APPLICAZIONI INTERNET

Esercitazione n.4 (gio 14/05)

Realizzazione UI web con componenti Angular Material

-- consegna entro dom 31/5 --

Si richiede di sviluppare una applicazione front-end, basata su Angular, per la gestione di gruppi di studenti all'interno di corsi universitari.

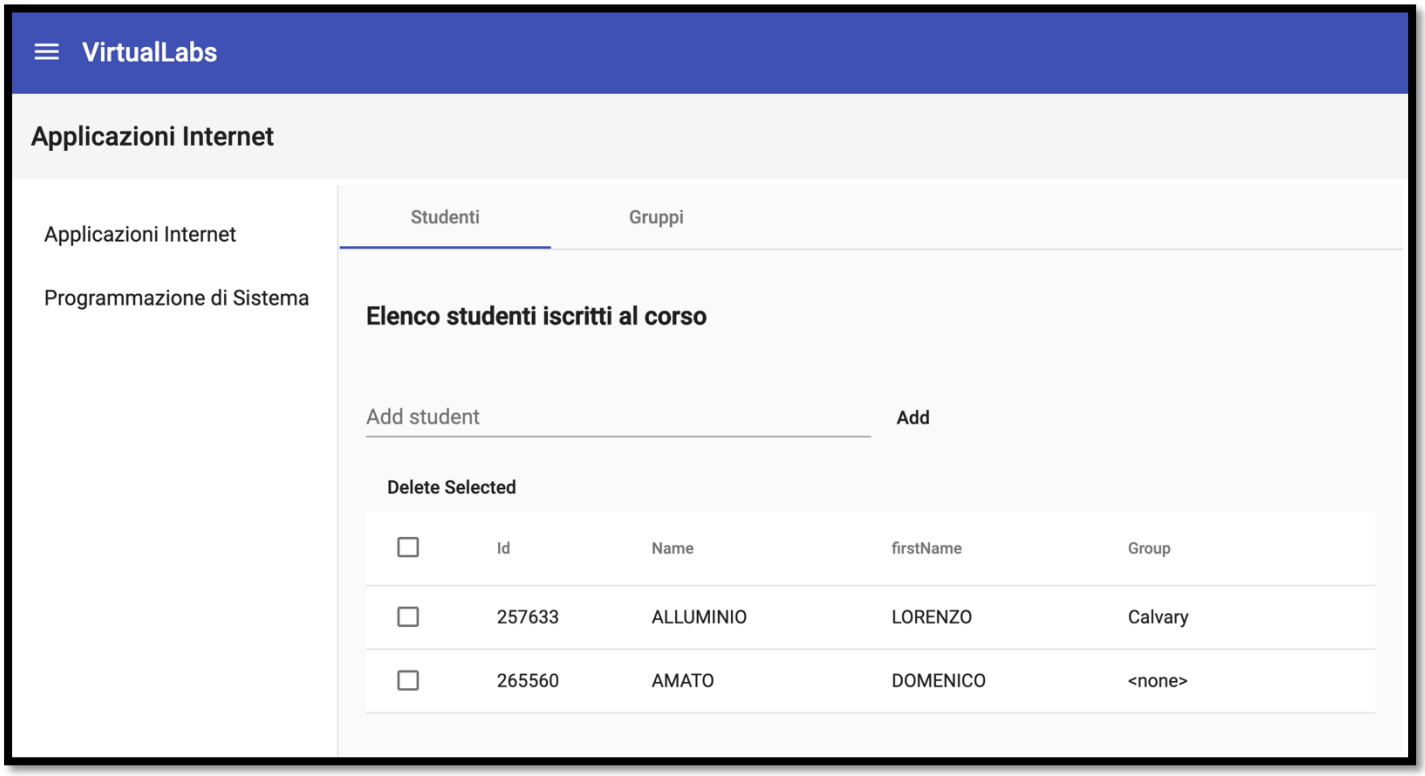

Figura 1: Vista applicazione.

0. Si installi Angular (eventualmente Nodejs), si crei il progetto e si verifichi che venga correttamente aperto ed eseguito dal browser (si scelgano le impostazioni di default quando richieste)
   - npm install @angular/cli
   - ng new ai20-lab04
   - cd ai20-lab04
   - ng add @angular/material
   - ng build –prod

    - Si verifichi che viene prodotto il codice per l'hosting nella cartella dist
- ng serve –open

Mantenere un terminale aperto con il comando ng serve in esecuzione per poter vedere in tempo reale le modifiche al codice riflesse nella applicazione live.

