione crea il progetto di sviluppo. Di seguito vengono descritti i principali file creati:

* + /app/app.module.ts 🡪 E 'il modulo radice della nostra applicazione, che indica come AngularJs assembla i vari moduli dell'applicazione.
  + main.ts 🡪 Compila l’applicazione con il JIT Compiler e configura il bootstrap dell’applicazione
  + index.html 🡪 Pagina indice della nostra applicazione
  + package.json 🡪 Contiene i metadati (nome, versione, descrizione, autore, ecc) della nostra applicazione ed elenca le dipendenze esterne (analogo del pom.xml)
  + style.css 🡪 File dove verranno importati eventuali CSS esterni all’applicazione
  + tsconfig.json 🡪 File di configurazione di Typescript
  + angular-cli.json 🡪 Configurazione di ANGULAR-CLI per il progetto
  + karma.conf.js 🡪 Configurazione dei test Karma
  + Crea inoltre un componente Angular di esempio
* Provata la corretta creazione del progetto attraverso il comando “ng serve” (se da errore vedi la seguente conversazione https://github.com/angular/angular-cli/issues/641)



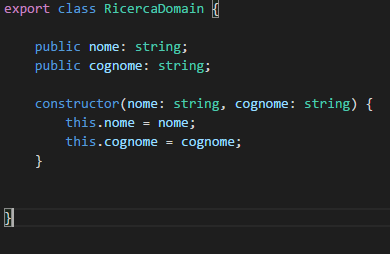
# SVILUPPO APPLICAZIONE BASE

* Creare una classe di dominio che conterrà i valori di ricerca del form (comando ng generate class RicercaDomain --spec=false)

Questa operazione crea i file

src/app/ricerca-domain.ts (classe vera e propria)

* Fatta implementazione della classe RicercaDomain

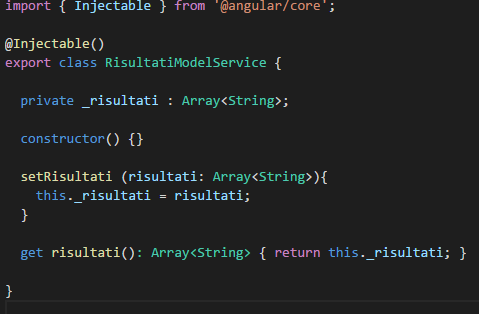


* Lanciato il comando ng generate service RisultatiModel --spec=false per creare il service singleton che manterrà i risultati di ricerca in tutta l’applicazione in modo da essere disponibile fra i componenti

Questa operazione crea i file

src/app/risultati- model.service.ts (classe vera e propria)

* Fatta implementazione di RisultatiModelService



* Lanciato il comando ng generate service Ricerca --spec=false per creare il service singleton che si occuperà di effettuare la ricerca

Questa operazione crea i file

src/app/ ricerca.service.ts (classe vera e propria)

* Fatta implementazione di RicercaService



---------------------------------------------------------------------------------------------------------

* Creato un nuovo componente Angular con il seguente comando ng generate component ricerca --spec=false

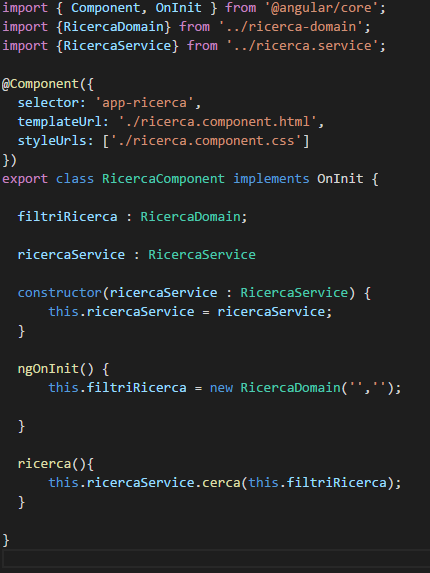
Il comando crea tutti i file necessari all’implementazione del component:

\ricerca.component.css 🡪 CSS del componente

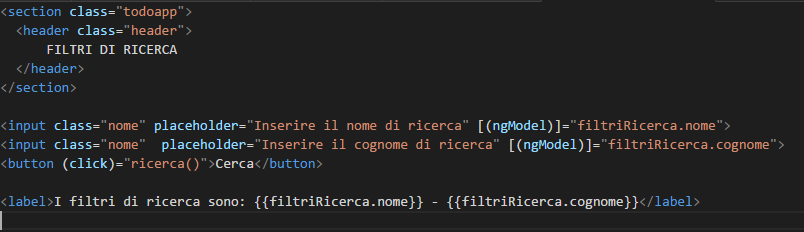
\ricerca.component.html 🡪 Layout del componente

