ANNEXES • MAI 2026

Cercle Vulnérabilités et Société

FEUILLES DE ROUTE OPÉRATIONNELLES

Small Companions IA

Lucie • Étienne • Hector

Développement, expérimentation et déploiement des assistants d'intelligence artificielle pour l'accompagnement social et médico-social des personnes vulnérables

EN PARTENARIAT AVEC :

malakoff humanis

FEHAP Santé Social - Privé Solidaire

udsm

ET LE CONCOURS DE :

Alia Santé

TABLE DES MATIÈRES

INTRODUCTION

TROIS ASSISTANTS IA POUR TROIS IRRITANTS MAJEURS

Les feuilles de route présentées dans ce document détaillent le développement opérationnel de trois assistants d'intelligence artificielle, conçus pour répondre à des problématiques concrètes identifiées lors de la démarche d'évaluation et d'expérimentation des potentiels de l'IA dans l'accompagnement des vulnérabilités sociales et médico-sociales menée entre mai et décembre 2025.

Ces trois solutions — baptisées « Small Companions » — incarnent une approche délibérément ciblée, restreinte et éthique de l'IA dans le secteur social et médico-social. Contrairement à une logique de plateforme généraliste, elles visent chacune un irritant majeur, avec un périmètre fonctionnel limité, une prise en main rapide et une validation humaine systématique.

Une logique commune : sobriété, éthique, progressivité

Ces trois assistants partagent sept principes non négociables :

- Simplicité : répondre à un problème clairement identifié
- Restriction fonctionnelle : 2 à 3 fonctions maximum
- Validation humaine obligatoire : aucune action automatique
- Pas de scoring ni profilage : aucune note sur les personnes
- Minimalisme des données : uniquement le strict nécessaire
- Opérationnalité immédiate : sorties directement actionnables
- Sobriété de design : prise en main en moins de 2 minutes

Ce document détaille :

- Le cadre méthodologique et architectural commun aux trois solutions
- Les feuilles de route opérationnelles de Lucie, Étienne et Hector (problème, valeur, périmètre, architecture, pipeline IA, données, risques, KPI, planning, budget, facteurs de succès)
- Une synthèse comparative des trois solutions
- Les hypothèses budgétaires globales
- Les modalités de mise en œuvre et la trajectoire d'évolution cible

LES SMALL COMPANIONS DEVELOPPES

Bien que conçus pour fonctionner de manière autonome, ces trois Small Companions peuvent évoluer à terme vers une articulation plus intégrée, voire une fusion en un point d'accès unique. Cette double perspective — autonomie initiale et convergence potentielle — structure l'ensemble des feuilles de route présentées dans ce document.

	Lucie	Étienne	Hector
Cas d'usage - Assistant	**Le booster de la qualité relationnelle**	**L'auxiliaire situationnel**	**L'agent inclusif territorial**
Problème traité	La perte de continuité lors des changements d'intervenant génère du stress pour les professionnels et impose aux personnes accompagnées de répéter des éléments essentiels de leur situation.	La difficulté à formuler ses besoins avant un rendez-vous, lorsque la charge cognitive est élevée ou que la peur du jugement freine l'expression, dégrade la qualité de l'échange.	Le maquis administratif — multiplicité des guichets, procédures opaques, pièces manquantes, éloignement géographique — génère découragement, erreurs et abandon des démarches.
Solution proposée	Transformation de notes éparses en **une carte de continuité synthétique** d'une page, mobilisable en moins de deux minutes par un remplaçant. L'outil préserve la mémoire relationnelle : ce qui compte pour la personne, comment entrer en relation, quoi éviter.	Transformation d'une parole libre en **carte d'attentes structurée**, en langage accessible de type FALC. L'outil permet de préparer un rendez-vous en amont et de transmettre une demande plus claire au professionnel.	Guidage pas à pas dans les démarches administratives et orientation vers les bonnes ressources locales. L'outil convertit un besoin exprimé en **parcours actionnable** : checklists, messages prêts à envoyer, scripts d'appel, avec redirection vers un humain en cas de blocage.
Logique d'intervention	**Métier augmenté** : l'IA restructure l'information existante, sans prédire ni décider. Le professionnel valide, corrige et reste maître du contenu.	**Expression augmentée** : l'IA clarifie et structure, sans interpréter d'intentions cachées. La personne valide toujours sa carte avant partage.	**Inclusion augmentée** : l'IA guide procéduralement à partir d'une base de connaissances territoriale, mais bascule vers un humain dès que la situation devient complexe.
Public prioritaire pour l'expérimentation	Professionnels médico-sociaux	Personnes en situation de précarité et travailleurs sociaux	Aidants familiaux et personnes âgées vivant à domicile

PARTIE 1 : CADRE COMMUN

1. Finalité du document

Cette annexe a pour objectif de fournir un cadre opérationnel détaillé pour le développement, l'expérimentation et le déploiement des trois Small Companions IA présentés dans la note principale. Elle constitue un référentiel de mise en œuvre, destiné à :

- orienter les équipes de développement ;
- sécuriser les décisions d'investissement ;
- faciliter la coordination entre partenaires (techniques, institutionnels, territoriaux) ;
- garantir l'alignement avec les exigences éthiques et opérationnelles du secteur.

