

# WEB Programming and testing with a modern framework: AngularJS

***“MyFastNote”***

Documentazione elaborato

*Angelo Tallarita*  
matr.: O55000283

Anno accademico 2016/2017

# *Indice*

1. Introduzione.....	2
2. I nuovi campi del task.....	3
3. Modifica task.....	5
4. Task scaduti.....	6
5. Ricerca.....	7
6. Ordinamento.....	8
7. Conclusione.....	10

# *1. Introduzione*

Il progetto di fine corso consiste nell'apportare delle modifiche ad una *web application*, che consenta all'utente di gestire le proprie attività giornaliere, aggiungendo nuove features e funzionalità.

Il progetto è stato sviluppato interamente in *AngularJS* sfruttando le features di *HTML5* e *Angular Material*.

Durante la fase di sviluppo sono state seguite le principali *best practise* trattate durante il corso ed è stato seguito il paradigma “*Single Page Application*”

L'obiettivo del lavoro è stato quello di aggiungere funzionalità realmente utili a un potenziale utente e di mantenere l'interfaccia utente semplice e minimale, evitando di appesantire inutilmente il design, per questo motivo dal punto di vista visuale non sono state apportate particolari modifiche.

Il nome scelto per l'applicazione è “*MyFastNote*” e come suggerisce il nome è pensata per essere di utilizzo rapido e immediato.

Rispetto al progetto originale sono stati aggiunti nuovi campi al task, permettendo all'utente di inserire i nuovi campi durante la creazione dello stesso ed è stato reso possibile modificare un task già creato. Inoltre è stata aggiunta una nuova schermata per identificare e raccogliere i task già scaduti ed è stata implementata la ricerca e l'ordinamento. Sono state inoltre apportate migliorie minori relative alla risoluzione di alcuni bug.

Il primo passo dello sviluppo è stato eseguire una revisione generale del progetto, adattandolo alle principali *best practise*, in questa fase è stato individuato e corretto un bug relativo alla cancellazione del task.

## 2. I nuovi campi del task

La prima importante modifica è stata quella di introdurre i nuovi campi del task. I campi di partenza erano inadatti ad una corretta organizzazione e descrizione degli impegni di un potenziale utente, per questa ragione sono stati complessivamente implementati i seguenti campi:

- Titolo
- Descrizione
- Tags
- Tempo stimato (ore)
- Data
- Ora
- Priorità
- Stato

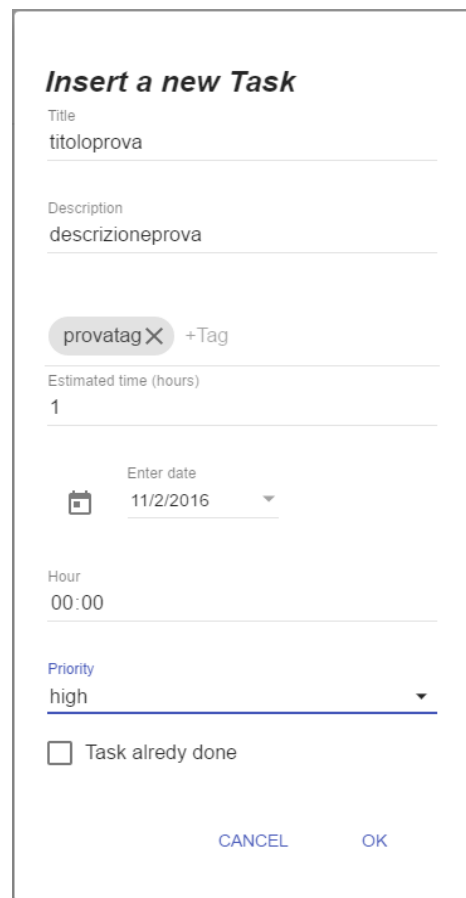
The image shows a web form titled "Insert a new Task". It contains several input fields: "Title" with the value "titoloprova", "Description" with "descrizioneprova", "Estimated time (hours)" with "1", "Enter date" with a calendar icon and the date "11/2/2016", "Hour" with "00:00", and a "Priority" dropdown menu set to "high". There is also a checkbox labeled "Task already done" which is unchecked. At the bottom right, there are "CANCEL" and "OK" buttons. A tag "provatag" with a close button and a "+Tag" button are also visible.