Per quanto segue quando non indicato si faccia riferimento alla documentazione disponibile sul sito di Angular [1] e di Angular Material [2] o MDN [3].

CAVEAT. In questa esercitazione ci concentreremo in particolare sull'utilizzo del template e del component, al momento ignoreremo i service e il router. Questi elementi sono esssenziali in una applicazione vera e propria e non si deve prescindere da essi, ma li approfondiremo successivamente, sicuramente, ri-adattando e ri-modulando il codice prodotto in questo laboratorio.

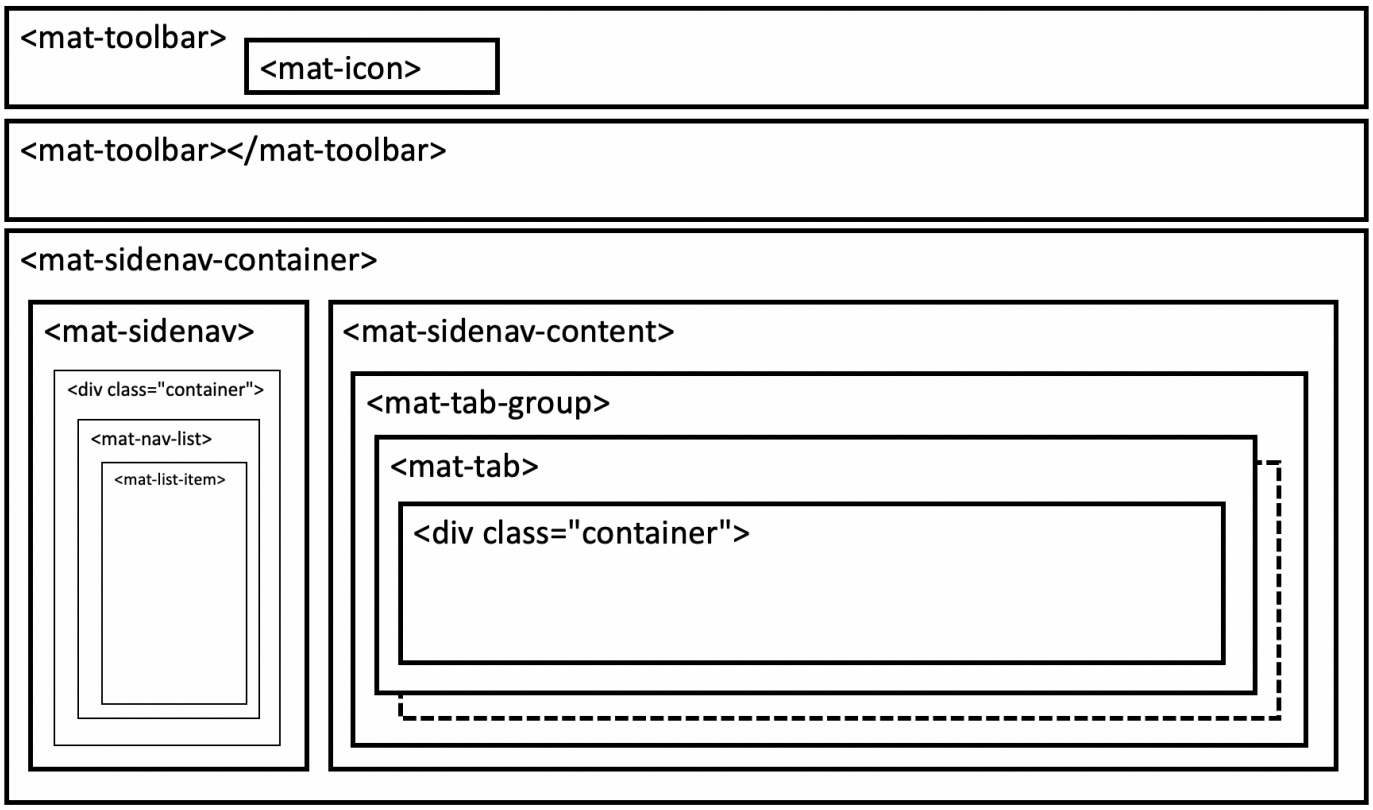

Figura 2: Layout applicazione.

