

SUPSI

DocTT — Document Tagging Tool

Studente/i

Cristian Spazio

Denys Vitali

Relatore

Daniele Puccinelli

Correlatore

Salvatore Vanini

Corso di laurea

Ingegneria Informatica

Modulo

M00009 - Progetto di diploma

Anno

2019

Data

2 maggio 2019

STUDENTSUPSI

Indice

1	Introduzione	5
1.1	Struttura	5
1.1.1	Home	5
1.1.2	Document Upload	6
1.1.3	Tree View	6
1.1.4	Tree Upload	7
1.2	Tecnologie	7
2	Funzionamento	9
2.1	Processo di tagging	9
2.1.1	Floating menu	9
2.1.2	Nested tags	10
2.1.3	Tag removal	10
3	Specifiche tecniche	11
3.1	JavaScript/TypeScript Environment	11
3.1.1	Angular	11
3.1.1.1	App structure	11
3.1.1.2	Routing	12
3.1.2	npm	12
3.1.3	Webpack	12
3.2	Compilazione e Test	12
3.2.1	Compilazione	12
3.2.1.1	npm	12
3.2.2	Test	12
3.2.2.1	Jest	12
3.2.2.2	npm	13

Elenco delle figure

1.1	Home section	5
1.2	Document Upload section	6
1.3	Tree View section	6
1.4	Tree Upload section	7
2.1	Floating Tagging Menu	9
2.2	Nested Tags Example	10
2.3	Tag Removal Example	10

Elenco delle tabelle

STUDENTSUPSI

Abstract

DocTT is a tagging tool which allows its users to tag text documents such as press conferences or pitches. The tags take form in coloured rectangles surrounding the text allowing a fast and easy visual recognition of a tag. Tags are defined through a pre-uploaded tree by the user. The files involved (document files and tree files) follow the XML standard and their parsing is managed by the tool.

Riassunto

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetur odio sem sed wisi.

Capitolo 1

Introduzione

1.1 Struttura

DocTT si divide in 4 sezioni ognuna adibita ad una funzione specifica:

- Home
- Document Upload
- Tree View
- Tree Upload

1.1.1 Home

La **Home** di *DocTT* è la sezione in cui viene presentata la lista dei documenti caricati, dalla quale è possibile accedere ai documenti singoli e quindi alla loro modifica.

The screenshot shows the 'Your Documents' section of the DocTT interface. It displays two documents:

- Document 1**: Created on 2019-04-02. Preview: Executives Michael Spencer - Microsoft Corp. Greetings, and welcome to the Microsoft Fiscal Michael Spencer - Microsoft Corp. Good. Buttons: VIEW, DELETE.
- Document 2**: Created on 2019-04-02. Preview: Executives Michael Spencer - Microsoft Corp. Greetings, and welcome to the Microsoft Fiscal Michael Spencer - Microsoft Corp. Good. Buttons: VIEW, DELETE.

At the bottom of the page, there is a footer with the following information:

- DocTT**: Open Source project by Cristian Spazio and Denys Vitali.
- Get In Touch**: DocTT on GitHub.
- Some Title**: Some placeholder text about the Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus ut fringilla neque. Nulla facilisi. Curabitur vitae convallis dolor. Vivamus tortor magna, euismod id vehicula sed, dignissim eu orci. Morbi et enim tortor. Aliquam libero nulla, blandit non interdum ac, fermentum fermentum urna. Aliquam luctus viverra pretium. Fusce eu pretium mauris.

Figura 1.1: Home section

1.1.2 Document Upload

La sezione **Document Upload** di *DocTT* è quella che permette l'upload dei documenti da taggare ed il loro salvataggio nello storage del browser.