Fig. 1: Form di inserimento del task

Per permettere all'utente di inserire i nuovi campi è stato necessario utilizzare una *Custom Dialog* creando manualmente un template apposito in HTML5 utilizzando il material design. La finestra di inserimento richiede all'utente i campi precedentemente elencati che passerà successivamente al controller che provvederà alla creazione del task. Non sono stati fatti particolari controlli sul riempimento dei campi da parte

dell'utente poiché l'applicazione è stata pensata per permettere all'utente di appuntare solo ciò che è necessario e nel modo che preferisce, tuttavia è necessario inserire almeno il titolo per procedere alla creazione della nuova attività.

Per implementare i tag è stato usato il componente `<md-chips>` come previsto dal material design, è possibile inserire più tag (confermando con il tasto “invio”) e rimuovere quelli già inseriti. I tag risultano particolarmente utili nella ricerca per raggruppare immediatamente task affini.

Per la data è stata usata la direttiva `<md-datepicker>` e seppur data e ora sono due campi separati ai fini dell'inserimento è considerato un unico campo ai fini della creazione dell'oggetto relativo al task.

Tramite il campo priorità è possibile specificare quest'ultima fin da subito e lo stato indica semplicemente se il task è già stato effettuato. È stato scelto di permettere all'utente di specificare lo stato poiché potrebbe voler tenere traccia di vecchie attività già svolte per svariati scopi.

```
function addTask(ev) {  
  
  var parentEl = angular.element(document.body);  
  
  $mdDialog.show({  
    parent: parentEl,  
    targetEvent: ev,  
    templateUrl: 'app/components/customPrompt.template.html',  
    controller: TodoController,  
    controllerAs: 'ctrl',  
  }).then(function(task){  
  
    vm.createItem(task.title, task.done, task.priority, task.date,  
task.estimated, task.tags, task.description, task.time);  
  
  });  
  
}
```

*Fig. 2: Porzione di codice relativa alla funzione che lancia la form*

### *3. Modifica task*

In seguito è stata inserita la funzionalità relativa alla modifica di un task già inserito che permette appunto di modificare il valore di tutti i campi.

Selezionando un'attività il pulsante relativo alla modifica viene attivato e reso cliccabile, a questo punto cliccandolo sarà possibile lanciare la form di modifica, quest'ultima è molto simile alla form di inserimento con la sola differenza che i campi appariranno già riempiti con i valori attuali.

I valori attuali vengono passati alla form tramite il campo `locals`, ciò che viene passato al form è una copia dell'oggetto selezionato (effettuata rapidamente grazie ad `angular.copy`) in modo tale che le modifiche vengano apportate solo una volta ricevuta conferma dall'utente.

```
function editTask(task){
    vm.selectedItem.title = task.title;
    vm.selectedItem.done = task.done;
    vm.selectedItem.priority = task.priority;
    vm.selectedItem.date = task.date;
    vm.selectedItem.estimated = task.estimated;
    vm.selectedItem.tags = task.tags;
    vm.selectedItem.description = task.description;

    storageService.set(vm.items);

}
```

*Fig. 3: Porzione di codice relativa al salvataggio della modifica sul task*

## 4. Task scaduti

Durante lo sviluppo è rapidamente emerso come il campo data e ora fosse di importanza fondamentale per un potenziale utente dell'applicativo, sorge dunque l'importanza di raggruppare in una schermata apposita (*“Expired”*) tutti i task scaduti, sia per averli semplicemente raccolti insieme sia per toglierli dalla lista delle attività ancora da svolgere.