Si inizi a strutturare il layout dell'applicazione rimuovendo il contenuto del file app.component.html e sostituendolo con quanto di seguito indicato per ottenere un aspetto simile a quanto in Figura 1.

1. Realizzare il layout della pagina utilizzando i componenti di Angular UI indicati nella Figura 2
    - Per ogni componente è necessario includere il relativo HTML nella pagina (file HTML)
    - Importarne il Modulo all'interno dell'applicazione Angular inserendo nel file app.module.ts quanto segue. Si faccia sempre riferimento alle indicazioni e agli esempi forniti nelle tab "API" ed "EXAMPLES" delle corrispondenti pagine sul sito di Angular Material (es. https://material.angular.io/components/toolbar/overview)
        - L'import del modulo JS:

        - e.g. import { MatToolbarModule } from '@angular/material/toolbar';
    - L'import del modulo Angular nell'array @NgModule(imports: [
        - e.g. imports: [MatToolbarModule,]
- Si aggiunga la seguente specifica CSS nel file style.css per utilizzare il 100% della pagina in verticale che altrimenti verrebbe "tagliata". Ed eventuali altre proprietà che ritenete opportune per padding, margini o allineamento degli elementi.
    - mat-sidenav-container{ margin: 20px; height: 100%; }
- Se collocate nel codice HTML più tag <mat-tab> uno dopo l'altro con attributo label="Studenti", label="Gruppi", e così via dovreste vedere che è già possibile alternare la visualizzazione delle tab cliccando sull'etichetta (label). Inserite al loro interno del contenuto a scelta per differenziarli.

Automazione della barra di navigazione laterale, sidenav. La barra di navigazione laterale può essere utilizzata per fornire un menu di navigazione attraverso le varie viste dell'applicazione. In questo momento ne realizzeremo solo una (perché non usiamo il router), ma in seguito verrà estesa includendo altri elementi. La barra di navigazione può essere gestita a "scomparsa" sfruttando i metodi .open(), .close(), .toggle() che essa stessa esporta. La struttura è quella indicata in Figura 2 che avete già realizzato al passo precedente: un mat-sidenav-container con all'interno due figli/fratelli mat-sidenav (che è la parte che contiene il menù e viene fatta scorrere) e mat-sidenav-content che conterrà le differenti viste/route della nostra applicazione.

2. Per realizzare l'automazione occorre definire lo stato iniziale della sidenav e richiamare una funzione del componente, ad esempio, quando viene cliccata l'icona menu in alto a sinistra (hamburger button). Si faccia riferimento anche alle indicazioni sul sito di Angular Material https://material.angular.io/components/sidenav/overview (sempre tutte e tre le tab: overview, api, examples).
    - Si aggiungano al tag mat-sidenav gli attributi mode, opened, e position in modo da visualizzarla come in Figura 1.
    - Si aggiunga nella mat-icon il binding con l'evento (click) che effettui la chiamata alla funzione toggleForMenuClick del componente
    - Si implementi nel componente app.component.ts il codice per far apparire e sparire la sidenav
        - ATTENZIONE: per poter agire sulla sidenav (del template) dall'interno del componente è necessario ottenerne un riferimento (template reference) utilizzando in quest'ultimo l'annotazione @ViewChild. Oltre alla guida di Angular potete fare riferimento anche a questa guida su Angular University [4]

La vista corrente ha come obiettivo quello di visualizzare l'elenco degli studenti di un corso all'interno di una tabella offrendo altresì la possibilità di editarne l'elenco. Per la visualizzazione dell'elenco degli studenti occorre definire il modello dei dati e uno store (store che noi realizzeremo in modo fittizio ed estenderemo nel laboratorio successivo).

