Laboratorios

Ejercicio 01

Creamos un proyecto Angular, al cual le vamos a definir los siguientes componentes, que van a ser mostrados desde el componente principal (app.component.html):

- driven.component:
  - HTML: Crear un formulario del tipo Template-driven con tres campos:
    - nombre: tipo texto, requerido
    - apellido: tipo texto
    - email: tipo email
  - Clase: crear un método que capture el submit del formulario y muestre en el html los valores enviados.
- validar-driven.component:
  - HTML: Crear un formulario Template-driven con tres campos:
    - nombre: tipo texto, requerido, asociado a un objeto definido por la clase a través del ngModel, requerido.
    - director: tipo texto, asociado a un objeto definido por la clase a través del ngModel.
    - genero: campo select, asociado a un objeto definido por la clase a través del ngModel. Los options de este combo van a ser establecidos por un array definido por la clase. Por lo tanto la generación de cada option del combo select se va a hacer a través de la directiva *ngFor
    - A través de la propiedad hidden, se va a mostrar los datos enviados una vez que se presione el botón que de lugar al evento submit.
  - Clase: Instanciar una clase entidad llamada Pelicula, que va a ser utilizada como enlace para los campos del formulario.
  - crear un método que capture el evento submit de la vista y que establezca como true una propiedad de la clase que me sirva como flag para mostrar los datos enviados.
- reactive.component:
  - HTML: Crear un formulario Reactivo que conste de los siguientes campos:
    - nombre: tipo texto (No olvidar el formControlName)
    - apellido: tipo texto
  - Clase: Instanciar un objeto del tipo FromGroup definiendo en su constructor los FormControl que corresponden al formulario (nombre,apellido). Definir un método que capture los valores enviados a través del formulario y lo muestre en pantalla.
- validar-reactive.component:
  - HTML:
    - Crear un formulario Reactivo con los siguientes campos:
      - nombre: tipo texto,
      - apellido: tipo texto,
      - email: tipo email.
      - Si el estado del formulario es inválido, esconder el botón del submit (La directiva *ngIf podría ser útil en este caso)
  - Clase: Instanciar un objeto del tipo FormGroup definiendo en su constructor los FormControl que corresponden al formulario (nombre,apellido,email). Crear un método que capture el submit del formulario y muestre los valores enviados en pantalla. Las validaciones que vamos a hacer en cada campo son:
    - nombre: requerido, minLength=5, maxLength=30
    - apellido: requerido, maxLength=30
    - email: requerido, tipo email.

- custom-validator.component:
    - HTML: Crear un formulario Reactivo con los siguientes campos:
        - nombre: tipo texto
        - apellido: tipo texto
        - telefono: tipo texto
    - Debemos crear en el template un mensaje de error que solo se mostrará cuando el teléfono ingresado tenga un formato inválido.
    - Función validadorTelefono: creamos en la misma carpeta donde está definido el componente un archivo typescript con una función llamada validadorTelefono que devolverá una función del tipo ValidatorFn. Esta función validará una expresión regular en relación al valor ingresado en el campo y determinará su validez.
    - Clase: Instanciar un objeto del tipo FormGroup definiendo en su constructor los FormControl que corresponden al formulario (nombre,apellido,telefono). Crear un método que capture el submit del formulario y muestre los valores enviados en pantalla. Las validaciones que vamos a hacer en cada campo son:
        - nombre: required, minLength= 4
        - apellido: required,
        - telefono: required, validadorTelefono (A esta función le vamos a pasar como parámetro la expresión /^[456]{1}[0-9]{9}$/ para que el usuario deba ingresar siempre el número 456 y luego los nueve números restantes.