Corso di Laurea Magistrale in Ingegneria Informatica

WEB PROGRAMMING AND TESTING WITH A MODERN FRAMEWORK : ANGULAR JS

Progetto sviluppato nell'ambito del Corso di AngularJS

Paolo Walter Modica O55000288

Salvatore Quattropani O55000294

ANNO ACCADEMICO 2016-2017

Indice generale

Introduzione

Il progetto illustrato nella presente relazione tratta dello sviluppo di un'applicazione Web realizzata utilizzando il Framework AngularJS.

In particolar modo, partendo da un'applicazione elementare, tale "**ToDo App**", fornitaci dall'Ing. Burgio di BAXEnergy e avente funzionalità di registrazione di promemoria, sono state aggiunte diverse caratteristiche e funzionalità che ne hanno arricchito il funzionamento.

Obiettivi di progetto

Lo scopo principale di questo elaborato progettuale è di comprendere ed utilizzare le varie funzionalità offerte dal framework AngularJS ed il modello architetturale **Model-View-Viewmodel** (o **Model-View-Controller**) sul quale questo si basa.

L'applicazione realizzata – TaskApp

Partendo dal progetto d'esempio è stata realizzata un'applicazione Web, denominata ai fini del progetto "**TaskApp**".

TaskApp è un'agenda online che permette all'utente di gestire in maniera semplice ed efficiente i propri impegni.

Attraverso un'interfaccia grafica snella ed intuitiva, usabile sia su dispositivi desktop che mobile, l'utente ha la possibilità di inserire i propri impegni, consultarli in qualsiasi momento, modificarne lo stato (da fare/fatto) e il livello di priorità attribuitogli e, quando non più interessato, eliminarli dalla lista.

Inoltre l'applicazione implementa la funzionalità di reminder, trasmettendo una e-mail all'utente per notificarlo di eventuali incombenze da fare e prossime alla data di scadenza.

Funzionalità Implementate

Per realizzare TaskApp sono state eseguite le seguenti modifiche ed aggiunte le seguenti funzionalità all'applicazione d'esempio fornitaci inizialmente:

- modifica e personalizzazione dell'interfaccia grafica, resa completamente responsive;
- aggiunta del profilo utente e della registrazione dello stesso;
- validazione dei campi in fase di registrazione dell'utente, con verifica del formato dell'indirizzo e-mail inserito;
- verifica dell'esistenza di un utente in fase di registrazione e login;
- funzionalità di modifica del profilo utente, con chiusura della sessione corrente per evitare inconsistenze nei dati;
- funzionalità di login e logout;
- trasmissione di una e-mail per notificare l'avvenuta e corretta registrazione dell'utente nel sistema;
- trasmissione di una e-mail per notificare l'avvenuto cambio dell'indirizzo e-mail associato all'utente.
- mantenimento della sessione in caso di chiusura della finestra del browser senza logout;
- aggiunta all'oggetto ToDo dei seguenti campi informativi: descrizione, priorità e data di scadenza;
- aggiunta di una select box che permette di cambiare lo stato del ToDo da “da fare” a “fatto” e viceversa;
- validazione dei campi in fase di inserimento di un nuovo impegno, con limite sulla data di scadenza dell'impegno (no date antecedenti alla data odierna) e sul numero di caratteri dei vari campi testuali;
- visualizzazione specifica delle informazioni del singolo ToDo;

- gestione unificata della lista dei ToDo selezionati tra le diverse tab dell'applicazione;
- selezione e cancellazione multipla degli impegni in lista;
- filtri di ricerca dei ToDo per nome e per data di scadenza;
- persistenza dei dati utilizzati dall'applicazione con database relazionale MySQL;
- back end dell'applicazione Web con Node.Js;
- funzionalità di reminder dell'applicazione, con trasmissione di una email di notifica per ricordare all'utente di eventuali incombenze da fare e prossime alla scadenza.

1. Modifica dell'interfaccia grafica

L'interfaccia grafica è stata realizzata sfruttando i diversi elementi forniti dal framework **Angular Material**, insieme con diverse librerie di terze parti ed un CSS personalizzato e completamente realizzato da noi.

Per permettere l'utilizzo dell'applicazione su qualsiasi dispositivo, sia esso desktop o mobile, il CSS è stato realizzato in accordo ai principi del responsive design.

2. Il profilo Utente

Al fine di restituire un'esperienza d'uso dell'applicazione in oggetto il più personale possibile è stata aggiunta la possibilità di inserire il proprio profilo Utente.

L'utente è caratterizzato da nome, cognome ed un indirizzo e-mail, che utilizzerà come credenziale di accesso al sistema.

L'inserimento delle informazioni per la registrazione dell'utente avviene attraverso una finestra di dialogo ottenuta grazie al servizio **$mdDialog**, sul quale sono stati inseriti i controlli per la validazione dei dati inseriti.

I controlli di validazione sugli inserimenti sono stati effettuati sfruttando gli attributi HTML5 **required**, **maxlength** e **minlength** per impedire eventuali problemi in fase di inserimento dei dati dell'utente nel database relazionale MySQL costituito per l'applicazione (vedi punti successivi).

