esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2025-2026 ANGULAR

Workshop n°7 :

Reactive Forms

Les objectifs :

- Créer un formulaire avec l'approche Reactive Form.
- Ajouter des validateurs
- Gérer les messages d'erreurs.
- Créer un validateur personnalisé

Le rendu final :

Créer le formulaire suivant permettant l'ajout d'un événement :

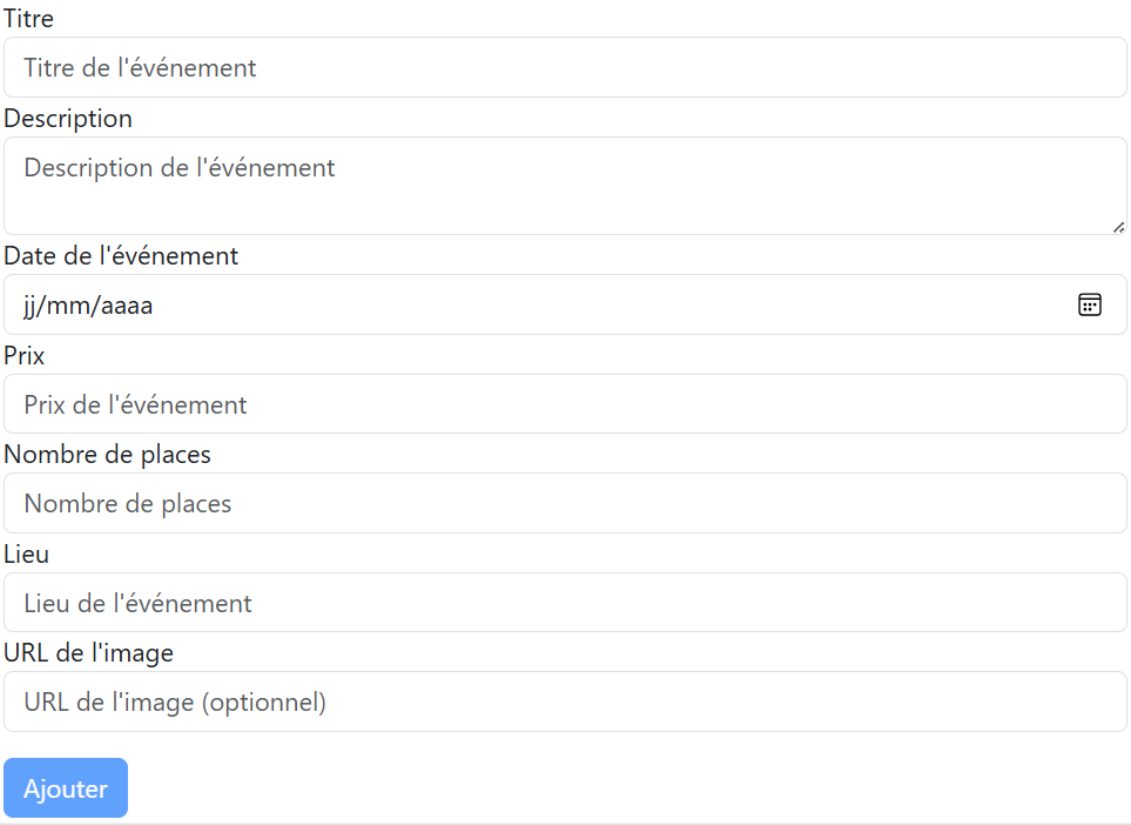

Figure 1 : addEventComponent : Formulaire d'ajout d'un évènement

Les instructions à suivre :

A- Création de formulaire

1- Dans le composant ListEventCompenent, ajouter un bouton « Proposer un événement » comme indiqué dans la figure 2.