2. Cadre méthodologique commun

2.1 Principes structurants

Les trois Small Companions ne doivent pas être conçus comme trois démonstrateurs indépendants, ni comme trois produits entièrement distincts. Ils répondent à des irritants différents, mais reposent sur une même philosophie d'intervention : traiter un besoin simple, fréquent, critique pour la qualité de l'accompagnement, au moyen d'une IA restreinte, contrôlable, compréhensible et immédiatement utile. Cette logique est explicitement posée dans la note à travers le refus d'une « méga-plateforme » généraliste, la préférence pour des assistants restreints fonctionnellement, la validation humaine obligatoire, l'absence de scoring et le minimalisme des données.

Le présent cadre méthodologique a donc quatre fonctions :

- mutualiser ce qui doit l'être : les composants techniques transverses, les exigences de sécurité, la gouvernance des données, les mécanismes de journalisation, les règles d'orchestration des modèles et les principes d'interface.
- séparer clairement ce qui doit rester spécifique : les logiques métiers propres à Lucie, Étienne et Hector, leurs schémas de données utiles, leurs workflows opérationnels et leurs métriques d'évaluation.
- réduire le risque projet. Dans ce type de contexte, le risque principal n'est pas seulement technologique. Il est aussi organisationnel, éthique, réglementaire, budgétaire et d'appropriation. Un cadre commun permet d'éviter la multiplication de solutions hétérogènes, coûteuses à maintenir, difficiles à auditer et encore plus difficiles à déployer.
- préparer le passage de l'expérimentation à l'industrialisation. La note insiste sur le fait que la réussite dépendra de l'interopérabilité, de la formation, de l'accompagnement au changement et de la capacité à éviter la création « d'outils de plus » déconnectés des pratiques réelles. Le cadre commun répond précisément à cette exigence.

2.2 Principes directeurs de conception

2.2.1 Principe de proportionnalité fonctionnelle

Chaque compagnon doit être conçu pour résoudre un problème principal clairement identifié, avec un périmètre initial volontairement limité.
Ce choix n'est pas seulement une préférence produit ; c'est une contrainte méthodologique issue du diagnostic. Les recherches montrent que les usages les plus acceptables et les plus prometteurs ne sont pas les plus spectaculaires, mais ceux qui traitent des irritants du quotidien : continuité relationnelle, clarification d'un besoin, orientation dans des démarches complexes.
En conséquence, le cadrage initial de chaque compagnon doit respecter les règles suivantes :

- un irritant central ;
- deux à trois fonctions cœur au maximum ;
- un temps de prise en main court ;
- une sortie utile immédiatement actionnable ;
- un coût d'intégration compatible avec un pilote réel.

Ce principe évite trois dérives fréquentes : l'effet « couteau suisse », l'accumulation de fonctionnalités non stabilisées et la dépendance à des données ou à des intégrations indisponibles au moment du pilote.

2.2.2 Principe d'augmentation, non de substitution

Les trois solutions doivent être conçues comme des systèmes d'assistance augmentative et non comme des systèmes de remplacement de la décision, de la relation ou du discernement professionnel.
La justification est centrale dans la note : les participants rejettent toute logique de substitution du professionnel ou de la relation d'aide, et l'acceptabilité repose sur l'idée que l'IA doit amplifier le lien humain, alléger des tâches périphériques, améliorer la compréhension mutuelle et renforcer le pouvoir d'agir, jamais se substituer à l'humain dans les arbitrages sensibles.
D'un point de vue de conception, cela implique que :

- les sorties du système ont une valeur de proposition, pas de décision ;
- toute action engageante doit être validée par un humain ;
- l'outil doit pouvoir être contredit, corrigé ou ignoré ;
- l'interface doit expliciter le statut de la sortie : suggestion, reformulation, résumé, aide à l'orientation, et non verdict.

Ce principe protège à la fois l'éthique du dispositif, la sécurité juridique et la qualité de l'appropriation par les professionnels.

2.2.3 Principe de contrôlabilité

Un système fondé sur un LLM ne doit jamais être laissé en autonomie fonctionnelle complète dans un environnement social ou médico-social.
La raison est double. D'une part, les risques de biais, d'hallucinations, d'inexactitudes ou de simplifications abusives sont explicitement rappelés dans le document. D'autre part, l'un des principaux freins identifiés est la « boîte noire » algorithmique, qui détruit la confiance même lorsqu'un outil paraît utile.

La contrôlabilité doit donc être pensée à trois niveaux :

- contrôle des entrées, pour éviter l'injection brute de données sensibles ou incohérentes ;
- contrôle des traitements, pour cadrer le rôle exact du modèle ;
- contrôle des sorties, pour s'assurer que ce qui est présenté est structuré, lisible, modifiable et conforme au périmètre attendu.

Autrement dit, le modèle génératif ne constitue pas le produit ; il n'en est qu'un composant, encapsulé dans un système plus large de règles, de garde-fous et de validations.

