Étape 4 : Système d'annonces

Objectif

Créer un système d'annonces avec chargement de données et gestion des états loading/error.

Concepts pratiqués

- useEffect pour les effets de bord
- Simulation d'appel API (fetch)
- Gestion des états : loading, error, data
- Formulaire d'ajout

À créer

1. Composant `AnnouncementCard.jsx`

Une carte d'annonce affichant :

- Titre de l'annonce
- Contenu (avec mise en forme)
- Auteur et son rôle
- Date de publication (formatée)
- Badge si épinglée
- Catégorie

2. Composant `AnnouncementList.jsx`

La liste qui :

- Simule un chargement de données (setTimeout)
- Affiche un loader pendant le chargement
- Gère les erreurs potentielles
- Affiche les annonces épinglées en premier
- Permet de filtrer par catégorie

3. Composant `AnnouncementForm.jsx`

Un formulaire pour ajouter une annonce :

- Champs : titre, contenu, catégorie
- Validation basique (titre requis, contenu min 10 caractères)
- Ajout à la liste après soumission

> **Note** : Pour l'instant, tout le monde peut poster une annonce. À l'étape 6, on pourra conditionner l'affichage du formulaire selon le rôle de l'utilisateur sélectionné.

Pattern de fetch avec useEffect

```
// ⚠ ATTENTION : useEffect ne peut PAS être async directement !

// ❌ ERREUR COURANTE
useEffect(async () => {  // NON !
  const data = await fetch(...);
}, []);

// ☑ CORRECT : Créer une fonction async à l'intérieur
useEffect(() => {
  async function loadData() {
    const response = await fetch(...);
    const data = await response.json();
    setData(data);
  }

  loadData();
}, []);
```

Indices

▶ 💡 Structure du state pour le loading

```
const [announcements, setAnnouncements] = useState([]);
const [isLoading, setIsLoading] = useState(true);
const [error, setError] = useState(null);

// Dans le useEffect
try {
  setIsLoading(true);
  // ... charger les données
  setAnnouncements(data);
} catch (err) {
  setError(err.message);
} finally {
  setIsLoading(false);
}
```

▶ 💡 Simuler un délai de chargement

```
// Simuler un appel API avec délai
function simulateFetch(data, delay = 1000) {
  return new Promise((resolve) => {
    setTimeout(() => resolve(data), delay);
  });
}
```

```
// Utilisation
const data = await simulateFetch(announcements, 1500);
```

▶ 💡 Trier les annonces (épinglées en premier)

```
// sort() modifie le tableau original, donc on fait une copie
const sortedAnnouncements = [...announcements].sort((a, b) => {
  // Les épinglées en premier (true > false)
  if (a.isPinned && !b.isPinned) return -1;
  if (!a.isPinned && b.isPinned) return 1;
  // Puis par date (plus récent en premier)
  return new Date(b.createdAt) - new Date(a.createdAt);
});
```

▶ 💡 Ajouter une annonce à la liste

```
function handleAddAnnouncement(newAnnouncement) {
  // Créer l'annonce avec les données manquantes
  const announcement = {
    id: Date.now(), // ID temporaire unique
    ...newAnnouncement,
    author: "Utilisateur actuel", // À l'étape 6, viendra du contexte
    authorRole: "Employé",
    createdAt: new Date().toISOString(),
    isPinned: false,
  };

  // Ajouter en début de liste
  setAnnouncements(prev => [announcement, ...prev]);
}
```

Points d'attention

⚠ Le tableau de dépendances de useEffect

```
// S'exécute à CHAQUE rendu (rarement voulu)
useEffect(() => { ... });

// S'exécute UNE SEULE FOIS au montage
useEffect(() => { ... }, []);

// S'exécute quand `id` change
useEffect(() => { ... }, [id]);
```

⚠ Cleanup function

Si le composant peut être démonté pendant le chargement :

```
useEffect(() => {
  let isMounted = true;

  async function loadData() {
    const data = await fetch(...);
    // Vérifier que le composant est encore monté
    if (isMounted) {
      setData(data);
    }
  }

  loadData();

  // Cleanup : appelé quand le composant est démonté
  return () => {
    isMounted = false;
  };
}, []);
```

⚠ Formater les dates

```
// Utiliser toLocaleDateString pour un affichage lisible
const formattedDate = new Date(announcement.createdAt).toLocaleDateString('fr-FR',
{
  day: 'numeric',
  month: 'long',
  year: 'numeric',
  hour: '2-digit',
  minute: '2-digit'
});
// Résultat : "15 janvier 2025 à 10:30"
```

Critères de validation

- ☐ Un loader s'affiche pendant le chargement
- ☐ Les erreurs sont gérées et affichées
- ☐ Les annonces épinglées apparaissent en premier
- ☐ Le filtre par catégorie fonctionne
- ☐ Les dates sont formatées correctement
- ☐ Le formulaire permet d'ajouter une annonce
- ☐ La validation empêche les soumissions invalides