Alla pressione del bottone **Save** viene richiamato un metodo del servizio **userService**, il quale genera un evento sulla socket del sistema che viene catturato dal back end dell'applicazione.

L'evento **registration** innesca nel back end una funziona cui è demandato il controllo dell'eventuale presenza all'interno del database di un profilo utente avente il medesimo indirizzo e-mail di quello ricevuto e, in caso di esito negativo, innesca l'inserimento delle informazioni ricevute sul database relazionale e l'invio via e-mail di un messaggio di notifica di avvenuta registrazione.

Ciascun utente registrato è individuato in modo univoco da un identificativo **IDUser** calcolato in modo pseudocasuale.

3. Funzionalità di Login e Logout

Una volta effettuata la registrazione, l'utente ha la possibilità di accedere al sistema attraverso la funzionalità di **login**.

Questa richiama una finestra di dialogo del servizio **$mdDialog** nel quale l'utente ha la possibilità di inserire le proprie credenziali d'accesso (il solo indirizzo e-mail).

Alla pressione del bottone Login viene richiamato un metodo del **loginService** cui è demandato la generazione di un evento che viene trasmesso, attraverso la socket di sistema, verso il back end.

Quest'ultimo, alla ricezione dell'evento, lancia una funzione che verifica l'eventuale corrispondenza delle credenziali inserite con i profili presenti all'interno del database e, in caso di esito positivo, lancia un evento verso il front end, il quale cambia lo stato dell'utente corrente in **loggedIn = true**.

Al cambiamento dello stato dell'utente corrente, l'interfaccia grafica cambia mostrando la lista dei ToDo, caratteristica saliente dell'applicazione, e accogliendo l'utente con un messaggio personalizzato con il nome dello stesso, ottenuto sfruttando il **data binding** con i dati presenti nel controllore associato.

4. Modifica del profilo utente

L'utente ha la possibilità di modifica qualsiasi informazione a lui associata e presente sul suo profilo.

Nell'eventualità in cui questi decida di modificare il proprio indirizzo e-mail con uno differente da quello già registrato, l'applicazione lancerà un evento sulla socket di sistema e verso il back end per notificare il cambiamento. L'applicazione di back end quindi lancerà una funzione che trasmetterà un messaggio e-mail di notifica al nuovo ed al vecchio recapito dell'utente.

5. Mantenimento della sessione

In caso di volontaria o accidentale chiusura della finestra del browser senza aver effettuato il logout, l'applicazione è in grado di ripristinare l'ultima sessione attiva dell'utente.

Ciò è stato realizzato con la creazione, in fase di login ed all'interno del **localStorage** del web browser, di un oggetto avente le informazioni relative all'ultimo utente che ha effettuato il login con successo.

In fase di apertura dell'applicazione sulla finestra del browser, e conseguente inizializzazione del DOM, viene richiamata, sfruttando la direttiva **ngInit**, una funzione del **LoginController** preposta a ripristinare eventuali sessioni attive presenti sul localStorage.

6. Inserimento dei ToDo

Gli oggetti ToDo rappresentano gli impegni che l'utente vuole inserire nella lista delle cose da fare.

Per meglio descriverli, ed ottenere così un'esperienza d'uso quanto più simile a quella di un'agenda, sono stati aggiunti a questi degli ulteriori campi, in particolare la descrizione, la priorità (che può assumere i valori -1, 0 ed 1) e la data di scadenza.

L'inserimento delle informazioni relative al ToDo dell'utente avviene attraverso una finestra di dialogo ottenuta grazie al servizio **$mdDialog**, sul quale sono stati inseriti i controlli per la validazione dei dati inseriti.

I controlli di validazione sugli inserimenti sono stati effettuati sfruttando gli attributi HTML5 **required**, **maxlength** e **minlength** per impedire eventuali problemi in fase di inserimento dei dati dell'utente nel database relazionale MySQL costituito per l'applicazione (vedi punti successivi).

Inoltre per impedire la selezione di una data antecedente alla data odierna sul datepicker è stato sfruttato l'attributo di Angular Material **md-min-date**, inizializzato al valore della data odierna attraverso la funzione **Date**() di JavaScript.

Ciascun ToDo inserito è associato all'utente attraverso il campo univoco **IDUser**.

Alla pressione del bottone **Add** viene richiamato un metodo del servizio **itemService**, il quale genera un evento sulla socket del sistema che viene catturato dal back end dell'applicazione.

L'evento innesca nel back end una funziona cui è demandato l'inserimento delle informazioni ricevute sulla tabella del database relazionale relativa ai ToDo.

7. Gestione dei ToDo

I ToDo così inseriti dall'utente vengono visualizzata sull'interfaccia grafica.

Tramite questa è possibile cambiare la priorità di ciascun impegno, visualizzare in modo approfondito la sua descrizione, cambiare il suo stato (da “da fare” a “fatto” e viceversa) ed eventualmente eliminarlo.

E' possibile filtrare gli impegni visualizzati nella lista per nome del compito e per data di scadenza dello stesso. Per fare ciò è stata utilizzata la proprietà **filter** dei ToDo presenti nella vista.