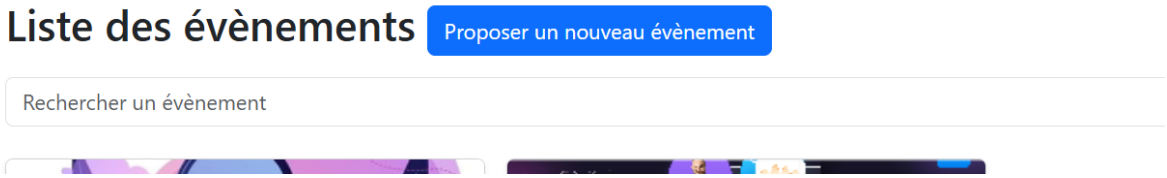

Figure 2 : ListEventComponent : bouton « Proposer un évènement »

2- Générer le composant addEventComponent sous features/events/components :
Taper la commande **: ng g c features/events/components /addEvent – -skip-tests**

3- Ajouter une route menant vers ce composant dans EventsRoutingModule et lier cette route avec le bouton « Proposer un événement »

4- Créer le formulaire permettant l'ajout d'un évènement selon l'approche Reactive Forms.
Un petit rappel sur le modèle de donnée « Event »

```
export interface Event {
  id : number;
  title : string;
  description: string;
  date : Date;
  place:string;
  price:number;
  organizerId:number;
  imageUrl:string;
  nbPlaces:number;
  nbLikes:number;
}
```

Figure 3 : l'interface Event

NB :

- Le id est auto-incrément
- Pour l'image, on se content de saisir une chaine de caractère.
- Les champs relatifs aux attributs « nbLikes » et « OrganizerId », ne figurent pas dans le formulaire. Lors de l'ajout d'un évènement, par défaut nbLikes est égal à 0 et OrganizerId est égal à 1.

B- Validation de de formulaire – Validateurs prédéfinies

5- En tenant compte des remarques cité plus haut, ajouter les validateurs suivants au niveau du formulaire :

- Tous les champs sauf le champ de l'image sont obligatoires.
- Le champ « Titre » contient au minimum 5 caractères, et ne contient que des lettres.

expression régulière : "[a-zA-Z]*"

- Le champ « Description » contient au minimum 30 caractères.
- Le champ « Prix » n'accepte que des chiffres et des points.
  expression régulière : "^\d+(\.\d+)?$"
- Le champ « Nombre de places » accepte un nombre inférieur ou égal à 100.
  expression régulière : "^[1-9][0-9]?$|^100$"
- Le bouton « Ajouter » est désactivé tant que le formulaire est INVALID .

6- Ajoutez un message d'erreur bien précis associé à chaque validateur.

7- Une fois le bouton « Ajouter » est actif, en cliquant dessus, les informations sont saisies au niveau du formulaire sont ajoutées dans la liste des évènements.

8- Ajouter les classes CSS suivants au niveau du fichier styles.css et détecter les changements sur le formulaire.

```
input.ng-touched, textarea.ng-touched{
    border: 2px solid blue;
    background-color: #e3f2fd;
}
input.ng-untouched, textarea.ng-untouched {
    border: 2px solid lightgray;
    background-color: #f5f5f5;
}
input.ng-valid.ng-touched, textarea.ng-valid.ng-touched {
    border: 2px solid green;
    background-color: #e8f5e9 ;
}
input.ng-invalid.ng-touched, textarea.ng-invalid.ng-touched  {
    border: 2px solid red;
    background-color: #f8d7da;
}
```

C- Validation de formulaire – Validateurs personnalisés

9- Créer un validateur personnalisé qui permet de vérifier que la date de l'évènement aura lieu dans au moins de 7 jours. Pour cela,

9.1- Créer sous le dossier Shared/Validators le fichier futur-date.validator.ts

9.2- Ecrire dedans la fonction suivante :

```
import { AbstractControl,ValidationErrors, ValidatorFn } from "@angular/forms";

export function futurDateValidator(days : number): ValidatorFn {

return (control: AbstractControl): ValidationErrors | null => {

const selectedDate = new Date(control.value);

const today = new Date();
```

```
const minDate = new Date(today.setDate(today.getDate() + days));

if (isNaN(selectedDate.getTime()) || selectedDate < minDate) {

return { futureDateInvalid: 'La date doit être dans le futur, minimum ' + days +' jours à partir d\'aujourd\'hui.' }; }

 return null;

};}
```

9.3- Appliquer ce validateur synchrone sur le champ date et afficher le message correspondant en cas d'erreur.