\ricerca.component.ts 🡪 Logica del componente

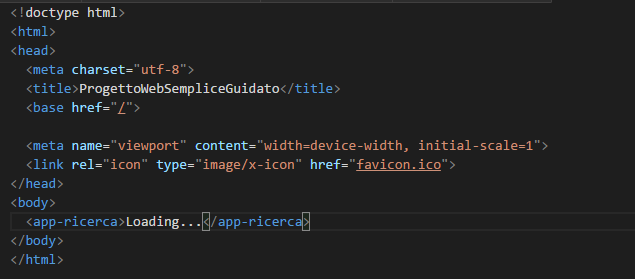
* Implementata la logica del componente (ricerca-component.component.ts)



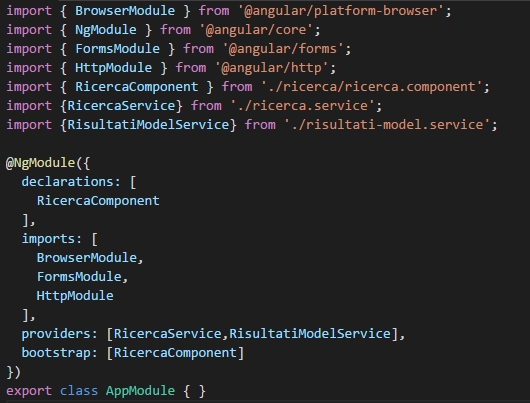
* Implementata la pagina web di ricerca (ricerca-component.component.html)



* Inserito il componente nella pagina Index.html dell’applicazione



* Inserito il componente (e relativi services) nel modulo app.module.ts dell’applicazione



* Lanciato server e fatto piccolo test (ng serve)

---------------------------------------------------------------------------------------------------------

* Creato un nuovo componente Angular con il seguente comando ng generate component risultati --spec=false

Il comando crea tutti i file necessari all’implementazione del component:

\risultati.component.css 🡪 CSS del componente

\risultati.component.html 🡪 Layout del componente

\risultati.component.ts 🡪 Logica del componente

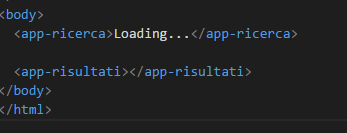
* Implementata la logica del componente (risultati-component.component.ts)



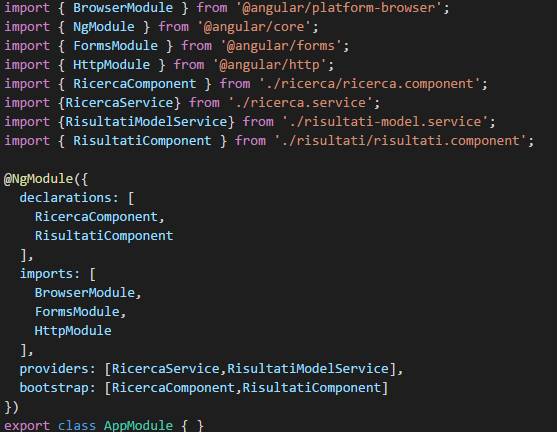
* Implementata la pagina web dei risultati (risultati-component.component.html)



* Inserito il componente nella pagina Index.html dell’applicazione



* Inserito il componente (e relativi services) nel modulo app.module.ts dell’applicazione



* Lanciato server e fatto piccolo test (ng serve)

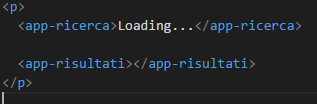
# ESEMPI AVANZATI – INSERITO ROUTING APPLICAZIONE

N.b. con l’attuale versione di Angular-CLI è stata disabilitata la generazione automatica del routing: verrà implementato manualmente.