3. Il modello del dato studente può essere realizzato nel file student.model.ts e consiste in una classe Typescript denominata Student con un costruttore e le proprietà id, name e firstName definite nei laboratori precedenti.
    - Realizzare la classe Student ed esportarla/importarla nell'AppComponent

- Creare un array di studenti (già) iscritti al corso per avere qualcosa da visualizzare nella tabella

La costruzione della tabella avviene utilizzando il componente Angular Material Table (mat-table) descritto all'indirizzo https://material.angular.io/components/table/overview e la forma più semplice di sorgente dati, un array (si invitano gli studenti a considerare invece l'utilizzo di stream di Observable per i loro progetti, come vedremo nelle lezioni successive e soprattutto nella lezione sui Form).

4. La realizzazione di una tabella è strettamente legata alla struttura degli oggetti che la andranno a popolare. Si invita a partire o dalle slides fornite dal docente o da un esempio fornito dalla pagina di angular material.
    - Occorre dichiarare il template HTML per ogni colonna che si vuole realizzare. L'attributo matColumnDef di un ng-container definisce il nome della colonna (necessario quando si dovrà scegliere quali colonne visualizzare). All'interno dell'ng-container viene definito anche il template per i tag TH e TD. Per il tag TD occorre instanziare una variabile corrispondente all'oggetto di ciascuna riga e definire una espressione per visualizzarne il contenuto.
    - Occorre dichiarare il template HTML per ogni riga attraverso i tag mat-header-row e mat-row. Prestare attenzione alla necessità di definire ed utilizzare nel tag un array di stringhe columnsToDisplay per indicare i nomi delle colonne da visualizzare.
    - Effettuare il binding tra l'attributo datasource della mat-table e l'oggetto dell'AppComponent che ne rappresenta la sorgente dati (Array di Student) da visualizzare (e creare, sempre nel componente, l'array colsToDisplay con il nome e l'ordine delle colonne da visualizzare.

La manipolazione dei dati della tabella può avvenire, da parte dell'utente, utilizzando una colonna di checkbox per selezionare le righe da cancellare.

5. Aggiungere alla tabella una colonna di checkbox costituita da una checkbox _globale_ nell'intestazione, che serve a selezionare/deselezionare tutte le righe, ed una checkbox individuale per ogni riga. L'informazione sulle checkbox viene poi utilizzata per individuare gli elementi da rimuovere.
    - Si utilizzi il seguente template base per la colonna e si aggiunga il suo nome nell'array colsToDisplay

```
<ng-container matColumnDef="select">
  <th mat-header-cell *matHeaderCellDef><mat-checkbox></mat-checkbox></th>
  <td mat-cell *matCellDef="let row"><mat-checkbox></mat-checkbox></td>
</ng-container>
```

    - Per le checkbox di ogni riga si realizzi un funzione che intercetti l'evento (change), riceva l'evento e il valore della riga, e ne memorizzi lo sato di selezione o deselezione in una apposita struttura dati.
    - Si rifletta quindi, tramite il binding con l'attibuto [checked] della checkbox, lo stato di selezione o deselezione della riga (anche in questo caso è utile definire e richiamare un metodo del component che, ricevuto il valore della riga, ne verifichi lo stato nella struttura dati.
    - ATTENZIONE: Estendere la funzionalità per (change) e [checked] anche per la checkbox "master" dell'intestazione che però deve avere tre stati: a) nessuna checkbox selezionata, b)

qualche checkbox selezionata, c) tutte le checkbox selezionate. Quando viene checkkata ed è nello stato a) o b) passa allo stato c), viceversa se è nello stato c) passa allo stato a). Questa "master" checkbox passa dallo stato intermedio b quando tramite l'azione sulle checkbox delle righe non tutte le checkbox sono nello stesso stato. N.B. Lo stato a) corrisponde a checked=true, lo stato c) a checked=false, lo stato b) a indeterminate=true (vedi [5]).

- Predisporre un bottone che, se premuto, cancella dall'elenco degli studenti del corso tutti gli studenti corrispondenti alle linee selezionate. N.B. La tabella non viene aggiornata automaticamente quando si modifica l'array dei suoi dati, le modifiche vanno applicate SOSTITUENDO l'array associato alla sorgente dati (quando cambia il puntatore all'array dei dati, questo scatena un refresh che aggiorna la tabella), come se fosse di per sé immutabile (*immutable pattern*).

L'inserimento di un nuovo studente tra gli studenti del corso deve avvenire utilizzando un campo input testuale associato ad una lista di "autocompletion" all'interno della quale sia possibile ricercare e selezionare il nome dello studente (e di conseguenza l'oggetto corrispondente allo stesso con le altre informazioni utili per la tabella). L'inserimento vero e proprio avverrà con un apposito bottone Add.

