Desenvolupament Web amb

Angular – (MEAN stack 2)

Formularis

Generalitat de Catalunya

Raúl García Lepe raulgl@gmail.com

Formularis

- Els formularis Angular són més complexos que els HTML però ens permet nivells molt interessants de validació de dades, comunicació a temps real, etc.

- Els formularis fan servir tot el que hem vist fins ara més alguns tipus de notacions especials que ens permetran conectar les vistes amb la lógica per a enviar i rebre informació:
  - Doble binding,
  - Directives (ngModel)
  - Decoradors (@input, @output),...

- Els formularis ens serveixen per a treballar la separació de capes, el desacoblament de la lògica de les vistes i la comunicació bidireccional entre les capes.

Formularis

- Els formularis d'Angular no requereixen d'un action per a indicar a on s'envia la información.
- Per a l'estructura del formulari utilitzarem la notació d'Angular per a identificar els camps (#nom).
- Per a mostrar informació del component a la vista utilitzem el binding de propietats (notació [propietat]='').
- Per a enviar informació de la vista al component utilitzem el binding de events i s'envien els objectes html complerts.
- Per a mostrar els resultats utilitzem la interpolació de variables *({{variable}})*
  - Fem servir les directives estructurals (*ngIf) amb <ng-container> i <ng-templates>

Exercici 1

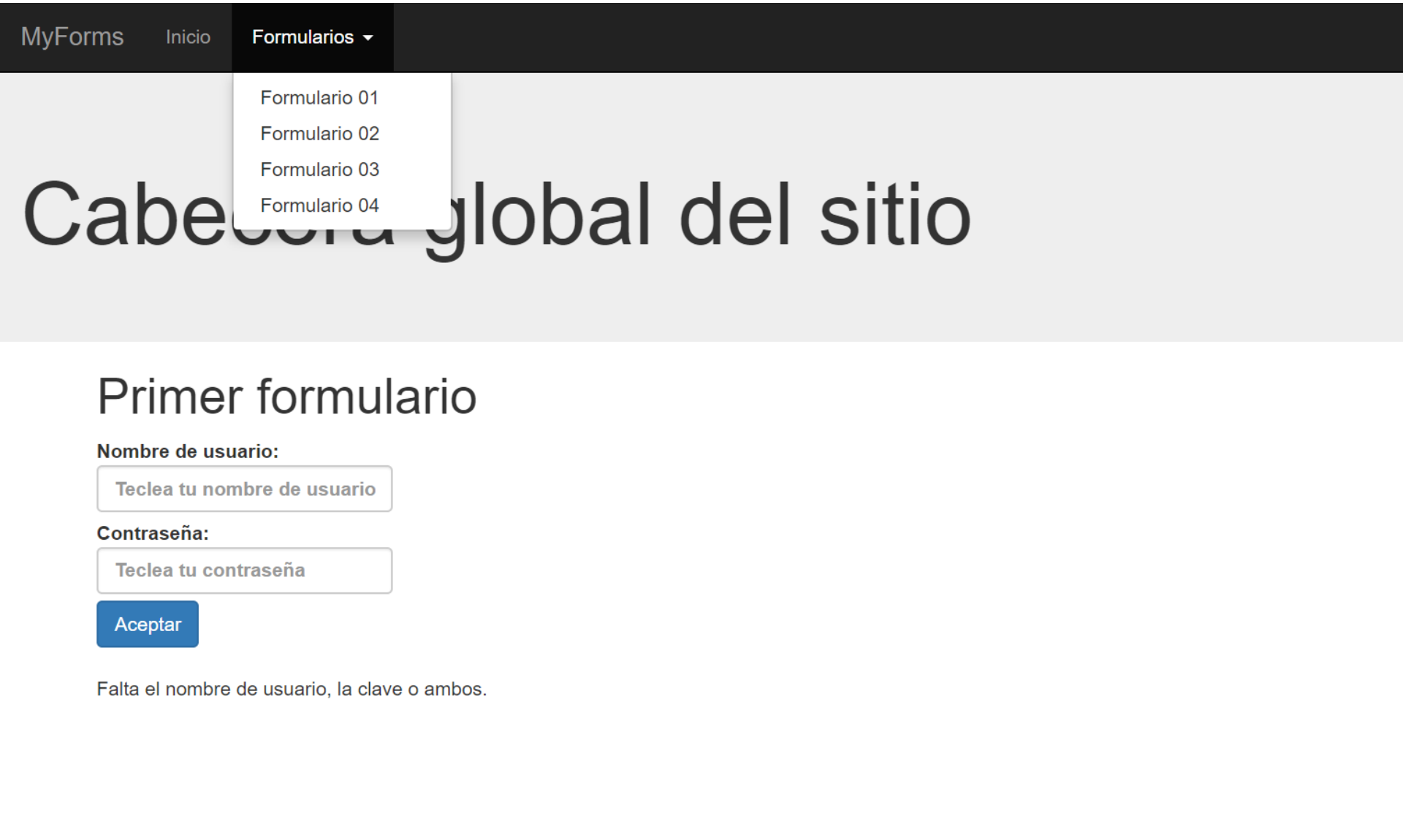
- Partim d'una aplicació base amb dos mòduls: Commons i MyForms.
  - CommonModule, sense routerModule propi, inclourà els components comuns de l'aplicació:
    - HeaderComponent
    - NavbarComponent
    - HomeComponent
    - NotFoundComponent
  - MyFormModule es carregarà en mode *Lazy Load* i contindrà un component:
    - Form01Component
- Afegir 3 llibreries externes: jquery, bootstrap i moment
- El formulari contindrà un camp usuari i un camp contrasenya. Es validarà que no hi hagi cap dels camps buits.
- Es mostrarà l'usuari i la contrasenya a la mateixa pantalla.

