Architecture Decision Record (ADR)
ADR-001 — Choix du framework frontend

Statut : Accepté
Date : 07/11/2025
Portée : Micro-service Frontend de la solution *MediLabo* Solutions
Décideur : Équipe technique (référent : Michael ARTRU)
Contexte projet : Application web médicale de dépistage du risque de diabète, architecture micro-services exposés via Spring Cloud Gateway

I. Contexte

Le projet MediLabo est bâti sur une architecture de micro-services Java/Spring Boot (patients, notes, calcul de risques) exposés derrière un *Gateway*, avec un front séparé à faire évoluer au fil des 3 sprints (affichage patients, notes, puis rapport de risque). Le *frontend* doit donc :

1. Consommer des APIs REST sécurisées exposées par le *Gateway Spring Cloud*.
2. Évoluer incrémentalement (sprints 1 à 3) sans remettre en cause le socle à chaque ajout fonctionnel.
3. Rester sobre et maintenable.
4. S'intégrer facilement à la stack technique Java/Spring déjà posée (culture d'équipe backend, CI/CD *GitHub*, dockerisation).

Dans ce contexte, il faut arrêter un **framework frontend unique**.

II. Problème

Quel framework frontend choisir pour :

1. Développer rapidement les écrans patients / notes / rapport
2. Garantir la cohérence du code sur plusieurs sprints
3. Être facilement pris en main par une équipe plutôt orientée backend Java
4. Rester compatible avec un déploiement *dockerisé*

III. Options considérées

Option 1 : Angular

Avantages :

- Framework complet et opinionné, fournissant une architecture claire.
- CLI puissante facilitant la génération et la maintenance du code.
- Support intégré pour les formulaires, la validation et les requêtes HTTP.
- Idéal pour les applications d'entreprise avec API REST.
- Structure proche des pratiques d'équipes backend Java.

Inconvénients :

- Courbe d'apprentissage potentiellement raide.
- Poids initial du bundle supérieur aux alternatives plus légères.
- Framework très structuré, laissant moins de liberté d'implémentation.

Option 2 : React

Avantages :

- Très populaire, communauté active et riche écosystème.
- Grande flexibilité dans le choix des outils (router, gestion d'état, tests).
- Bonnes performances globales.

Inconvénients :

- Nécessite de choisir et configurer plusieurs bibliothèques externes.
- Manque d'architecture imposée : risque d'incohérence sur plusieurs sprints.
- Plus difficile à industrialiser rapidement dans un contexte *Java/Spring*.

Option 3 : Vue

Avantages :

- Syntaxe simple et approche intuitive.
- Léger et rapide à mettre en place.
- Documentation claire, bon compromis entre *Angular* et *React*.

Inconvénients :

- Moins présent dans les environnements Java/Spring.
- Moins de standardisation dans les grands projets d'entreprise.
- Courbe de maturité moindre pour certaines bibliothèques avancées.

Option 4 : Thymeleaf

Avantages :

- Intégration native avec Spring Boot, idéale pour les développeurs Java.
- Pas besoin d'un front séparé, simplifiant le déploiement et la maintenance.
- Génération côté serveur, donc rendu immédiat sans dépendre du JavaScript.
- Très bon choix pour des interfaces simples ou administratives.

Inconvénients :

- Faible interactivité côté client (rechargement complet des pages).
- Moins adapté aux applications dynamiques à base d'API REST.
- Évolutivité limitée si l'interface doit devenir une SPA complète.

IV. Décision

Nous choisissons d'utiliser **Angular** comme framework frontend officiel du projet.

V. Justification

1. Angular offre un cadre fort, facilitant les développements incrémentaux.
2. Son HttpClient simplifie l'intégration avec les micro-services REST.
3. Les formulaires réactifs conviennent aux données "patient".
4. TypeScript et l'outillage CLI s'alignent bien avec la culture Java/Spring.
5. Build et déploiements faciles dans Docker.

VI. Conséquences

Positives :

- Structure claire et évolutive.
- Intégration REST native.
- Uniformisation du code.
- Sécurisation aisée via interceptor.

Négatives :

- Courbe d'apprentissage.
- Poids du bundle initial.
- Moins de flexibilité pour les développeurs préférant des approches libres.

VII. Références

- Document ***Exigences du produit***
- Document ***Résumé des besoins fonctionnels***