2.2.4 Principe de minimisation des dépendances

Le socle doit rester fonctionnel même en l'absence d'intégration profonde aux logiciels métiers existants.
Ce choix est directement justifié par l'état des lieux : qualité de données hétérogène, faible interopérabilité, multiplicité des éditeurs, difficulté à déployer des architectures trop lourdes dans le secteur. La note recommande donc, dans un premier temps, des outils autonomes, à données réduites, ne nécessitant pas une infrastructure lourde.
Techniquement, cela signifie que la première version de chaque compagnon doit pouvoir fonctionner :

- avec une base de données propre ;
- avec quelques imports/exports maîtrisés ;
- avec des interfaces simples de saisie ou de consultation ;
- sans dépendre d'un SI unifié.

Ce principe n'exclut pas l'interopérabilité ; il la diffère intelligemment. On conçoit d'abord pour être utile sans intégration lourde, puis on ouvre progressivement les connecteurs nécessaires lorsque la valeur métier est démontrée.

2.2.5 Principe d'interopérabilité native

Même si l'intégration n'est pas requise au démarrage, l'architecture doit être conçue dès le départ pour être interopérable.
La note souligne explicitement qu'une IA pensée isolément produira des effets limités et que son impact dépendra de sa capacité à s'inscrire dans un écosystème, avec standards communs, gouvernance partagée et articulation avec les systèmes existants.
C'est pourquoi le cadre commun doit imposer :

- des APIs stables ;
- des schémas de données versionnés ;
- des exports structurés ;
- une couche de services découplée du front ;
- la possibilité de brancher ultérieurement des connecteurs vers logiciels métiers, référentiels territoriaux ou services tiers.

L'enjeu est d'éviter le piège du prototype « jetable » : un outil utile mais impossible à raccorder ensuite au réel.

2.2.6 Principe de développement incrémental par noyau de valeur

Le développement doit partir d'un noyau fonctionnel extrêmement clair, puis s'enrichir uniquement après validation terrain.
La note insiste sur la progressivité, les pilotes volontaires, la co-construction et l'importance d'usages techniquement accessibles à court terme. Cela implique un mode projet séquencé :

- cadrage métier et risques ;
- prototype fonctionnel ;
- tests internes ;
- pilote terrain ;
- ajustements ;
- extension.

La bonne question n'est pas « que pourrait faire l'outil à terme ? », mais « quel plus petit périmètre démontre une valeur réelle sans créer de rejet ? ».

2.2.7 Principe de qualification multidimensionnelle

Chaque livraison doit être qualifiée selon quatre dimensions :

- fonctionnelle : l'outil fait-il ce qu'il promet ?
- ergonomique : est-il réellement utilisable dans les contraintes terrain ?
- éthique et réglementaire : respecte-t-il les garde-fous annoncés ?
- d'acceptabilité : suscite-t-il confiance, compréhension et sentiment de maîtrise ?

Cette approche est cohérente avec l'analyse du paradoxe utilité/acceptabilité : une solution techniquement « bonne » peut échouer si elle n'est pas acceptée.

2.2.8 Principe de co-construction continue et non ponctuelle

La co-construction ne doit pas être limitée à la phase d'idéation.
Le workshop a permis d'identifier irritants, fonctionnalités, risques et garde-fous, mais la note précise aussi que la confiance se construit dans la durée, par participation réelle et communication transparente. Cela implique :

- revues régulières avec utilisateurs réels ;
- tests sur cas d'usage concrets ;
- arbitrages documentés ;
- prise en compte des refus et des irritants de déploiement.

Autrement dit, le terrain ne doit pas seulement « valider » ; il doit contribuer à orienter les priorités.

2.2.9 Principe d'observabilité pour audit, pas pour surveillance

Le système doit être observable, mais cette observabilité doit être conçue pour l'exploitation, la sécurité, l'amélioration produit et l'évaluation du pilote, non pour surveiller les individus.
C'est un point critique, car le document identifie explicitement le risque de contrôle hiérarchique des pratiques et le sentiment de surveillance comme facteurs de défiance.

Les logs doivent donc être :

- proportionnés ;
- contextualisés ;
- protégés ;
- gouvernés ;
- exclus des usages RH ou disciplinaires.

3. Architecture technique cible

3.1 Objectif de l'architecture cible

L'architecture technique des Small Companions doit répondre à une tension structurante propre au projet. D'un côté, les trois cas d'usage ont besoin d'un socle suffisamment robuste pour garantir la sécurité, la traçabilité, la qualité de service et la réutilisation des briques communes. De l'autre, ils doivent rester suffisamment simples, découplés et autonomes pour pouvoir être expérimentés rapidement dans un environnement où les systèmes d'information sont hétérogènes, parfois peu interopérables, et où l'acceptabilité dépend fortement de la capacité à éviter les dispositifs perçus comme trop intrusifs, trop complexes ou trop centralisés.
L'architecture cible doit donc satisfaire simultanément six objectifs :

- permettre un développement rapide de pilotes exploitables ;
- garantir une maîtrise forte des appels IA ;
- isoler clairement les règles métier de la logique générative ;
- réduire la dépendance à des intégrations externes complexes ;
- préparer l'interopérabilité future ;
- rendre possible une industrialisation progressive sans réécriture complète.