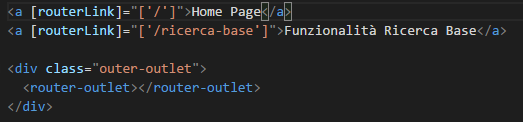
* Creato un nuovo componente Angular con il seguente comando ng generate component dashboardhome --spec=false (sarà una pagina vuota contenente semplicemente un testo fisso) e ne ho implementato la semplice e base logica



* Creato un nuovo componente Angular con il seguente comando ng generate component funzionalitaricerca --spec=false (sarà la pagina che ingloberà i due componenti implementati nel capitolo precedente) e ne ho implementato la semplice e base logiche

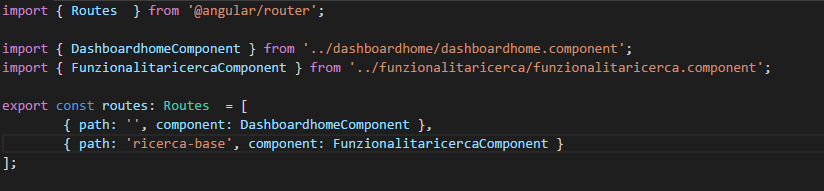
* Creato un nuovo componente Angular con il seguente comando ng generate component menu --spec=false (sarà il menù dell’applicazione)
* Ho implementato la pagina web del menù:



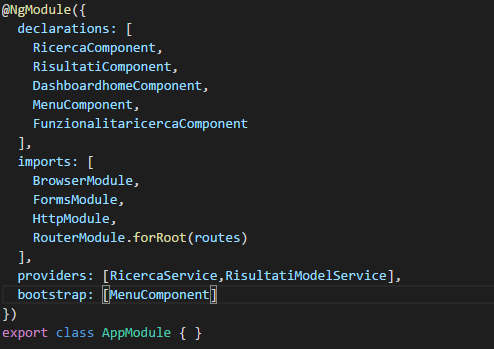
Gli attributi “routerLink” servono per mappare il link ai quali i vari componenti rispondono.

Il tag “router-outlet” è il placeholder dove andrà inserito il componente selezionato dalle logiche di routing definite al punto successivo (file app.routes.ts).

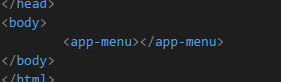
* Il file ts menu.component.ts non avrà alcuna logica e rimarrà quello generato da Angular-cli
* Creato il file app.routes.ts che conterrà le logiche di routing



* Aggiunto nel file app.module.ts il bootstrap di Angular Routing (importando in testa al file i moduli necessari) e modificato il componente di bootstrap dell’applicazione, che dovrà essere MenuComponent



* Modificata la pagina Index.html inserendo il menù come unico componente

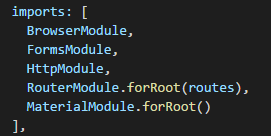


# ESEMPI AVANZIATI – INSERIMENTO DI MATERIAL DESIGN NELL’APPLICAZIONE

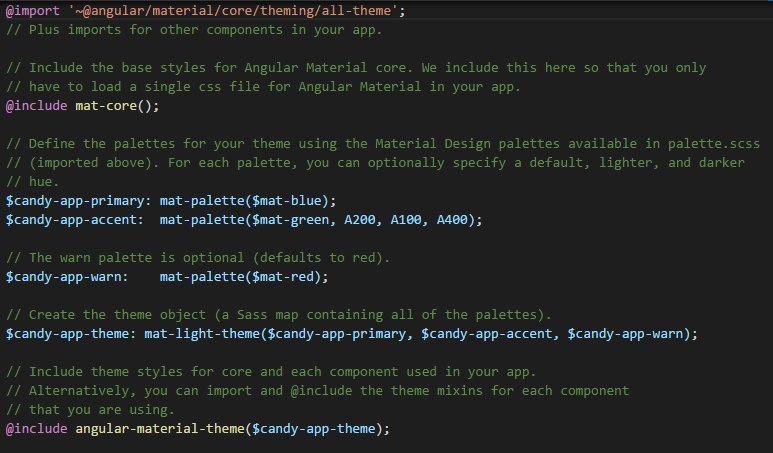
Google mette a disposizione un pacchetto di componenti chiamato “Material Design” da integrare nelle applicazioni angular 2 (ce ne sono anche altri, es. ng-bootstrap e primefaces).

