TP "Book Catalog"
Angular 20 (Standalone & Template-Driven)

I. Objectifs pédagogiques

À la fin de ce TP, l'étudiant sera capable de :

- Créer un projet Angular 20 en Standalone Components
- Développer un CRUD complet (Create / Read / Update / Delete)
- Construire un formulaire template-driven avec validations
- Utiliser [(ngModel)], @for, @if, @Input, @Output
- Gérer l'état d'un formulaire et la réinitialisation
- Communiquer entre composants parent/enfant
- Appliquer validation + styles dynamiques

II. Partie0 — Initialisation

1. Créer un projet Angular :

ng new tp-books --standalone

2. Effacer le contenu de app.html.

III. Partie1 — Structure générale

3. Créer une classe Book avec les champs :
   - id : number
   - title : string
   - author : string
   - publisherEmail : string
   - publisherPhone : string
   - releaseDate : string
   - category : string
   - isAvailable : boolean
   - stock : number (facultatif)
4. Générer les composants Standalone :
   - book-container (parent)
   - book-form (ajout / modification)
   - book-list (affichage & actions)
5. Utiliser book-container comme composant principal.

IV. Partie2 — Données

6. Dans book-container, créer une liste initiale de livres avec uniquement les champs retenus.
7. Créer un tableau de catégories : Roman, Science, Histoire, Informatique, Art, Autres.
8. Transmettre cette liste au composant book-list via @Input().

V. Partie3 — Formulaire (CREATE & UPDATE)

9. Dans book-form, créer un formulaire template-driven :
<form #bookForm="ngForm" (ngSubmit)="onSubmit(bookForm)">.
10. Ajouter les champs suivants avec [(ngModel)] + name + id :
- title (text)
- author (text)
- publisherEmail (email)
- publisherPhone (Number, tunisien)
- releaseDate (date)
- category (select)
- isAvailable (checkbox)
- stock (number facultatif)

VI. Partie4 — Validations obligatoires

11. Validations :
- title : required, minlength=3
- author : required, minlength=3
- publisherEmail : required, email
- publisherPhone : facultatif, mais pattern="^[0-9]{8}$"
- releaseDate : required, > 1900
- category : required
- stock : min=0 (si rempli)

VII. Partie5 — Messages d'erreur

12. Sous chaque champ, afficher un message si invalid & dirty.
13. Ajouter un style rouge sur les champs invalides avec :
.ng-invalid.ng-dirty { border-left: 5px solid red; }

VIII. Partie6 — CREATE

14. Lors du ngSubmit :
- envoyer le book au parent via @Output()
- générer automatiquement l'id
- ajouter dans la liste du container
- réinitialiser le formulaire

IX. Partie7 — READ

15. Dans book-list :
- recevoir la liste via @Input()
- afficher un tableau avec : title, author, category, disponible, actions.
16. Utiliser @for (book of books; track book.id).

X. Partie8 — DELETE

17. Dans book-container, implémenter deleteBook(id) pour retirer l'élément.

XI. Partie9 — UPDATE

18. Lorsqu'un livre doit être modifié :
- envoyer le livre depuis book-list vers book-container
- précharger le formulaire dans book-form
- changer bouton "Ajouter" → "Mettre à jour"
19. Lors du submit :
- mettre à jour le livre dans la liste
- revenir en mode ajout
- reset du formulaire

XII. Partie10

20. Interdire un titre composé uniquement de chiffres.
21. Ajouter un champ de recherche filtrant la liste.
22. Ajouter un tri par catégorie ou disponibilité.
23. Ajouter un compteur dynamique du nombre total de livres.