3.2 Principe général : architecture modulaire à socle partagé

L'architecture recommandée repose sur un socle technique mutualisé sur lequel viennent se brancher des modules fonctionnels distincts correspondant à Lucie, Étienne et Hector.
Ce choix est préférable à deux alternatives extrêmes :

- développer trois applications entièrement séparées, ce qui créerait de la redondance, des coûts supplémentaires, des risques d'incohérence et une dette de maintenance importante ;
- construire d'emblée une plateforme unifiée très intégrée, ce qui serait prématuré au regard du niveau de maturité des usages, des contraintes de terrain et du risque de surconception.

Le bon compromis consiste donc à distinguer :

- un socle partagé, regroupant les composants transverses ;
- des modules métier spécifiques, responsables de la logique propre à chaque compagnon ;
- des interfaces d'intégration prévues dès le départ, mais activées progressivement selon les besoins réels.

3.3 Découpage fonctionnel de l'architecture

L'architecture cible doit être organisée en six couches logiques clairement séparées :

- couche d'interface utilisateur ;
- couche d'API et de gestion des accès ;
- couche métier ;
- couche d'orchestration IA ;
- couche de données et de persistance ;
- couche transverse d'exploitation, sécurité et observabilité.

Ce découpage n'est pas purement théorique. Il répond à une exigence opérationnelle : permettre à plusieurs équipes de travailler sans se gêner, tout en maintenant un haut niveau de contrôle sur les composants les plus sensibles, notamment les traitements IA et les données.

3.4 Couche 1 — Interfaces utilisateur

Rôle

La couche d'interface a pour fonction de recueillir les entrées utilisateur, restituer les sorties du système et permettre la validation humaine des contenus générés.
Dans ce projet, l'interface n'est pas un simple habillage. Elle joue un rôle central dans l'acceptabilité. La note rappelle que la confiance, la transparence, la possibilité de refuser, le maintien de la maîtrise humaine et la simplicité de prise en main sont des conditions non négociables.
L'interface doit donc rendre visibles plusieurs éléments :

- ce que l'utilisateur saisit ;
- ce que le système transforme ;
- ce que le système propose ;
- ce qui doit être validé ;
- ce qui sera éventuellement conservé ou partagé.

Autrement dit, l'interface doit matérialiser la promesse éthique du produit.

Exigences de conception

Les interfaces doivent respecter plusieurs exigences communes :

- sobriété : parcours courts, vocabulaire clair, faible densité cognitive ;
- traçabilité visible : statut des contenus, dernière mise à jour, version validée ;
- éditabilité : tout contenu généré doit pouvoir être relu, modifié ou refusé ;
- accessibilité : prise en charge du texte simple, du vocal si pertinent, et à terme d'adaptations FALC, pictogrammes ou traduction selon le compagnon concerné ;
- sortie actionnable : affichage sous forme de fiche, carte, étapes ou document utilisable, et non seulement comme conversation libre.

3.5 Couche 2 — API d'accès et gestion des identités

Rôle

Cette couche constitue le point d'entrée contrôlé de l'application. Elle assure :

- l'authentification ;
- l'autorisation ;
- la gestion des sessions ;
- le routage vers les services appropriés ;
- l'application de politiques de sécurité et de quotas.

Dans un système qui manipule potentiellement des données sociales ou médico-sociales, on ne peut pas laisser chaque module gérer ses accès de façon autonome. Une gestion centralisée des identités et des permissions permet d'éviter des implémentations hétérogènes et fragiles, de garantir une traçabilité homogène et de faire évoluer plus facilement les droits par rôle ou par contexte.

3.6 Couche 3 — Services métier

Rôle

Les services métier portent la logique applicative propre à chaque compagnon. Ce sont eux qui déterminent :

- les workflows autorisés ;
- les objets manipulés ;
- les règles de validation ;
- les transitions d'état ;
- les droits d'accès contextuels ;
- les conditions de partage ou d'export.

Pourquoi il faut les isoler de l'IA

Il est essentiel que les règles métier ne soient pas confiées au modèle génératif. Le LLM peut reformuler, synthétiser, classer ou simplifier dans un cadre donné. En revanche, les règles qui fondent le fonctionnement réel du service doivent rester déterministes, explicites et testables.
Ce point est fondamental dans le contexte du document. La note insiste sur le fait que les compagnons ne doivent pas décider, profiler, scorer ou remplacer l'arbitrage humain.

3.7 Couche 4 — Orchestration IA

Rôle

La couche d'orchestration IA est le composant qui pilote l'usage des modèles génératifs et, le cas échéant, d'autres briques NLP. Elle joue le rôle d'intermédiaire intelligent entre les services métier et les modèles.
Elle doit assurer :

- la construction des prompts ;
- l'injection du contexte strictement utile ;
- le choix du bon modèle ou du bon pipeline ;

- le contrôle du format de sortie ;
- le filtrage ou la normalisation des réponses ;
- la gestion des cas d'erreur, d'incertitude.

L'orchestrateur est précisément la couche qui permet de transformer un LLM potentiellement imprévisible en composant contrôlé.