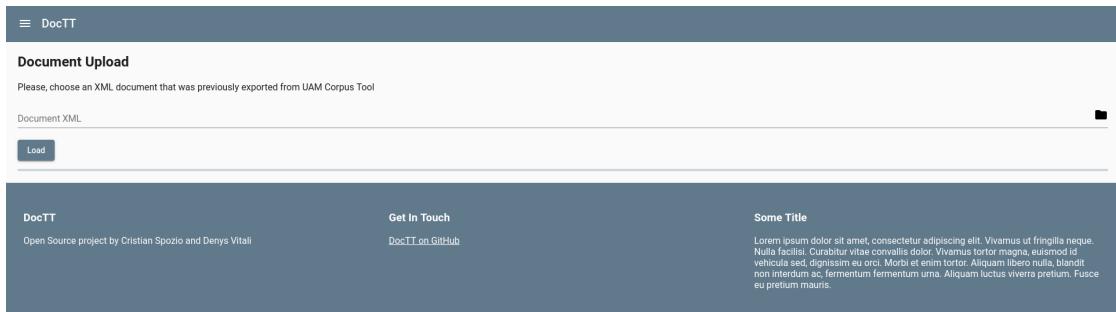


Figura 1.2: Document Upload section

1.1.3 Tree View

La sezione **Tree View** di *DocTT* è la parte che mostra l'albero di tagging attualmente in uso come un elenco puntato espandibile per ogni sottosezione presente.



Figura 1.3: Tree View section

1.1.4 Tree Upload

La sezione **Tree Upload** di *DocTT* è il componente che permette l'upload dell'albero di tagging. Offre inoltre un'anteprima dell'albero attualmente in uso (nel caso ci fosse), secondo la stessa visualizzazione della sezione **Tree View**.



Figura 1.4: Tree Upload section

1.2 Tecnologie

DocTT è stato sviluppato in **Angular**¹ secondo le direttive fornite dal relatore. È quindi stata divisa *app* nelle sottosezioni directives, models, modules, services e shared. I *componenti Angular* sono stati quindi divisi in 3 parti: struttura, stile e funzione rispettivamente secondo i linguaggi **HTML5**, **SCSS** e **TypeScript**.

¹ Angular : Framework per la creazione di UI in TypeScript

Capitolo 2

Funzionamento

2.1 Processo di tagging

Il processo di tagging può venire effettuato solo dopo il caricamento di un albero di tagging e dopo la selezione di un documento, operazioni eseguibili nelle apposite sezioni del tool.

2.1.1 Floating menu

La procedura effettiva di tagging viene effettuata selezionando del testo, al completamento di questa operazione apparirà un menu alla fine della selezione, contenente l'albero di tagging dal quale l'utente potrà andare a scegliere quale tag attribuire alla selezione.

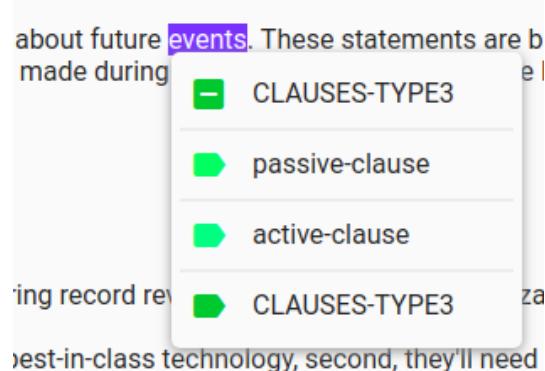


Figura 2.1: Floating Tagging Menu

2.1.2 Nested tags

Il tagging può essere effettuato a più livelli, quindi un tag (o più) possono essere contenuti in un altro tag (o più).

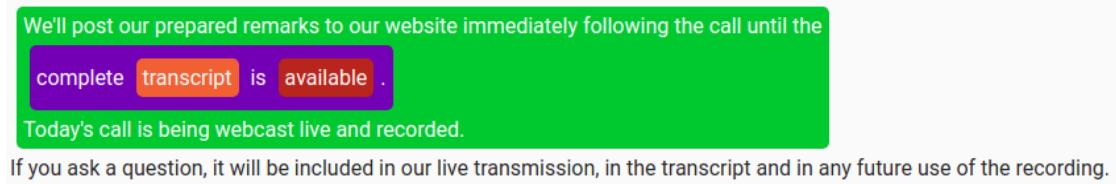


Figura 2.2: Nested Tags Example

2.1.3 Tag removal

I tag sono facilmente rimovibili semplicemente cliccando su di essi.