Di seguito gli step fatti per integrare Material Design nella nostra applicazione Angular 2:

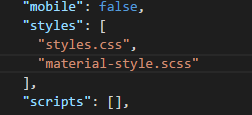
* Lanciato comando npm install --save @angular/material
* Importo angular material nell’applicazione (app.module.ts)



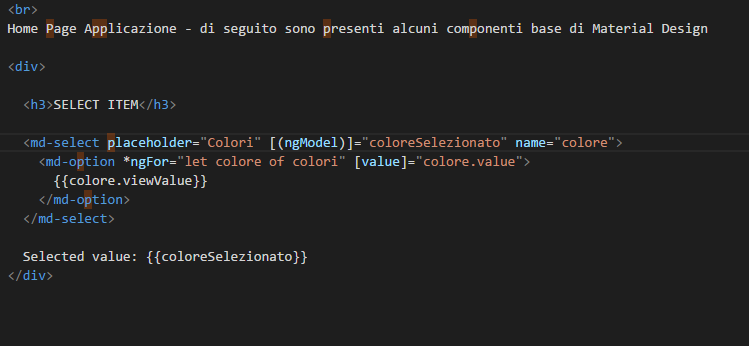
* Creato il file “material-style.scss” contenente il tema che andremo per la nostra applicazione (copiato ed incollato dal sito ufficiale)



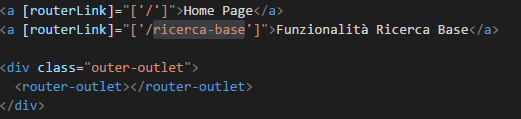
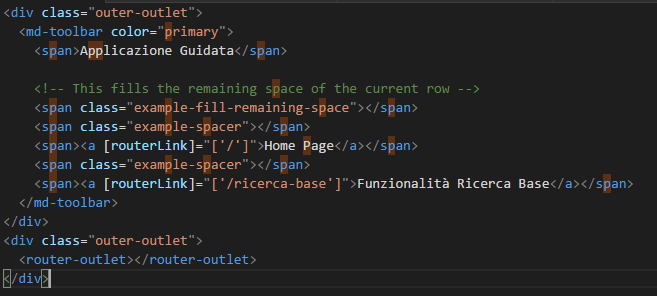
* Aggiunto il nuovo file scss nel file “angular-cli.json”



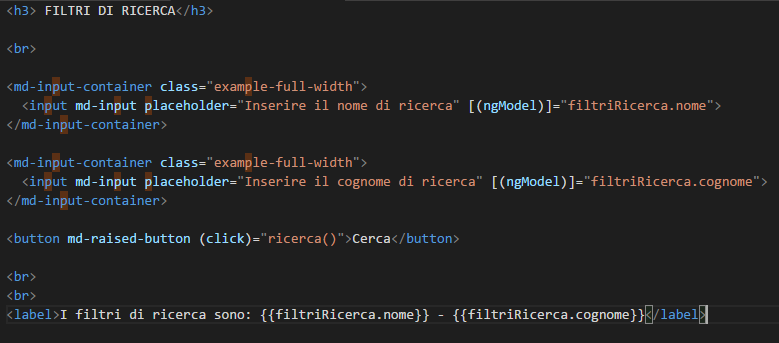
* Modificato il template della dashboard: sono stati inseriti alcuni componenti di material design di esempio



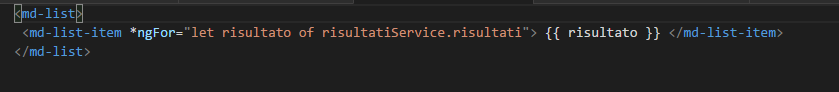
* Modificato il template del componente relativo al menu (menu.component.html) in modo da utilizzare i componenti forniti da Material Design

DA  A 

* Modificato il template del componente relativo alla ricerca (ricerca.component.html) in modo da utilizzare i componenti forniti da Material Design



* Modificato il template del componente relativo ai risultati (risultati.component.html) in modo da utilizzare i componenti forniti da Material Design