3.8 Couche 5 — Données et persistance

Rôle

Cette couche gère le stockage des objets utiles au fonctionnement du service, des données nécessaires à la continuité, des traces d'exploitation et, de manière distincte, des données analytiques utiles à l'évaluation.
La note insiste très fortement sur le minimalisme des données, les craintes de captation excessive, la nécessité de durées de conservation limitées et l'importance d'éviter le sentiment de surveillance.
Il ne faut donc pas raisonner en logique de « tout historiser par défaut ». La persistance doit être guidée par la finalité :

- persister ce qui est nécessaire au service ;
- ne pas conserver ce qui n'apporte pas de valeur d'usage ;
- distinguer clairement les données actives, les archives éventuelles, les logs techniques et les données analytiques agrégées.

3.9 Couche 6 — Exploitation, sécurité et observabilité

Rôle

Cette couche transverse regroupe les fonctions nécessaires au fonctionnement réel du service en environnement de production :

- journalisation ;
- supervision ;
- alertes ;
- sécurité opérationnelle ;
- auditabilité ;
- suivi de performance ;
- suivi des usages et incidents.

La confiance repose sur la transparence, la gouvernance claire, la possibilité de documenter les erreurs et la capacité à démontrer que l'outil n'est ni un système opaque ni un instrument de contrôle.

4. Principes non négociables

Les principes suivants doivent être respectés dans toute implémentation :

- Aucun appel direct du front vers le LLM
- Aucune logique métier critique dans le prompt seul
- Validation humaine obligatoire sur toute sortie engageante
- Séparation stricte entre données produit, données d'audit et données analytiques
- Dépendances externes encapsulées pour éviter le verrou fournisseur
- Observabilité native des coûts, erreurs, latence et corrections humaines

5. Choix de stack recommandé

5.1 Option pragmatique MVP / pilote

- Front : Next.js
- Backend métier : NestJS ou FastAPI
- Orchestrateur IA : FastAPI + Pydantic + Redis
- Base de données : PostgreSQL (RDS)
- Stockage : S3
- Cache : Redis
- Déploiement : ECS Fargate
- IAM / Auth : Auth0 ou Cognito
- Logs/metrics : CloudWatch + Datadog
- IaC : Terraform

Pourquoi cette option : rapide à mettre en œuvre, peu de charge ops, adaptée à une équipe réduite.