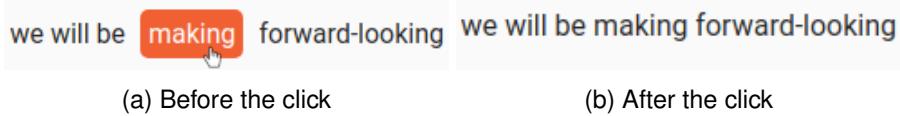


Figura 2.3: Tag Removal Example

Capitolo 3

Specifiche tecniche

Qui sotto verranno elencate le specifiche tecniche utilizzate durante lo sviluppo del tool.

3.1 JavaScript/TypeScript Environment

Durante lo sviluppo del tool il team si è trovato a confronto con l'ambiente di sviluppo di *JavaScript/TypeScript* e quindi con le rispettive tecnologie, qui elencate le più significative.

3.1.1 Angular

Come già citato in precedenza, **Angular** è un *web application framework* basato su *TypeScript*. Il concetto principale dietro *Angular* è quello di definire una struttura di base ed una gerarchia di componenti, che verranno poi inseriti all'interno del componente principale *app* come il riempimento di una cornice vuota.

3.1.1.1 App structure

Essendo *app* il componente principale la sua struttura è fondamentale. Noi abbiamo optato per suddividere il tutto, secondo convenzione di *Angular*, in 5 parti:

Directives Le *direttive Angular* sono strumenti che una volta definiti cambiano aspetto e/o comportamento di un elemento del DOM¹.

Models I *modelli Angular* sono templates che definiscono la struttura di classi o interfacce.

Components I *componenti Angular* sono l'elemento base per la costruzione di una UI² e sottostanno sempre ad una direttiva e sono spesso costruiti con dei templates. Sono definiti da tre segmenti: struttura, stile e funzione.

Services I *servizi Angular* sono dei componenti senza struttura né stile, utilizzati in tutto il complesso per le loro funzioni e le operazioni che offrono. Possono essere visti come delle classi statiche di utilità in **Java**.

¹DOM (Document Object Model): È una forma di rappresentazione di una pagina web come albero gerarchico

²UI: User Interface (Interfaccia Utente)

Modules I *moduli Angular* sono una raccolta di componenti, servizi, direttive, controller, filtri e informazioni sulla configurazione. Possono essere visti come le frazioni principali di una *app*.

3.1.1.2 Routing

Il routing è una parte fondamentale di una qualunque applicazione front-end che si possa definire tale. Grazie ad esso è possibile manovrare un'applicazione in esecuzione attraverso la richiesta di specifici path nell'URL. Questo permette di spostarsi attraverso l'applicazione senza continuare a passare da una pagina web all'altra, ma semplicemente andando a cambiare i contenuti della pagina attuale secondo le risorse richieste. Ciò comporta meno richieste al server ed un più rapido processo di caricamento e *switching* tra pagine/componenti.

3.1.2 npm

npm o *Node.js package manager* è un package manager per *JavaScript*. Esso consiste in un client da CLI³ che permette all'utente di utilizzare e/o distribuire moduli *JavaScript*.

3.1.3 Webpack

Webpack è un *module bundler JavaScript* open-source. Esso permette di definire *loaders*, *plugins*, ecc., utilizzato per un approccio modulare alle *web applications*.

3.2 Compilazione e Test

3.2.1 Compilazione

3.2.1.1 npm

npm offre anche la possibilità di definire degli *scripts* utilizzabili nel suo client per utilizzi vari, nel nostro caso la compilazione. Difatti il nostro progetto è compilabile tramite l'utilizzo del comando **npm run build:dev** sul client *npm*.

3.2.2 Test

3.2.2.1 Jest

Jest è un framework di Testing *JavaScript* basato sulla semplicità di configurazione ed utilizzo. Permette di definire delle *test suites*, delle collezioni di test che vadano a verificare il corretto funzionamento e comportamento delle funzionalità del tool.

³CLI (Command Line Interface): console

3.2.2.2 npm

Anche qui *npm* si rende utile, potendo appunto definire uno *script* di testing che lancerà le varie *test suites* per il controllo tramite il comando **npm test**.

Bibliografia