Per la realizzazione di questa funzionalità è stata semplicemente usata una funzione di filtro come negli altri casi già implementati.

```
function exp(item){  
    var data = new Date();  
    if(item.date < data.getTime())  
        return true;  
    else return false;  
}
```

Fig. 4: Porzione di codice che mostra la filter function

Quando un task scade viene dunque spostato nella scheda apposita, si noti come (per scelta implementativa) un task inserito senza specificare la data venga direttamente inserito nella scheda dei task scaduti poiché gli viene affibbiata la data attuale.

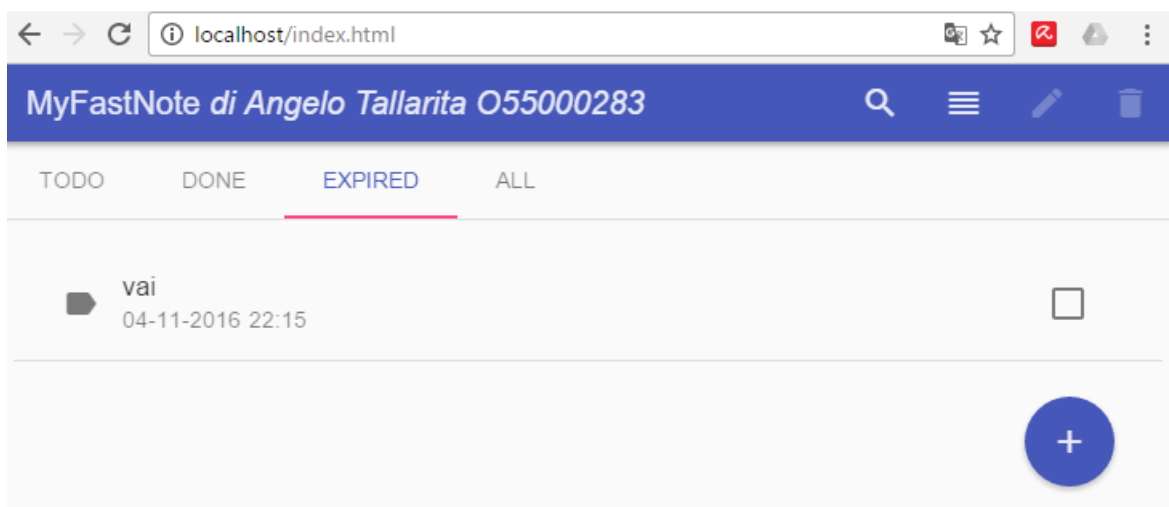


Fig. 5: Vista dei task scaduti

## 5. Ricerca

La ricerca è una funzionalità molto utile per trovare i task desiderati, soprattutto se diventano molti. Per scelta implementativa la ricerca viene effettuata su tutti i campi del task, è possibile ad esempio trovare un task scrivendo parte della sua descrizione o semplicemente un suo tag, in modo da rendere più semplice il procedimento.

Una volta cliccato sull'apposito pulsante viene mostrata, grazie alla direttiva ng-show, una search bar che effettua una ricerca in tempo reale su ciò che viene digitato.