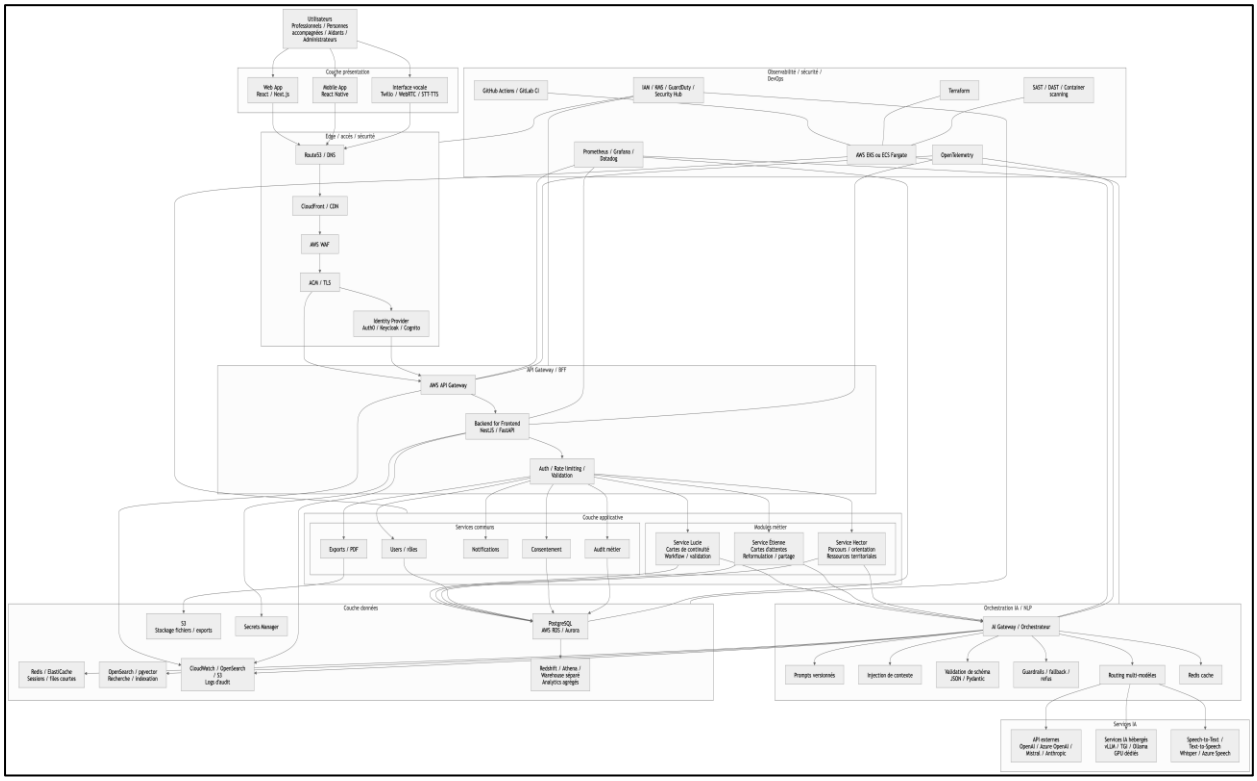
5.2 Option cible pré-industrialisation

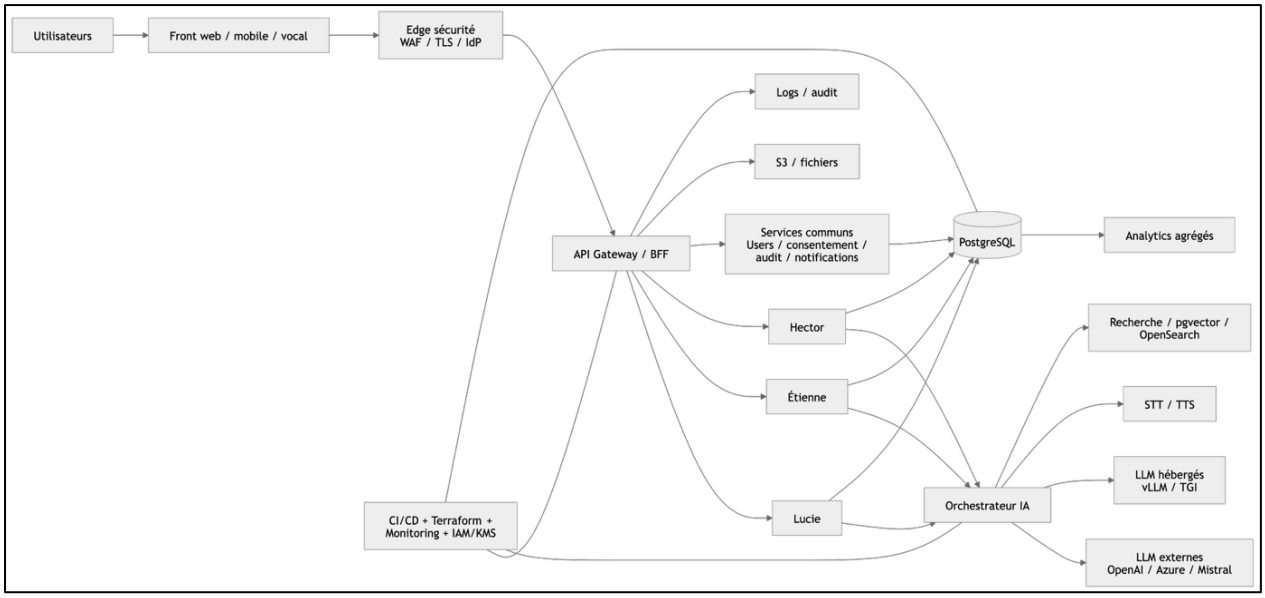
- Front : Next.js
- Backend : services NestJS/FastAPI
- Orchestrateur IA : FastAPI + LangGraph / custom orchestration
- Vector / recherche : pgvector ou OpenSearch
- Déploiement : EKS
- CI/CD : GitHub Actions + ArgoCD
- Observabilité : Prometheus + Grafana + OpenTelemetry
- Sécurité : KMS + Secrets Manager + GuardDuty
- Modèles : mix Azure OpenAI / Mistral / self-hosted

Pourquoi cette option : meilleure maîtrise du cycle de vie, scalabilité supérieure, plus adaptée à plusieurs partenaires ou territoires.

6. Schéma d'architecture technique cible

1. Architecture cible





2. Lecture architecturale

1. Couche présentation

Le choix recommandé est un front web React/Next.js, avec extension mobile si les usages terrain le justifient.
Pourquoi : rapidité de développement, mutualisation des composants UI, compatibilité avec des parcours assistés, formulaires guidés, cartes synthétiques et exports PDF. Pour Hector ou Étienne, une interface vocale pourra être ajoutée plus tard si les pilotes montrent une vraie valeur d'usage.

2. Edge / sécurité d'accès

Cette couche protège le système avant même l'entrée dans l'application. On y place le WAF, la gestion TLS, le CDN et l'identité.
Pourquoi : dans un projet manipulant des données sensibles, il faut centraliser très tôt la sécurité périmétrique et l'authentification, plutôt que la disperser dans chaque service.

3. API Gateway / BFF

Cette couche joue le rôle de point d'entrée unique.
Pourquoi : elle permet de gérer le routage, la validation des requêtes, les quotas, la journalisation et l'authentification de manière cohérente. Le BFF évite d'exposer directement les services internes au front.

4. Couche applicative

Les trois compagnons sont portés par des services métier distincts, déployés sur un environnement conteneurisé.
Recommandation :

- ECS Fargate si l'objectif est un déploiement rapide avec faible complexité ops
- EKS/Kubernetes si l'ambition est multi-environnements, multi-services, avec montée en charge et orchestration plus fine

Pourquoi : on garde un socle partagé, mais chaque compagnon conserve sa logique métier. Cela évite qu'un changement sur Hector casse Lucie, ou inversement.