Exercici 1(cont)

- Estructura del formulari:

```
<h1>Primer formulario</h1>
<form>
  <label>Nombre de usuario:
    <input type='text' class='form-control' #usernameField [placeholder]='placeholders.username'>
  </label>
  <br>
  <label>Contraseña:
    <input type='password' class='form-control' #usernamePass [placeholder]='placeholders.userpass'>
  </label>
  <br>
  <input type='button' (click)='recibirDatos(usernameField, usernamePass);' class='btn btn-primary'
    #sendButton value='Aceptar'>
</form>
```

Exercici 1


MyForms
Inicio
Formularios
Formulario 01
Formulario 02
Formulario 03
Formulario 04
Cabecera global del sitio
Primer formulario
Nombre de usuario:
Teclea tu nombre de usuario
Contraseña:
Teclea tu contraseña
Aceptar
Falta el nombre de usuario, la clave o ambos.

Formularis

- Ara farem servir classes per a desacoblar la informació que arriba del formulari:
  - ng g class *directori/nom-class*
- La classe es crea **buida** dins del directori indicat:
  - Declarem les propietats de la clase amb l'indicador **private**
  - Afegim els mètodes GET i SET per acceder a les propietats privades.
- Afegim el constructor per a inicialitzar les variables
- Afegim un mètode per a generar l'ID únic: ex.:

```
private uniqueId() {
 const thisMS: number = Date.now();
 const shake = Math.random();
 let unique: string = Math.pow(thisMS, shake).toString();
 unique = unique.toString().replace('.', thisMS.toString());
 return unique;
}
```

- Ara, al binding de events li passarem l'ID del formulari (#newUserForm)

.

Exercici 2

- Utilitzar una clase per a recollir les dades del formulari
- STEPS:
  1. Crear la clase a app/src/shared/**classes**
  2. Afegir contingut a la clase (propietats, constructor, mètodes d'accés i mètode per a generar l'ID)
  3. Crear un nou formulari (Form02) dins del mòdul MyForms enllaçat des de la barra de navegació
  4. Importar la clase creada dins del nou component
  5. Crear un nou objecte user del tipus de la clase (new *nom_clase()*)
  6. Crear el mètode createUser passant-li com a paràmetre el formulari
     1. Obtenir el valor del formulari i assignar-ho a la propietat de l'objecte.
  7. Modificar el formulari afegint nom al formulari i passant-ho al mètode del evento.
  8. Obtenir la mateixa sortida per pantalla que abans (ara els valors s'obtendrien dels camps de l'objecte user).