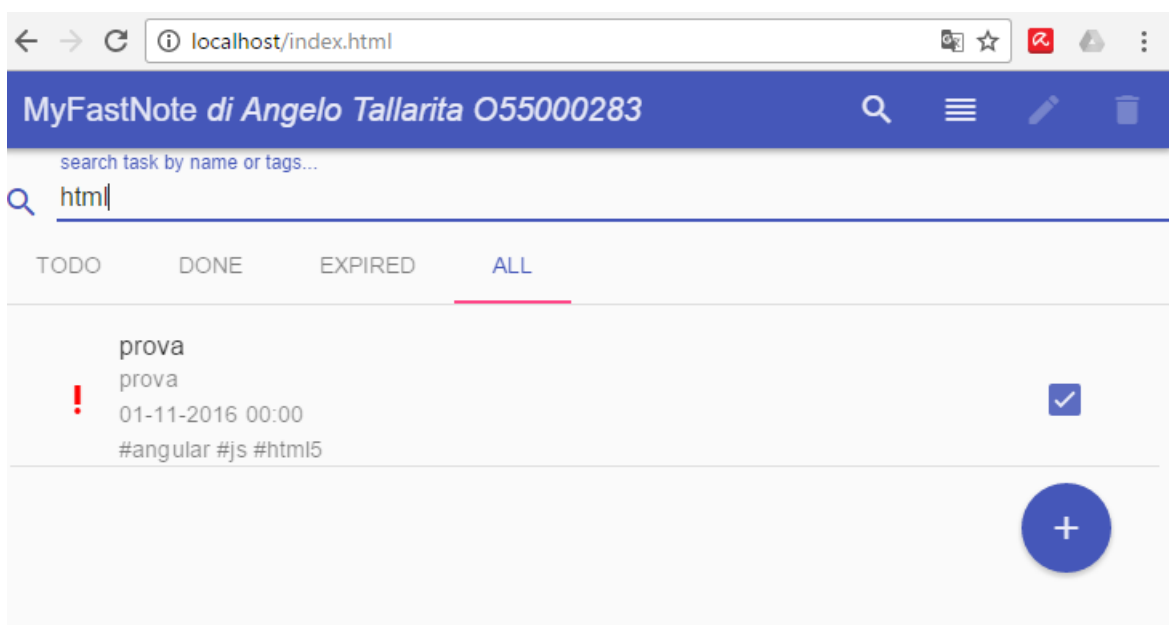


Fig. 6: Esempio di ricerca

Una volta cancellato il testo di ricerca essa verrà annullata, inoltre se in qualunque momento si clicca nuovamente il tasto di ricerca verrà nascosta la search bar e annullata la ricerca in corso.

Per far tutto ciò è stato necessario aggiungere un nuovo filtro a quelli già presenti nella lista situata nel template apposito, per far ciò è stato aggiunto un nuovo campo alla direttiva per passare il testo da impiegare nel filtro.



## 6. Ordinamento

L'ultima funzionalità aggiunta all'applicativo è l'ordinamento, essa consente all'utente di cambiare l'ordine in cui vengono mostrati i task rendendo dunque più agevole la navigazione.

Basta cliccare una volta l'apposito tasto per mostrare la lista ordinata in ordine alfabetico, cliccando nuovamente il tasto i task verranno questa volta ordinati per data (dal più vecchio al più recente), infine con un terzo click si ritorna all'ordinamento di partenza ovvero i task vengono mostrati per ordine di inserimento.

```
function orderItem(){
    if(vm.order==0){
        vm.items.sort(function(a, b){{
            var x = a.title.toLowerCase();
            var y = b.title.toLowerCase();
            if (x < y) {return -1;}
            if (x > y) {return 1;}
            return 0;
        }});
        vm.order = 1;
    }else if (vm.order==1){
        vm.items.sort(function(a, b){{
            var x = new Date(a.date);
            var y = new Date(b.date);
            if (x < y) {return -1;}
            if (x > y) {return 1;}
            return 0;
        }});
        vm.order = 2;
    } else if(vm.order==2){
        vm.items.sort(function(a, b){{
            var x = a.id;
            var y = b.id;
            if (x < y) {return -1;}
            if (x > y) {return 1;}
            return 0;
        }});
        vm.order = 0;
    }
}
```

*Fig. 7: Porzione di codice che mostra la funzione di ordinamento*

Come si evince in Figura 7, per l'ordinamento, è stata usata la funzione `.sort()` che ha permesso in maniera semplice e rapida di riordinare il vettore grazie a delle semplici funzioni di confronto tra elementi diversi. È stato inoltre impiegato un flag per individuare quanti click sono stati effettuati sul bottone.

