1. Mini-projet Spring Boot

API de gestion de conférences (Conference Manager API)

Objectif

Construire une **API REST propre, sécurisée et testée**, en appliquant les bonnes pratiques Spring modernes.

Fonctionnalités

- CRUD **Conférence**
- CRUD **Intervenant**
- Association Conférence ↔ Intervenants (Many-to-Many)
- Inscription d'un participant à une conférence
- Pagination et filtrage
- Gestion des erreurs (exceptions globales)
- Sécurité JWT (login / register)

Stack technique

- Spring Boot 3.x
- Spring Data JPA (Hibernate)
- PostgreSQL ou H2
- Spring Security + JWT
- Validation (`@Valid`)
- OpenAPI / Swagger
- Tests unitaires (JUnit + Mockito)

Modèle de données (simplifié)

```
Conference(id, title, date, location)
Speaker(id, name, expertise)
Participant(id, email)
```

Relations :

- Conference — Speaker
- Conference — Participant

Endpoints clés

```
POST   /auth/login
POST   /auth/register

GET    /conferences
POST   /conferences
PUT    /conferences/{id}
DELETE /conferences/{id}

POST   /conferences/{id}/register
```

Ce que tu maîtriseras

- Architecture Spring propre (Controller / Service / Repository)
- JPA et relations
- Sécurité réelle
- Documentation d'API
- Tests backend

---

2. Mini-projet Angular

Front-end de gestion de conférences

Objectif

Créer une **application Angular moderne**, structurée et maintenable, connectée à l'API Spring.

Fonctionnalités

- Page login / register (JWT)
- Liste des conférences (pagination)
- Détails d'une conférence
- Inscription à une conférence
- Création / édition (admin)

- Guards de routes
- Gestion d’erreurs HTTP

Stack technique

- Angular 17+
- RxJS
- Angular Router
- Reactive Forms
- Angular Material ou PrimeNG
- Interceptor HTTP (JWT)

Architecture recommandée

```
/core        (auth, guards, interceptors)
/features
  /conference
  /auth
/shared
```

Composants clés

- `ConferenceListComponent`
- `ConferenceDetailComponent`
- `ConferenceFormComponent`
- `LoginComponent`
- `RegisterComponent`

Ce que tu maîtriseras

- Architecture Angular scalable
- State management léger (services + RxJS)
- Sécurité côté client
- UX propre
- Consommation d’API REST

Bonus (fortement recommandé)

Si tu veux te démarquer :

- Dockeriser Spring + Angular
- Déployer sur AWS / Render / Railway
- Ajouter des tests Angular (Jasmine/Karma)
- Ajouter un README clair + diagramme d’architecture
- Diagramme UML simple (classes + séquence)

Variante orientée Design Patterns (bonus avancé)

Vu ton profil :

- Implémenter **Observer** pour notifier les participants
- **Factory** pour créer différents types de conférences
- **Strategy** pour les règles d’inscription