E' possibile effettuare una selezione multipla dei ToDo per rimuoverne più di uno contemporaneamente.

Inoltre gli impegni vengono mostrati sulla schermata in 3 tab differenti, a seconda che il loro stato sia “da fare”, “fatto” o che li si voglia visualizzare tutti.

Ogni singolo ToDo è visto come istanza di una **custom directive**, cui sono demandati l'istanziamento dell'oggetto nel DOM, le funzionalità di cambiamento di stato e priorità, di visualizzazione dei dettagli e di selezione (toggle).

Attraverso una struttura dati condivisa tra le due custom directives dei ToDo (una relativa ai ToDo in stato “da fare” ed una relativa a quelli in stato “fatto) viene gestita in maniera unificata la lista dei ToDo selezionati tra le diverse tab dell'applicazione, in modo da avere consistenza degli elementi (e dei loro attributi) mostrati nella vista.

Ogni modifica apportata a ciascun ToDo presente nella lista richiama una funzione di modifica che lancia un evento nella socket di sistema e verso il back end, il quale esegue una funzione di aggiornamento del contenuto del database MySQL.

8.Il reminder

Il **reminder** è una funzionalità peculiare dell'applicazione realizzata che trasmette una e-mail di notifica per ricordare all'utente di eventuali incombenze da fare e prossime alla scadenza.

Questa funzionalità è completamente implementata nel back end dell'applicazione, realizzato in Node.Js.

L'applicazione del back end esegue periodicamente un controllo di tutti i ToDo associati agli utenti presenti nel database ed in stato “da fare”, e per questi calcola la differenza tra la loro data di scadenza e la data odierna.

Se questa differenza risulta essere uguale o minore a 24 ore, allora l'applicativo del back end trasmetterà un messaggio all'utente notificando l'incombenza in scadenza, finché questa non verrà posta nello stato “fatto” o verrà superata la data di scadenza.

Specifiche dell'architettura del progetto

Il DataBase

Al fine della persistenza dei dati utilizzati dall'applicazione è stato realizzato un database relazionale con MySQL, strutturato in due tabelle:

- **Utenti**, contenente i dati relativi agli utenti dell'applicazione. Ogni utente è individuato in modo univoco attraverso un identificativo, **IDUser**.
- **Dati**, contenente i dati relativi ai ToDo inseriti dagli utenti. Ogni ToDo è associato al relativo utente attraverso l'identificatore **ID = IDUser**.

Il Back End

Per lo sviluppo del back end dell'applicazione prevista dal progetto è stato utilizzato il framework Node.js. L'applicazione server-side del sistema implementa al suo interno diversi moduli, ognuno dei quali adatti allo svolgimento di uno specifico compito e al soddisfacimento di uno dei requisiti del sistema:

- **http**, modulo facente parte dei core modules di Node.js e che permette di creare un'istanza di un Web server, necessario per gestire la comunicazione con l'applicazione del front end.
- **express**, che è il modulo di riferimento di Express.js, un framework minimalista per Node.js utilizzato per lo sviluppo di applicazioni web e mobile.
- **Socket.io**, che è una libreria JavaScript che viene utilizzata per la costruzione di un canale di comunicazione bidirezionale, real-time ed event-driven tra client e server di un'applicazione Web. Esso è inoltre composto da due parti, una libreria client-side, per l'implementazione nel front end, ed una libreria server-side per l'implementazione all'interno di un'applicazione Web in Node.Js . La socket creata con Socket.io è stata utilizzata per realizzare la comunicazione event-driven tra il front end ed il back end dell'applicazione, al fine della

trasmissione dei dati da inserire o modificare nel database e dell'implementazione della funzionalità di notifica via e-mail all'occorrenza di certi eventi.

- **Nodemailer**, che è un modulo che permette di utilizzare un transporter

SMTP per implementare il servizio di invio di messaggi di posta

elettronica all'interno di un'applicazione server in Node.Js . Viene utilizzato per realizzare il servizio di notifica all'utente in caso di registrazione, modifica dell'indirizzo e-mail associato e impegno prossimo alla data di scadenza.

- **mysql**, driver di Node.Js che permette di gestire la connessione ad un DB MySQL remoto e le operazioni di inserimento, modifica, cancellazione ed estrazione di dati dallo stesso.

La funzionalità di reminder del sistema viene implementata interamente in questa parte attraverso la funzione **notifierServer** che, richiamata con periodo T dalla funzione JavaScript **setInterval**, esegue un confronto fra la data odierna e la data di scadenza di tutti gli impegni in stato “da fare” relativi a ciascun utente registrato sul database e, in caso di scadenza imminente, trasmette un messaggio di posta elettronica all'utente interessato.

Il Front end

Il front end dell'applicazione è stato sviluppato interamente utilizzando il framework AngularJS.