6. La funzionalità di autocomplete è per molti versi simile al campo select di un form che al suo interno contiene una lista di options che, associate ai valori di un array di dati, possono essere selezionate (in questo caso limitatamente a una soltanto) per indicare il valore da utilizzare.
    - Creare un campo input testuale (o se si vuole type="search") utilizzando il componente Angular Material Input "arricchendolo" correttamente delle informazioni per guidare l'utente nel suo utilizza: label e placeholder. Verificare come la loro apparenza è gestita da Angular Material.
    - Associare al campo di input un componente Angular Material Autcomplete https://material.angular.io/components/autocomplete/overview ciò avviene, come indicato dal sito esportando l'istanza del pannello autocomplete in una variabile locale del template (e.g. #auto) e facendone il binding con la proprietà matAutocomplete del campo di input.
    - Si crei uno "pseudo" DB (basta un array) di studenti e si popoli l'autocomplete di tante options (<mat-option>) quanti sono gli studenti del DB, si associ una rappresentazione testuale dello studente al campo del tipo: <name> <firstName> (<id>). N.B. Si fa notare che associando una funzione all'attributo displayWith dell'autocomplete è possibile utilizzarla per creare in automatico la rappresentazione testuale della lista di opzioni. In altre parole, come indica la guida "Setting separate control and display values" è possibile continuare a passare oggetti "Student" ai campi option, ma averli automaticamente visualizzati (in stringhe) tramite la conversione predisposta dalla funzione "displayWith". (ciò diventa utile quando si deve recuperare l'oggetto dell'opzione selezionata per l'inserimento nella tabella).
    - Si implementi anche un filtro "custom" che attraverso il binding all'evento di scrittura nel campo di input (keyup) filtri l'array associato alla lista di opzioni mantenendo soltanto quegli studenti che contengono nel proprio nome la stringa digitata (questo per rendere più comoda la ricerca).

7. L'inserimento dello studente selezionato nella tabella avviene aggiungendo all'array di studenti visualizzato nella tabella l'oggetto che corrisponde alla selezione effettuata.

- All'avvenuta selezione di un elemento dalla lista delle opzioni, il componente AutoComplete, emette l'evento optionSelected si agganci questo evento e si associ ad esso un metodo del componente che dall'evento stesso recupera l'oggetto selezionato e ne mantiene memoria.
- Si inserisca un bottone add e, alla pressione del bottone add, si recuperi tale valore e lo si aggiunga alla tabella. ATTENZIONE: si eviti di inserire più volte lo stesso studente!

Si introducano altre due funzionalità utili nella tabella: l'ordinamento, come indicato nelle slides del corso (06.Angular-Material.pdf) e, opzionalmente, la paginazione (si veda il sito all'indirizzo https://material.angular.io/components/paginator/overview).

Indicazioni per la consegna

L'elaborato deve essere consegnato come archivio ZIP con nome come segue <nome-gruppo>-lab04-v<N>.zip, dove <nome-gruppo> è il nome, sperabilmente univoco, del gruppo e N è la versione dell'elaborato consegnato. Nota: non so se si possano eliminare consegne precedenti, nel caso non fosse possibile la prima avrà N=1 e le altre avranno valori di N interi successivi. Nota: non si faccia l'upload della cartella *node_modules* con le librerie!!!

Predisporre un file testuale denominato come README.TXT con:

- lista dei nomi/matricole dei componenti del gruppo
- istruzioni per l'avvio (dovrebbe essere sufficiente fare npm install, npm start ed aprire il browser su localhost:4200)
- indicazioni per la verifica delle funzionalità dell'applicazione sviluppata
    - dove cliccare, cosa scrivere, cosa aspettarsi

Si fornisca anche uno (o più) screenshot dell'interfaccia realizzata.

Riferimenti

[1] Angular – https://angular.io

[2] Angular Material – https://material.angular.io

[3] Mozilla Developer Network - https://developer.mozilla.org

[4] Angular @ViewChild: In-Depth - https://blog.angular-university.io/angular-viewchild/

[5] MDN: Indeterminate state checkboxes - https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/checkbox