Prácticas

Requisitos

- Visual Studio Code
- NODE
  - node –v
- Descargar seed de GitHub
  - https://github.com/anxofole/TodoApp

- Ejecutar
  - npm install -g http-server@0.9.0
  - Npm install
  - Npm start

Práctica 1

Controllers, Scopes y Directivas

Enunciado

- Añadir módulo (app.js) y controlador (TodoCtrl.js)
- El controlador mantiene una lista de Todo’s en memoria
  - Pintarlos en la lista
  - Todo: { text: ‘Task 1’, done: true|false }
- Añadir un nuevo Todo al pulsar en el botón ***add***
  - Sólo activo si hay texto en el input
- Marcar como terminado un Todo al macar el check
- Definir el contador de Todo’s en la lista
- Eliminar de la lista los Todo’s terminados con el botón ***remove done***
  - Sólo activo si hay Todo’s terminados
- Definir el contador de Todo’s pendientes en la lista

Práctica 2

Componentes

Enunciado

- Convierte el controlador anterior en un componente.
  - list.component.js
  - list.component.html

Práctica 3

Servicios

Enunciado

- Añadir servicio todo.service.js para centralizar la gestión de Todo’s
  - El servicio mantiene una lista en memoria de Todo’s
    - getAll
    - add
    - removeDoneTodos
  - Implementarlo como **Factory** y como **Service**

Práctica 4

Routing

Enunciado

- Añadir getById al servicio todo.service.js
- Añadir routing
  - Crear un componente que muestre el detalle del todo
  - Añadir un estado para el componente del detalle
  - Añadir un estado para el componente del detalle
  - Navegar al componente de detalle si clicko en el nombre del todo
  - Crear un nuevo botón que nos permita volver atrás a la lista.

Práctica 5

Comunicación con servidor y promesas

Enunciado

- Modificar el todo.service.js para que no persista localmente el estado y lo recupere llamando al API.

Práctica 6

Formularios

Enunciado

- Ir al nuevo estado /register
- Crear formulario de registro
  - Nombre
  - Select con países
  - Nombre de usuario
    - Mostrar error de que es requerido si no se rellena
  - Email
    - Validar formato
    - Mostrar error si no tiene un formato válido
    - Mostrar error de que es requerido si no se rellena
  - Contraseña
    - Cumplir patrón (Mayúsculas, minúsculas y numeros, 8 dígitos)
    - Mostrar error de que es requerido si no se rellena

Práctica 7

Testing

Enunciado

- Hacer test unitarios de detailComponent
  - Cuando se inicia el componente debería haber llamado a todoService.getById()
  - Cuando se inicia el componente debería recuperar el $stateParams correcto.

- Hacer test unitarios de todoService.
  - Cuando el servicio llama a getById debería llamar a recuperar por id del servidor.
  - PISTA: Usar $**httpBackend**

Práctica 8

Custom directives

Enunciado

- Añade otro campo al registro para el dni
- Crea una directiva custom llamada dniValidator que compruebe si el dni introducido es válido.
  - Si no lo es, el input deberá mostrar un error.
  - **PISTA:**
    - **DNI PATTERN = /^\d{8}[a-zA-Z]$/**