L'applicazione Web è costituita, in accordo al modello architetturale **MVVm** (o **MVC**) da:

- modulo **todoApp**, modulo principale dell'applicazione;
- i controllori **LoginController** e **TodoController**, cui sono demandate le funzionalità relative rispettivamente all'inserimento di un nuovo profilo utente ed operazioni di login/logout dello stesso e gestione dei ToDo;
- i servizi custom **socketService**, **storageService**, **userService**, **loginService** ed **itemService**

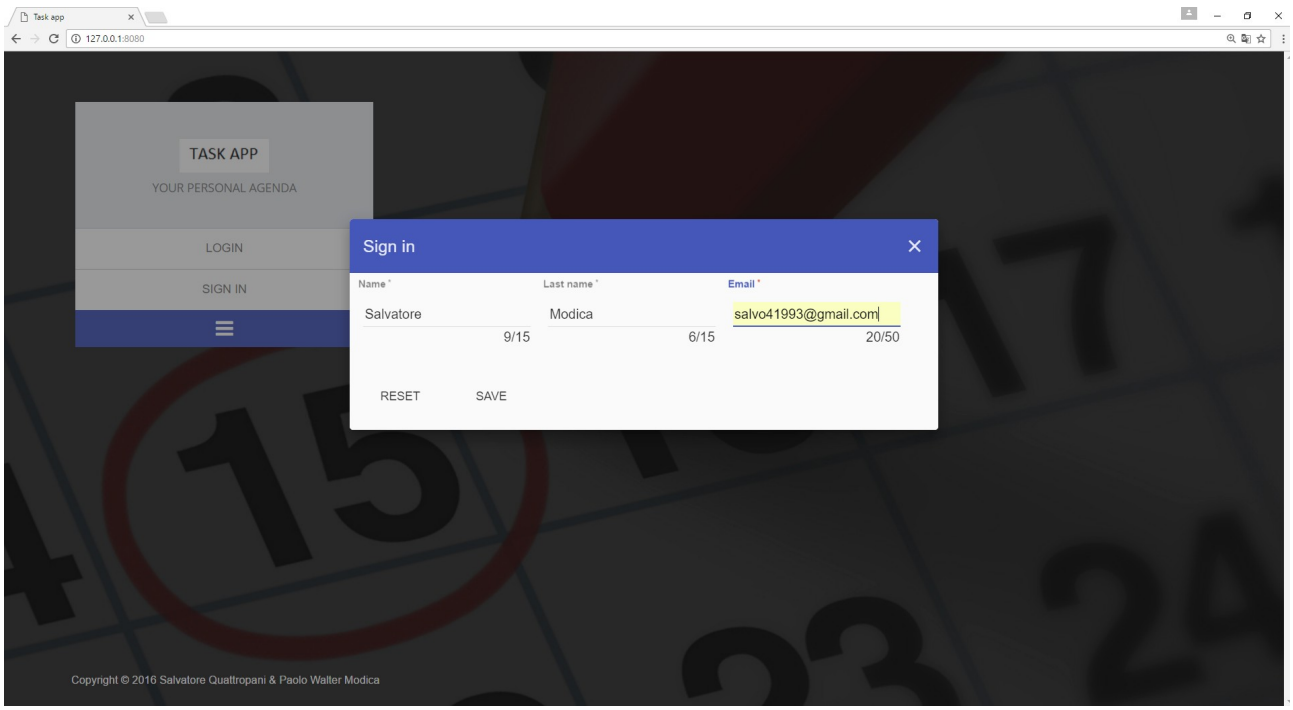
    1. socketService implementa i metodi che utilizzano la socket realizzata con Socket.IO;

2. storageService implementa tutti i metodi che si interfacciano con il back end per agire sul DB dell'applicazione;
3. userService implementa i metodi che, sfruttando il servizio storageService, permettono la creazione e la modifica del profilo utente;
4. itemService implementa i metodi che, sfruttando il servizio storageService, permettono la creazione e la cancellazione di un ToDo;
5. loginService implementa le funzionalità di login e logout di un utente.