D- FormArray – Manipulation

10- Ajouter un nouveau champ « Domaine » permettant l'ajout du domaine de l'évènement. Si un évènement inclut plusieurs domaines, il faut donner la main à l'utilisateur de saisir les différents domaines.

⇨ Le nombre des domaines varie d'un évènement à un autre.

⇨ Pour cela, il faut choisir d'ajouter un élément de type **FormArray** à votre formulaire.

```
   domaines : new FormArray([new FormControl('')])
```

11- Il faut ajouter deux méthodes : une pour la récupérration du domaine et une autre pour l'ajout d'un domaine

```
get domaines() {return this.eventForm.get('domaines') as FormArray; }

addDomain() { this.domaines.push(new FormControl(''));}
```

12- Chaque domaine ajouté est obligatoire, contient au minimum 3 caractères et au maximum 20 caractères. Afficher le message correspondant en cas d'erreur.

13- Mettre à jour votre interface du modèle de données « Event » en lui ajoutant un nouveau attribut optionnel : **domaines ?:string[]**

E- FormGroup imbriqué : Travail à faire (optionnel)

14- Penser à ajouter un nouvel attribut à votre modèle « Event » appelé « **detailedAddress**». L'adresse doit contenir l'avenue, la ville, le gouvernorat ainsi que le code postal sous forme de champs séparés comme indiqué ci-dessous.

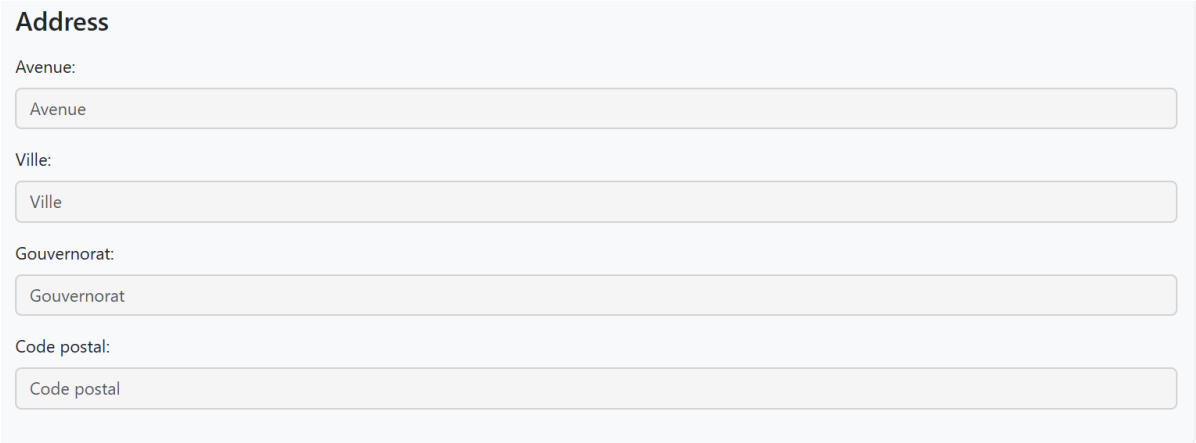

Figure 4 : Adresse détaillée

L'adresse récupérée sera sous forme d'un objet de type « Address » à créer.

Exemple **:**

detailedAddress: {street: 'Liberation', city: 'Ariana', governorate: '', zipcode: '8052'}