5. Orchestration IA

C'est la couche la plus critique. Elle encapsule les appels LLM et garantit :

- prompts versionnés
- injection contrôlée du contexte
- validation de structure en sortie
- règles de fallback
- routage entre modèles selon usage, coût ou sensibilité

Pourquoi : le LLM ne doit jamais être appelé directement depuis le front ni porter la logique métier. L'orchestrateur permet de maîtriser la qualité, les coûts, la sécurité et la conformité.

6. Services IA

Deux options doivent rester ouvertes :

- API externes pour vitesse de mise en œuvre
- modèles hébergés pour souveraineté, confidentialité renforcée ou optimisation de coûts à terme

Pourquoi : il faut éviter un verrou fournisseur. L'architecture doit permettre de commencer vite, puis de rapatrier certaines briques si le volume ou les exigences réglementaires l'imposent.

7. Couche données

La donnée doit être séparée par usage :

- transactionnel pour le produit
- audit/logs pour la traçabilité
- analytics pour le pilotage
- objet/fichier pour documents ou exports

Pourquoi : cela évite les confusions de finalité, facilite le RGPD, améliore la sécurité et limite les dérives de réutilisation des données.

8. Observabilité / DevOps / sécurité

Cette couche rend l'exploitation réelle possible.
Pourquoi : un système qui utilise des LLM sans observabilité fine devient rapidement impossible à piloter. Il faut suivre au minimum :

- latence
- taux d'erreur
- coût par appel
- taux de fallback
- taux de validation/correction humaine
- dérives de prompt ou de format

3. Choix de stack recommandé

1. Option pragmatique MVP / pilote

- Front : Next.js
- Backend métier : NestJS ou FastAPI
- Orchestrateur IA : FastAPI + Pydantic + Redis
- Base de données : PostgreSQL (RDS)
- Stockage : S3
- Cache : Redis
- Déploiement : ECS Fargate
- IAM / Auth : Auth0 ou Cognito
- Logs/metrics : CloudWatch + Datadog
- IaC : Terraform

Pourquoi cette option : rapide à mettre en œuvre, peu de charge ops, adaptée à une équipe réduite.

2. Option cible pré-industrialisation

- Front : Next.js
- Backend : services NestJS/FastAPI
- Orchestrateur IA : FastAPI + LangGraph / custom orchestration
- Vector / recherche : pgvector ou OpenSearch
- Déploiement : EKS
- CI/CD : GitHub Actions + ArgoCD
- Observabilité : Prometheus + Grafana + OpenTelemetry
- Sécurité : KMS + Secrets Manager + GuardDuty
- Modèles : mix Azure OpenAI / Mistral / self-hosted

Pourquoi cette option : meilleure maîtrise du cycle de vie, scalabilité supérieure, plus adaptée à plusieurs partenaires ou territoires.

4. Arbitrage recommandé

1. Pour la phase 1-2

Partir sur :

- monolithe modulaire applicatif
- orchestrateur IA séparé
- ECS Fargate
- PostgreSQL + Redis + S3
- un seul cloud principal

2. Pour la phase 3-4

Évoluer vers :

- services plus découpés
- EKS si nécessaire
- multi-modèles
- connecteurs SI
- warehouse analytique séparé

5. Principes non négociables

- Aucun appel direct du front vers le LLM
- Aucune logique métier critique dans le prompt seul
- Validation humaine obligatoire sur toute sortie engageante
- Séparation stricte entre données produit, données d'audit et données analytiques
- Dépendances externes encapsulées pour éviter le verrou fournisseur
- Observabilité native des coûts, erreurs, latence et corrections humaines

PARTIE 2 : FEUILLES DE ROUTE PAR SOLUTION

1. LUCIE — Le booster de la qualité relationnelle

1.1 Synthèse exécutive

Lucie est un assistant IA conçu pour préserver la continuité relationnelle lors des changements d'intervenants dans le secteur médico-social. Elle transforme des notes éparses en une carte de continuité synthétique, utilisable en moins de 2 minutes, permettant aux professionnels remplaçants de comprendre immédiatement l'essentiel : où on en est, quoi faire maintenant, comment entrer en relation, quoi éviter.
Lucie illustre le principe de « métier augmenté » : l'IA restructure l'information existante sans prédire ni décider. Le professionnel valide, corrige et reste maître du contenu.

1.2 Problème métier

Lucie vise à résoudre un problème structurel identifié dans l'ensemble des cas analysés : la dégradation de la continuité relationnelle et informationnelle dans les parcours de prise en charge.
Les professionnels produisent une quantité importante d'informations (notes, transmissions, comptes rendus), mais ces informations :

- sont hétérogènes ;
- sont difficilement exploitables rapidement ;
- se perdent lors des transitions ;
- ne permettent pas toujours de comprendre les besoins essentiels d'une personne.

Lucie n'a pas vocation à produire de nouvelles données, mais à transformer l'existant en information utile, lisible, partageable et actionnable.