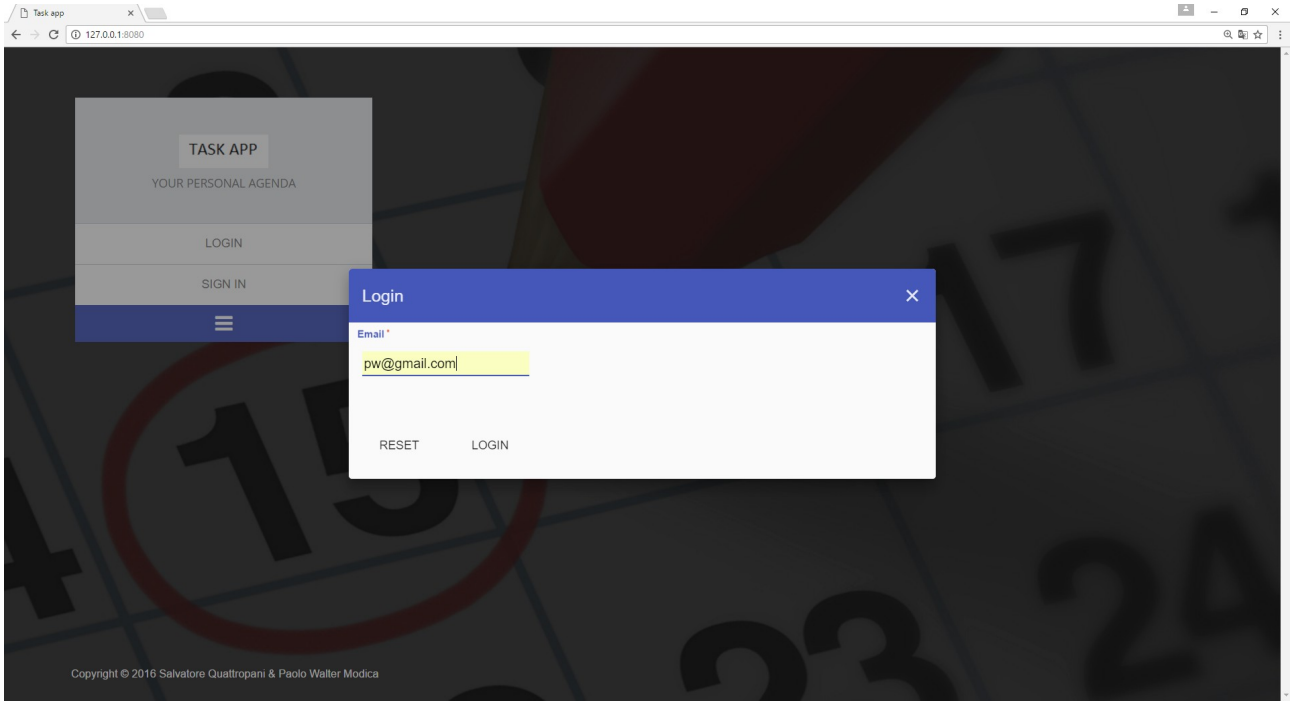
- Le direttive custom **customList** e **customListAll**, con il quale vengono istanziati nel DOM, e quindi nella View, gli oggetti relativi ai ToDo dell'utente, siano questi ancora da fare, fatti o in qualsiasi dei due stati.

Illustrazioni sul funzionamento dell'applicazione

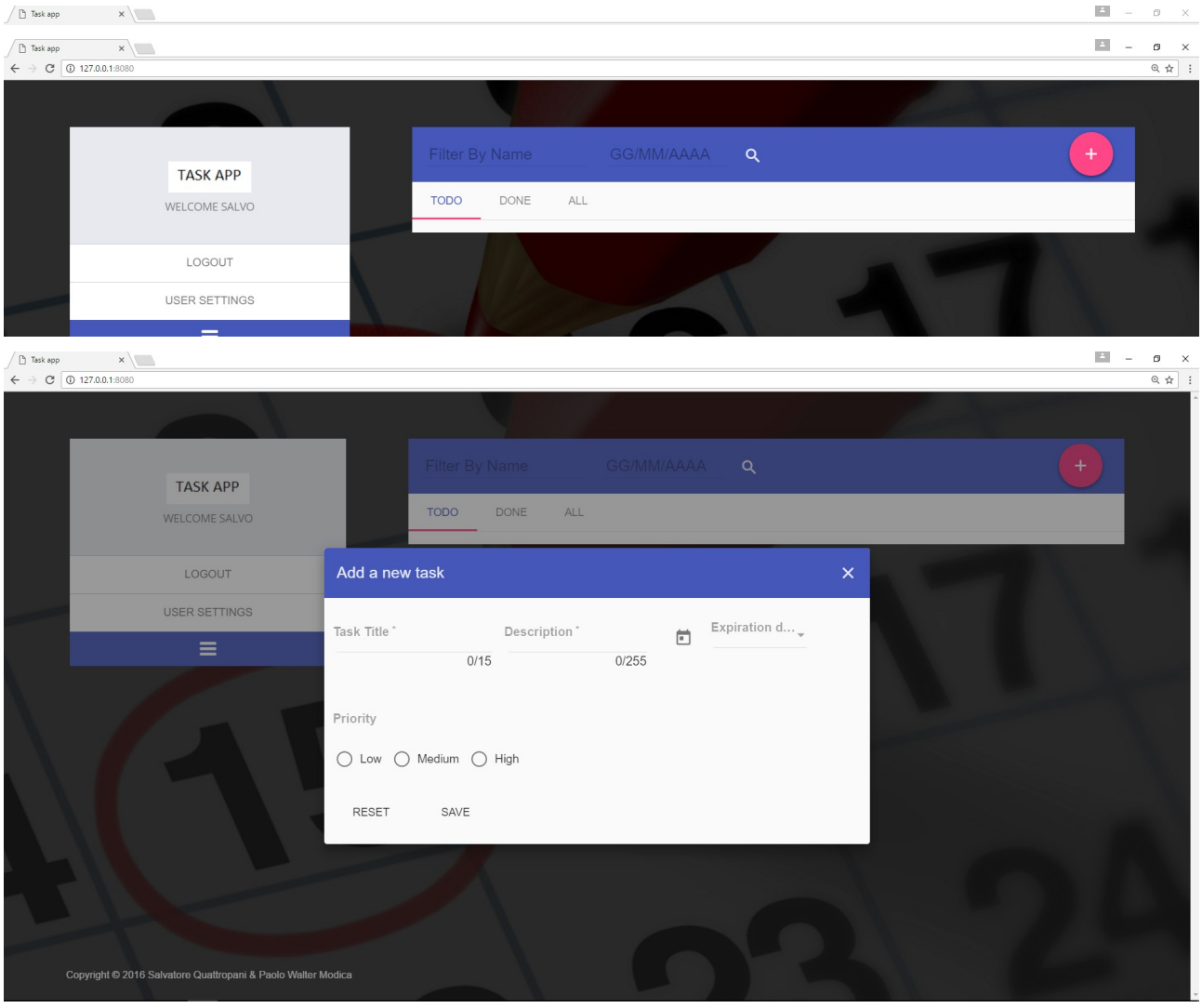
Qui di seguito vengono riportate alcune immagini dell'interfaccia grafica dell'applicazione durante l'uso ed il funzionamento della stessa.