Validació de formularis

- Per a validar un formulari farem servir el doble-binding. S'utilitza la notació *banana in a box* ("[(ngModel)]") per a passar en temps real el valor d'un objecte del formulari a un objecte de la classe.

[(ngModel)]='userData.realname'

- Substitueix al binding de la propietat *value* dins d'element HTML.
- S'utilitza la directiva ngModel = valor de la variable. Requereix:
  - que el camp tingui establerta la propietat *name* i
  - que s'hagi importat el FormsModule d'Angular dins del mòdul.

Exercici 3

- Farem la validació del formulari mitjançant un objecte temporal que modifica una variable quan es passan les validacions.
- STEPS:
  - Creem una nova classe FullUser:
    - amb més propietats: id, realname, username, password, singingUpDate, email, continente, gender, upTo18
    - el constructor inicialitza totes les propietats, en particular a singingUpDate se li assigna la data actual:

```
private getActualDate(): string {
  const actualDate = new Date();
  let actualDay = actualDate.getDate().toString();
  if (actualDay.length < 2) { actualDay = '0' + actualDay; }
  let actualMonth = actualDate.getMonth().toString();
  if (actualMonth.length < 2) { actualMonth = '0' + actualMonth; }
  const actualYear = actualDate.getFullYear().toString();
  const finalDate = actualDay + '-' + actualMonth + '-' + actualYear;
  return finalDate;
}
```

Exercici 3 (cont)

- STEPS:
  - Importem la classe al nou formulari: Form03.
  - Creem els placeholders al component (realname, username, userpass, userconfirmpass, email)
  - Creem un hash userData amb variables per a tots els camps del formulari. S'actualitzaran a temps real quan s'ompli el formulari.
  - Creem un hash per als valors dels radiobuttons:

```
public genders = {
  H: {
    label: 'Hombre',
    id: 'gender_H',
    value: 'H',
    name: 'genderOption',
    checked: false
  },
  M: {
    label: 'Mujer',
    id: 'gender_M',
    value: 'M',
    name: 'genderOption',
    checked: true
  }
};
```

Exercici 3 (cont)

- STEPS:
  - Creem el hash de continents:
  - Creem una variable per l'estat del botó.
  - Mètode checkButton que modifiqui l'estat de la variable quan tots els camps s'han omplert
  - Mètode saveUser() per a generar l'objecte user i enviar-ho a persistir

```
public continents = [
  {
    value: '0',
    text: 'Selecciona continente',
    selected: true,
    disabled: true
  },
  {
    value: 'EU',
    text: 'Europa',
    selected: false,
    disabled: false
  },
  {
    value: 'AM',
```

Exercici 3 (cont)

- STEPS:
  - Creem el formulari:

```
<div class="container">
    <h1>Tercer formulario</h1>
    <form (change)="checkButton()">
      <div class="row">
        <div class="col-sm-4">
          <label>Nombre real:
            <input type="text" class="form-control" name="realName" #realnamefield=""
                   [placeholder]="placeholders.realname"
                   [(ngModel)]="userData.realname">
          </label>
        </div>
```

  - El selector de continents:

```
    <div class="col-sm-4">
      <label>Continente:</label>
      <select class="form-control" name="userContinent" [(ngModel)]="userData.continent">
          <option *ngFor="let continent of continents"
            [value]="continent.value"
            [selected]="continent.selected"
            [disabled]="continent.disabled">{{continent.text}}
          </option>
      </select>
    </div>
```

Exercici 3 (cont)

- STEPS:
  - Els radiobuttons:

```
<div class="row">
  <div class="col-sm-2">
    <label>Género:</label>
  </div>
</div>
<div class="row">
  <div class="col-sm-2">
    <label>
      <input type="radio" #{{genders.h.id}}=""
             [value]="genders.H.value"
             [name]="genders.H.name"
             checked=""
             [(ngModel)]="userData.gender">
      {{genders.H.label}}
    </label>
  </div>
```

  - CheckBox per a la propietat upTo18
  - El botó acceptar amb el binding del mètode saveUser();

  - La resposta será una deserialització del JSON de l'usuari per la mateixa pantalla

Validació de formularis

- La validació requereix un disseny previ de les regles de validació:
  - Quins camps seran obligatoris, i quins seran opcionals.
  - Quin serà la longitud mínima o màxima d'una dada (en camps de text).
  - Quins seran els valors mínim i màxim en camps numèrics.
  - Quantes opcions es podran triar, per exemple, en un selector múltiple.
  - Si l'avís d'error es mostrarà en prémer un botó d'enviament (per exemple) o en abandonar un camp.
  - I, en general, qualsevol cosa que se'ns ocorri com a criteri.