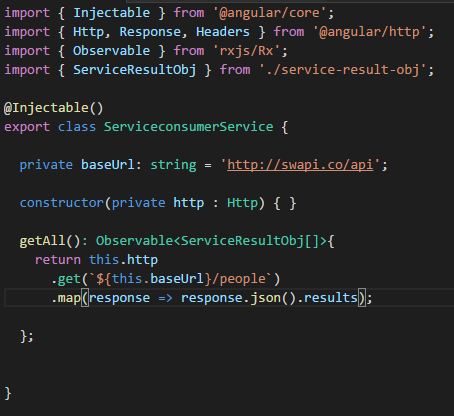


# ESEMPI AVANZATI – INSERITO CHIAMATA A SERVIZIO REST

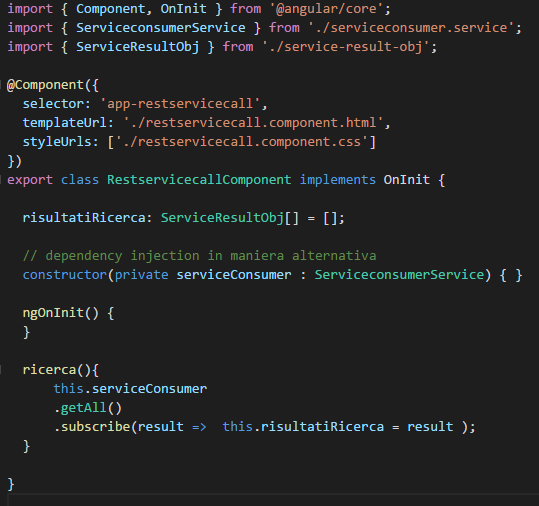
* Creato un nuovo componente Angular con il seguente comando ng generate component restservicecall --spec=false
* Creato un nuovo service Angular con il seguente comando ng generate service restservicecall/ serviceconsumer --spec=false
* Creato un nuovo oggetto Angular con il seguente comando ng generate class restservicecall/serviceResultObj che descriverà i singoli oggetti restituiti dal servizio http
* Inserito nel nuovo service le dipendenze necessarie ad effettuare una chiamata http



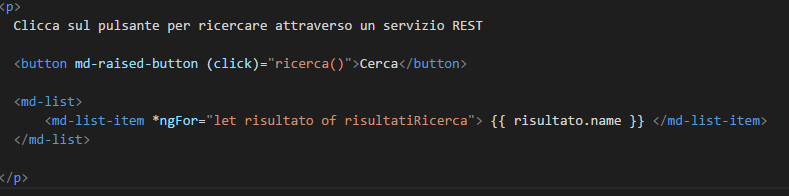
* Implementato il service che si occuperà di chiamare il servizio REST



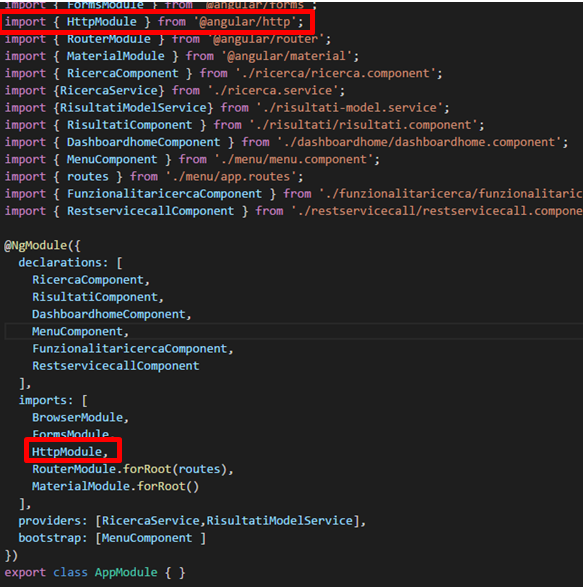
* Implementato il componente che si occuperà di richiamare il Service a fronte del click su un pulsante



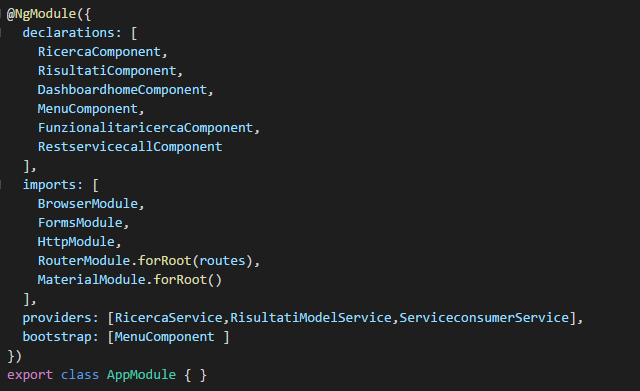
* Implementata la maschera contenente il pulsante di ricerca, e la visualizzazione dei risultati



* Aggiunto il modulo Angular “http” nel file app.module.ts



* Inserito il service implementato tra i provider



* Aggiunta la nuova funzionalità nel menù, con relativo routing