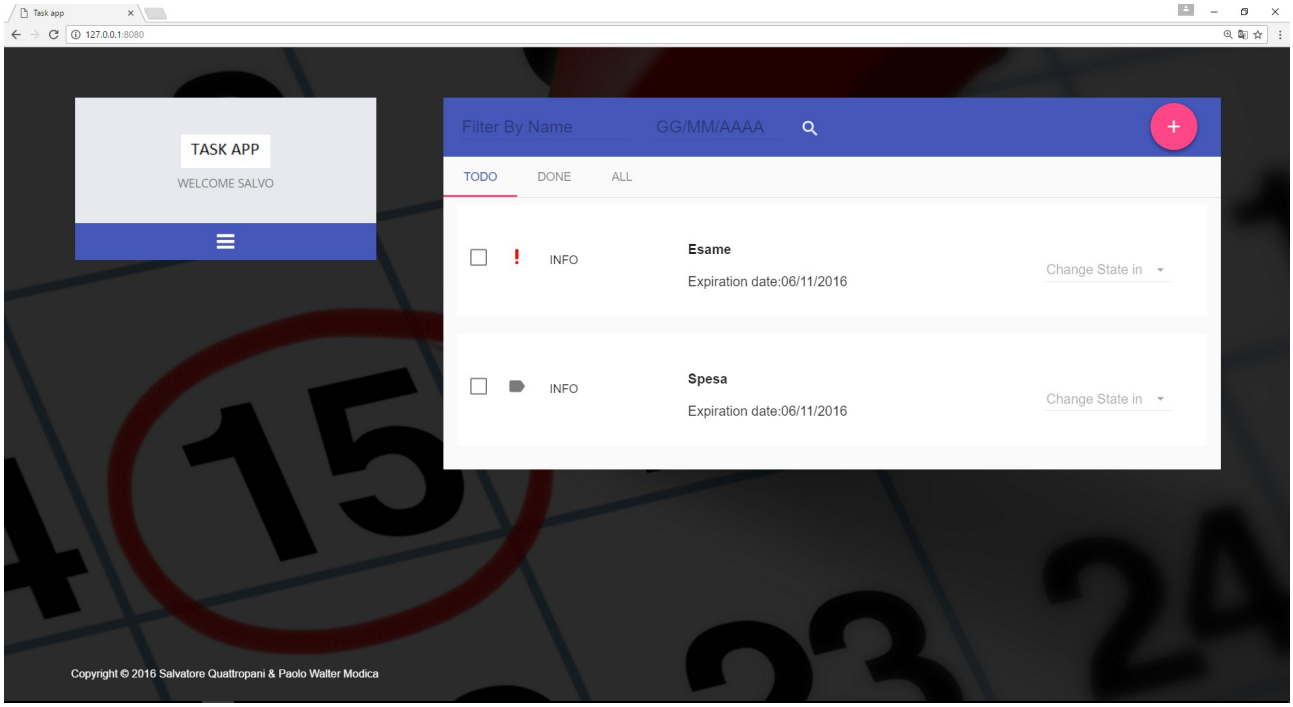
Registrazione di un nuovo profilo utente