Validació de formularis

- La assignació de la directiva ngModel com a valor al ID del camp (*local template variable*) del camp (#myVar = "ngModel") permet accedir les propietats associades al control del formulari:

Property	Type	Description
invalid	boolean | null	The control must have failed at least one of its validation checks.
errors	ValidationErrors| null	Returns any errors generated by failing validation. If there are no errors, it will return null.
dirty	boolean | null	A control is dirty if the user has changed the value in the UI.
touched	boolean | null	A control is marked touched once the user has triggered a blur event on it.

Exercici 4

- Formulari amb quatre camps, obligatoris, validats de diferents maneres, per veure les diferents possibilitats:
  - El camp del nom d'usuari es valida en temps real, segons es tecleja.
  - Els camps de contrasenya, repetició de la contrasenya i selecció d'idiomes parlats es validaran quan l'usuari premi el botó.
- Les regles de validació seran les següents:
  - Per al nom de l'usuari, és obligatori teclejar, almenys, quatre caràcters. Tindrà una limitació d'un màxim de deu caràcters
  - Per a la contrasenya es demana un mínim de sis caràcters, amb una limitació màxima de deu.
  - Per a la repetició de la contrasenya es requerirà que la dada teclejada coincideixi amb el del camp anterior.
  - Per a la llista d'idiomes parlats es requerirà que el candidat marqui, almenys, tres dels quals apareixen en la llista.

Exercici 4(cont)

- Fem una validació mixta del formulari. El camp userName es validarà a temps real mitjançant les propietats obtingudes de la directiva ngModel. La resta de camps es validen a la lògica del component.
- STEPS:
  - Construcció del formulari Form04 amb els elements enllaçat amb doble binding amb variables de la lògica.
  - Introduïm propietats de validació HTML (required, minlength, maxlength al cap username
  - Afegim un multiselector de països.
  - Construïm la part de validació

Exercici 4(cont)

- Part de validació errors:

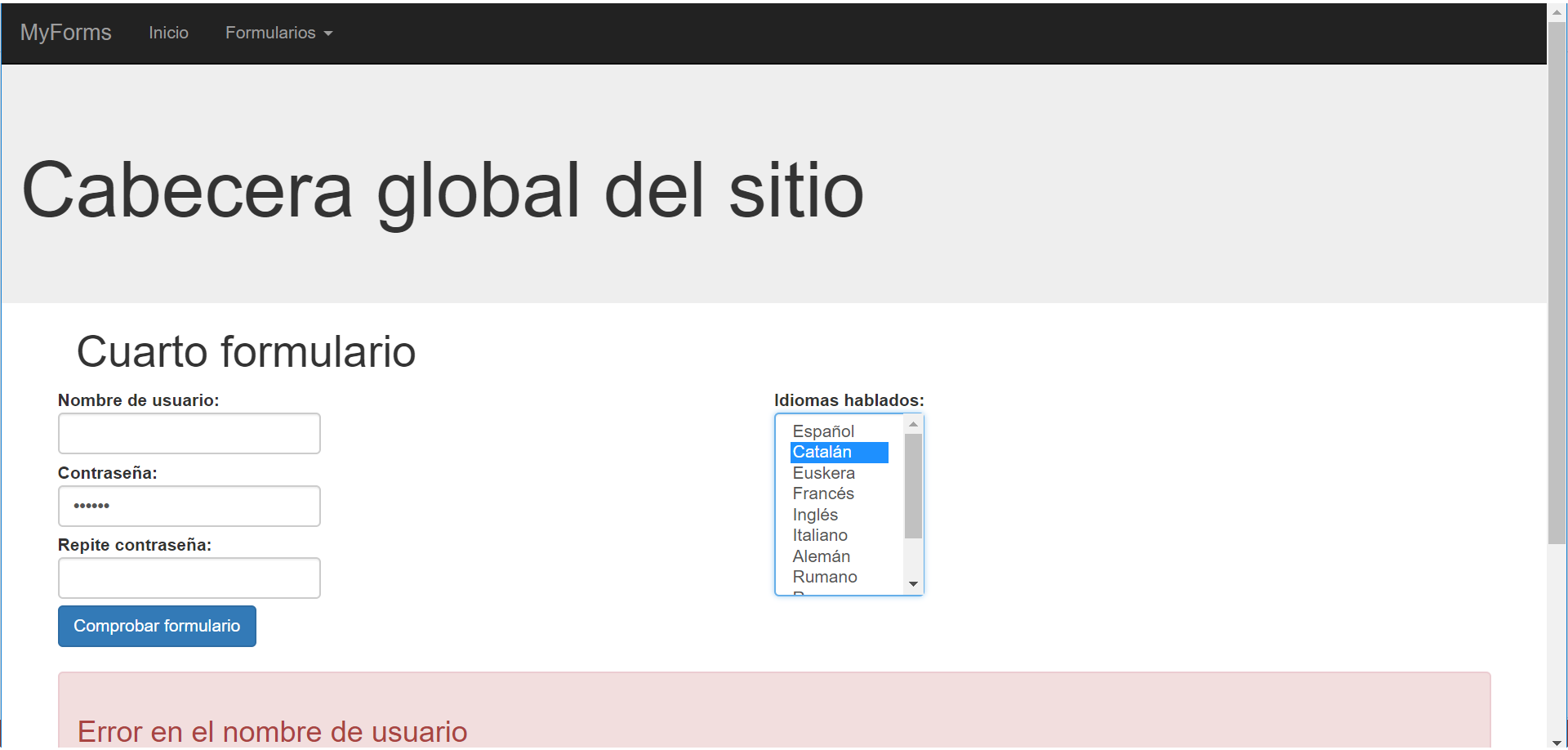
```
<div class="row alert alert-danger"
*ngIf="mensajeDeError || (userNameModel.invalid &amp;&amp; userNameModel.touched)">
  <div *ngIf="userNameModel.invalid">
    <h3>Error en el nombre de usuario</h3>
    <span *ngIf="!!userNameModel.errors.required">
      <p>Este campo es obligatorio</p>
    </span>
    <span *ngIf="!!userNameModel.errors.minlength">
      <p>Debes teclear 4 caracteres o más</p>
    </span>
  </div>
  <div *ngIf="errorPass">
    <h3>Error en contraseña</h3>
    <p>La contraseña es demasiado corta</p>
  </div>
  <div *ngIf="errorConfirmPass">
    <h3>Error en confirmación de contraseña</h3>
    <p>La confirmación no coincide con la contraseña</p>
  </div>
  <div *ngIf="errorLanguages">
    <h3>Idiomas insuficientes</h3>
    <p>Debes hablar, al menos, tres idiomas de la lista.</p>
  </div>
</div>
```

Exercici 4(cont)

- Part de validació ok:

```
<div class="row alert alert-success" *ngIf="mensajeDeOk">
  <h3>Correcto</h3>
  <p>Tus datos se han registrado correctamente.</p>
</div>
```

Exercici 4


MyForms
Inicio
Formularios
Cabecera global del sitio
Cuarto formulario
Nombre de usuario:
Contraseña:
••••••
Repite contraseña:
Comprobar formulario
Idiomas hablados:
Español
Catalán
Euskera
Francés
Inglés
Italiano
Alemán
Rumano
Error en el nombre de usuario

Jerarquia de components

- Requerim passar informació entre components per a fer un control més desacoblat.
- Definim un component *master* que controla la informació que es mostra als altres components *slaves.* La lògica es treballa al component màster centralitzant totes les accions.
- Requereix la definició de dos decoradors nous:
  - @input -- permet rebre dades d'altres components
  - @output + la classe EventEmitter -- permet enviar dades.