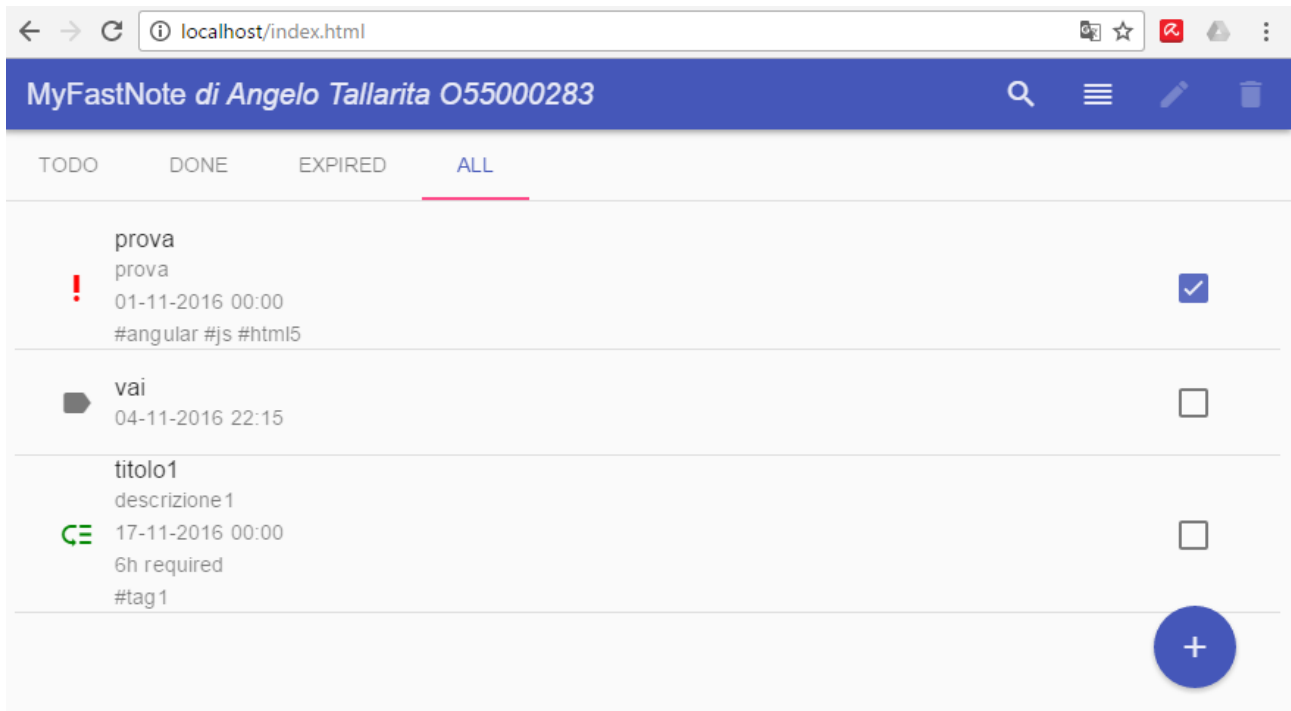


Fig. 8: Task ordinati per data

## ***7. Conclusioni***

Il progetto è stato svolto seguendo le principali best practise, è stato usato un solo controller principale poiché ha superato di poco le 200 righe di codice, contro le 400 di limite teorico, inoltre i template sono stati inseriti in file HTML appositi ed è stata utilizzata la dependency injection. I file sono stati opportunamente suddivisi in sottocartelle e il codice è stato commentato.

Tutta l'applicazione è situata su una singola pagina (SPA) ed è stata sviluppata con lo scopo di essere semplice ed efficace cercando di sfruttare le potenzialità di angularJS, HTML5 e utilizzando le direttive angular material il più possibile.

In un'eventuale iterazione successiva si potrebbero aggiungere ulteriori funzionalità, come una notifica automatica quando uno dei task sta per scadere o il supporto di un vero database.