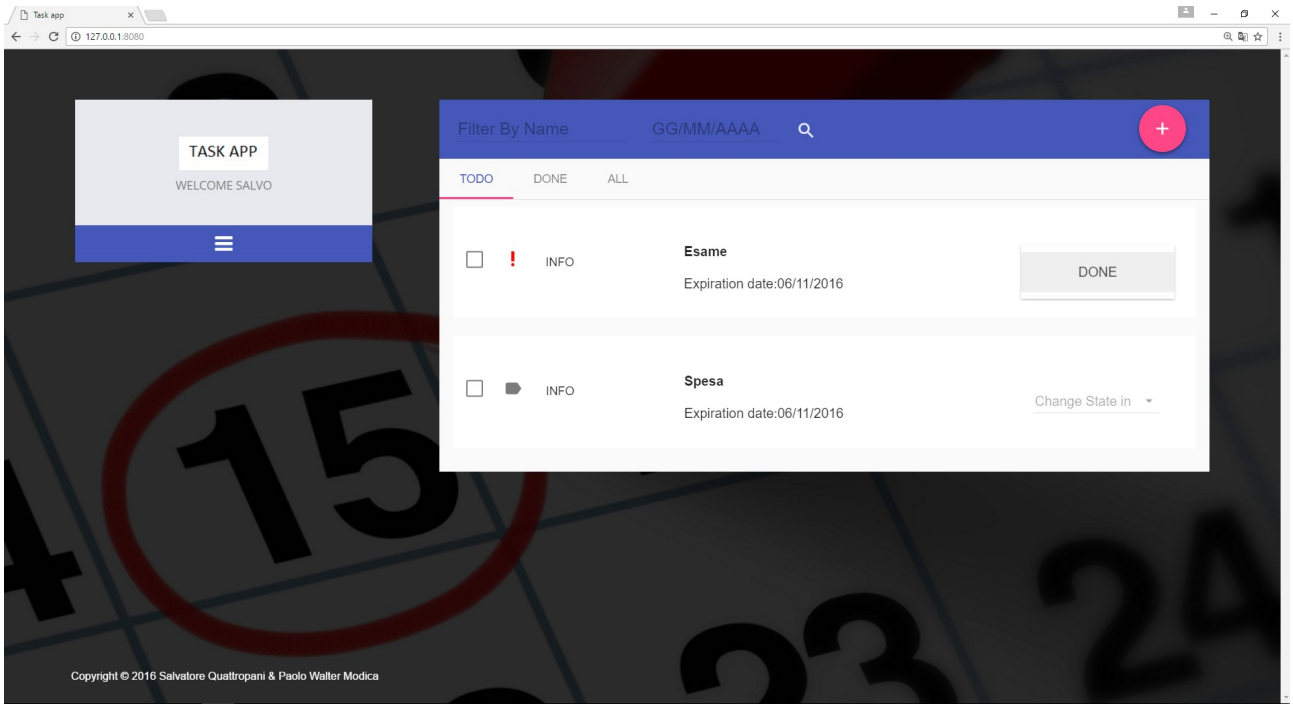
Fase di login



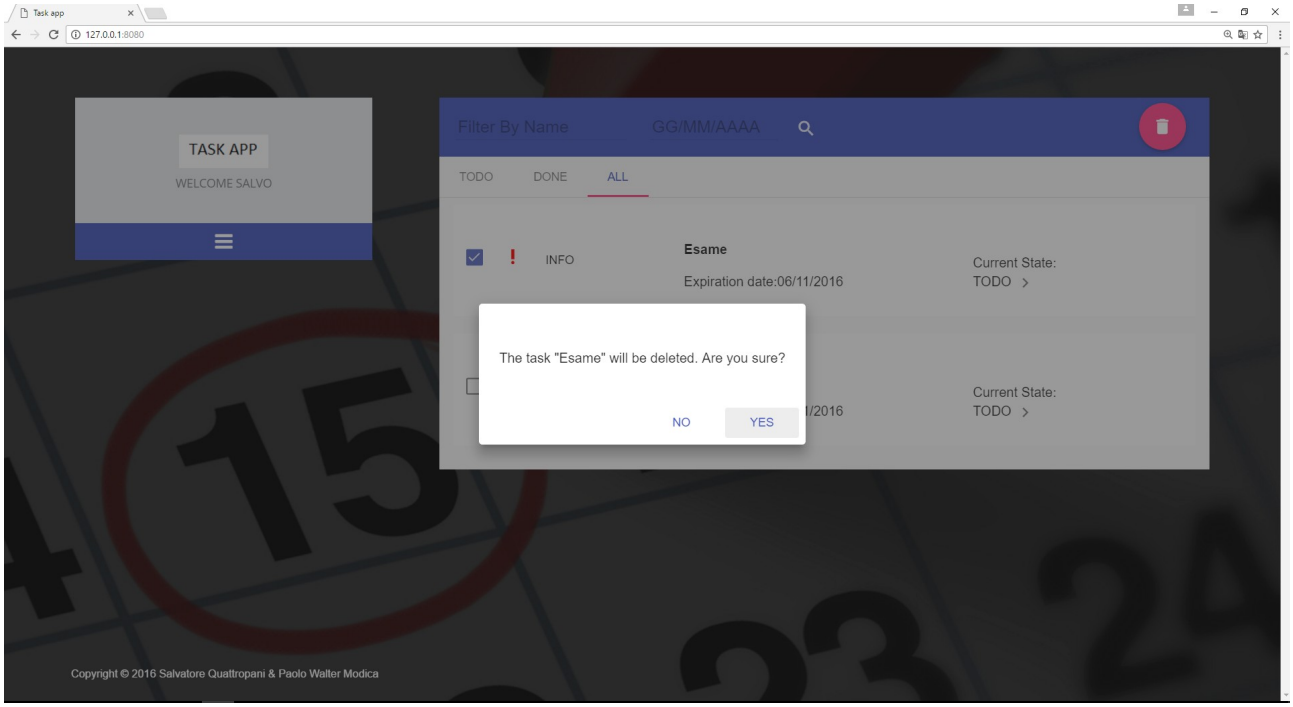
Inserimento di un nuovo impegno



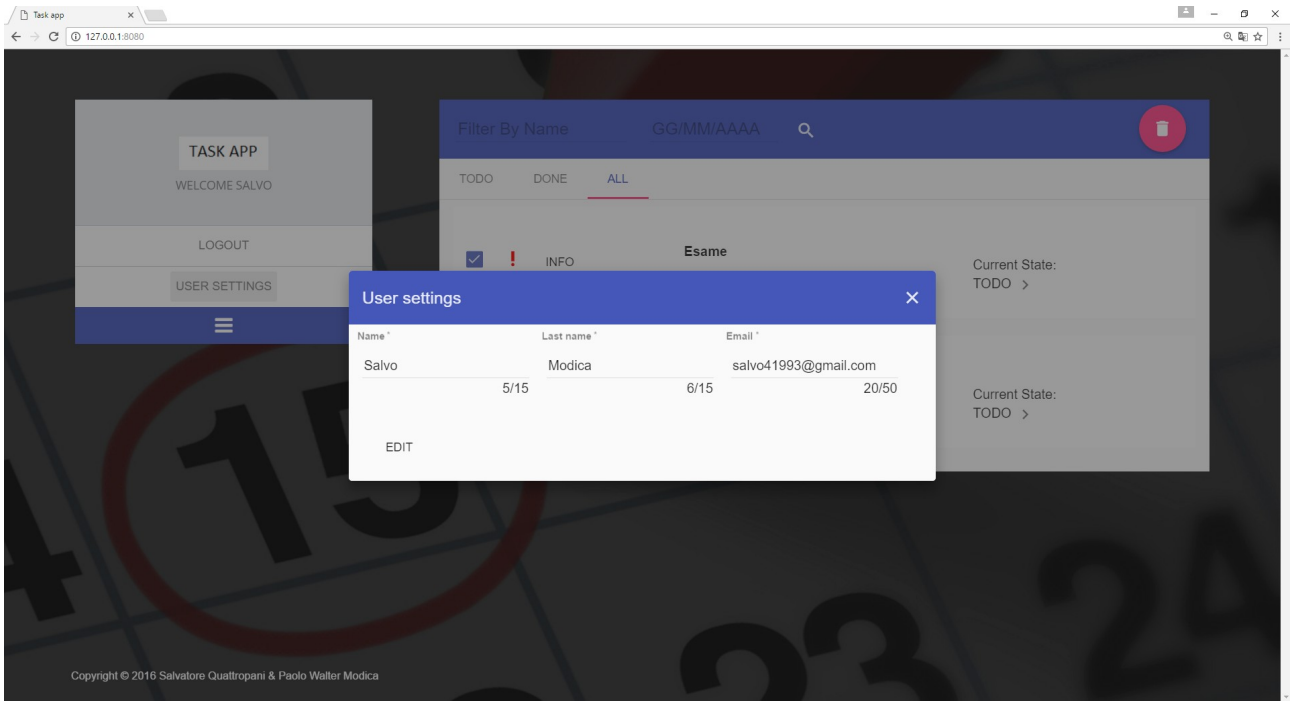
Interfaccia dell'applicazione con lista ToDo popolata