Exercici 5

- Fem una nova aplicació de control de socis. Tindrà un formulari per donar d'alta un soci i es mostrarà el registre nou i el llistat de socis.
  - El formulari serà la vista d'un component *slave*: SocioComponent, aquest enviarà la información al component *master* SocioManagerComponent, que mostrarà el registre i ho enviarà novament a una altra vista: SocioListComponent que mostrarà el llistat.
  - Tres mòduls:
    - GlobalModule: components comuns
    - MyFormModule: component formulari
    - FormManagerModule: components gestor i llistat
  - Una classe nova per a recollir les dades del nou soci.

Exercici 5(cont)

- La barra de navegació enllaçarà amb el formulari mitjançant el component master, el qual incorporarà al formulari.
- El formulari (SociosForm):
  - Porta dos camps DNI socio i Nombre.
  - Definim una validació de camps omplerts.
  - Afegim la declaración de un objecte amb el decorador @Output() d'una variable de la clase EventEmitter al que s'hi passa el tipus d'objecte que volem:

```
@Output() enviarUsuario = new EventEmitter<AcdUser>();
```

  - Ara, s'ha d'enviar l'objecte amb el mètode emit, associat a l'event click del botó:

```
saveRecord() {
  // Creamos un objeto de la clase AcdUser
  this.User = new AcdUser(this.dni, this.nombre);
  // Enviamos el objeto al componente maestro.
  this.enviarUsuario.emit(this.User);
```

Exercici 5(cont)

- El component SociosManager vs el formulari:
  - Implementació del formulari dins del Master. Dins de les etiquetes incorporen un binding de eventos amb la variable que hem exportat des del formulari:

```
<acd-socios-form
    (enviarUsuario)='recibir($event)'>
</acd-socios-form>
```

  - El mètode recibir() emmagatzema les dades d'usuari a l'array d'objectes usuari:

```
recibir(usuario) {
    this.UsersArray.push(usuario);
    this.registeredUsers = this.UsersArray.length;
  }
}
```

Exercici 5(cont)

- El component SociosManager vs el llistat (SociosList):
  - Implementació del llistat. Dins de les etiquetes incorporen un binding de eventos amb el mètode borrar usuari i les variables que volem traspassar al llistat:

```
<acd-socios-list
  [UsersArray]="UsersArray"
  [registeredUsers]="registeredUsers"
  (eraseUser)="erase($event)">
</acd-socios-list>
```

  - S'implementa el mètode erase() per a esborrar un usuari de l'array d'objectes. Utilitza el mètode splice per a esborrar el registre i actualitza la propietat registeredUsers

Exercici 5(cont)

- El component SociosList:
  - Construcció de la vista a partir de les dades de l'array d'usuaris

```
<div class="row">
  <strong>Número de usuarios registrados: {{registeredUsers}}</strong>
</div>
<div *ngIf="registeredUsers > 0">
  <div class="row">
    <h4>Lista de Usuarios</h4>
  </div>
  <div class="row">
    <div class="col-sm-4">ID</div>
    <div class="col-sm-2">DNI</div>
    <div class="col-sm-5">NOMBRE</div>
    <div class="col-sm-1"> </div>
  </div>
  <div *ngFor="let UserItem of UsersArray">
    <div class="row">
      <div class="col-sm-4">{{ UserItem.id }}</div>
      <div class="col-sm-2">{{ UserItem.dni }}</div>
      <div class="col-sm-5">{{ UserItem.nombre }}</div>
      <div class="col-sm-1 text-right">
        <input type="button" (click)="deleteUser(UserItem)" class="btn btn-danger btn-xs" value="X">
      </div>
    </div>
  </div>
</div>
```

Exercici 5(cont)

- El component SociosList: la lògica
  - Declaració amb @Input de les variables del binding de propietats (registeredUsers i ArrayUsers)
  - Declaració amb @Output del mètode eraseUser
  - Implementació del mètode deleteUser associat al evento click del botó esborrar.

Exercici 5(bonus)

- Afegeix la validació del formulari al component màster:
  - Camps omplerts i dni no repetit.