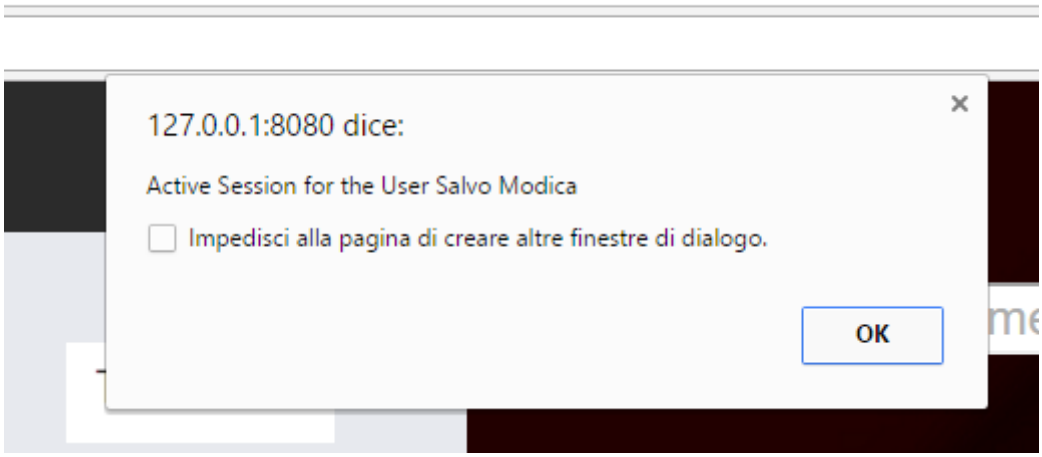
Cambiamento di stato del ToDo



Eliminazione di un ToDo



Modifica del profilo dell'utente



Notifica di sessione attiva presente sul browser