1.3 Proposition de valeur

La valeur apportée par Lucie repose sur sa capacité à produire, à partir de contenus existants, une synthèse structurée qui facilite la compréhension et la transmission.
Concrètement, Lucie permet de générer une « carte de continuité » qui organise l'information selon des catégories stables : situation actuelle, besoins prioritaires, points de vigilance, éléments relationnels et actions en cours. Cette structuration permet une lecture rapide et une appropriation immédiate, sans nécessiter la consultation exhaustive des documents sources.
L'IA intervient ici comme un outil de transformation, chargé de reformuler, d'extraire et d'organiser. Elle ne produit pas de jugement et ne prend pas de décision. La responsabilité de validation reste entièrement du côté de l'utilisateur, qui peut modifier, compléter ou rejeter la proposition.
Cette approche permet de concilier gain de temps, amélioration de la qualité d'information et maintien du contrôle humain.

1.4 Périmètre fonctionnel

MVP (phase pilote)

Dans sa version initiale, Lucie est conçue comme un outil simple, centré sur un nombre limité de fonctionnalités à forte valeur.
L'utilisateur peut soumettre un contenu sous forme de texte libre, issu par exemple de notes existantes ou d'un compte rendu. Ce contenu est traité par le système afin d'en extraire les éléments structurants, qui sont ensuite présentés sous forme d'une carte synthétique. Cette carte constitue l'objet central du produit.
L'utilisateur dispose ensuite de la possibilité de relire, modifier et valider cette carte. Ce moment de validation est une étape essentielle du fonctionnement, car il garantit que le contenu final correspond bien à la réalité de la situation. Une fois validée, la carte peut être conservée, mise à jour ou partagée dans un cadre contrôlé.
Fonctionnalités incluses :

- Soumission de texte libre (notes, comptes rendus)
- Restructuration automatique en carte synthétique
- Possibilité de relire, modifier et valider
- Conservation et mise à jour de la carte
- Partage contrôlé avec d'autres professionnels

Version étendue (phase industrialisation)

Dans une phase ultérieure, le périmètre pourra être étendu pour intégrer :

- Fonctionnalités collaboratives (plusieurs professionnels contribuent à la même carte)
- Gestion plus fine des droits d'accès par rôle
- Intégration avec systèmes existants (DPI, logiciels métiers)
- Historisation des versions de la carte
- Alertes automatiques en cas de carte obsolète

Ces évolutions ne sont cependant pas nécessaires pour démontrer la valeur du produit dans un premier temps.

1.5 Structure de la carte Lucie (output cible)

La sortie produite par Lucie prend la forme d'une carte structurée, dont la valeur repose précisément sur sa stabilité et sa lisibilité.
Elle s'organise autour de cinq sections constantes :

1. Situation actuelle

Restitution synthétique de la situation de la personne : contexte, évolution récente, état général. Maximum 3-4 phrases.

2. Besoins prioritaires

Les 2-3 besoins essentiels identifiés à ce stade, tels qu'ils apparaissent dans les éléments fournis.
Formulation claire et actionnable.

3. Points de vigilance

Éléments susceptibles d'avoir un impact sur la prise en charge : sujets sensibles, déclencheurs émotionnels, contraintes médicales ou comportementales. Liste courte et factuelle.

4. Éléments relationnels

Préférences ou spécificités relationnelles de la personne pour préserver la qualité du lien : ce qui rassure, ce qui inquiète, comment entrer en relation, ton à adopter. Cette section est essentielle pour éviter les impairs.

5. Actions en cours

État d'avancement de l'accompagnement : ce qui est prévu dans les prochains jours, ce qui a été initié, ce qui attend une validation ou un retour.
Ce format n'a pas vocation à être exhaustif, mais à être utile. Il doit permettre à un professionnel de comprendre rapidement une situation et d'agir en conséquence.

1.6 Architecture fonctionnelle

Le fonctionnement de Lucie repose sur un enchaînement simple, mais strictement contrôlé.
Flux détaillé :

- L'utilisateur fournit un contenu initial (notes, compte rendu)
- Le contenu est transmis au backend via API sécurisée
- Le backend qualifie la demande et détermine la tâche attendue
- La demande est transmise à l'orchestrateur IA
- L'orchestrateur construit la requête envoyée au modèle, en encadrant précisément le type de transformation attendue
- Le modèle retourne une sortie structurée (JSON avec les 5 sections)
- Cette sortie est vérifiée et formatée avant d'être présentée à l'utilisateur
- L'utilisateur peut modifier, compléter ou rejeter la proposition
- Validation finale par l'utilisateur
- Conservation selon règles définies (durée, droits d'accès)

Ce flux garantit que l'utilisateur ne dialogue jamais directement avec le modèle, et que la production finale reste sous contrôle humain.

1.7 Pipeline IA

Rôle du modèle

Dans Lucie, le LLM a une fonction unique : extraire et reformuler l'information existante selon un schéma fixe.
Il ne génère pas de contenu nouveau. Il ne fait pas de prédictions. Il ne porte pas de jugement. Il structure.

Configuration technique

Prompt type :
« Tu es un assistant qui aide les professionnels médico-sociaux à structurer leurs notes de prise en charge. À partir des notes fournies ci-dessous, extrait les informations selon exactement ces 5 catégories : Situation actuelle (3-4 phrases max), Besoins prioritaires (2-3 éléments), Points de vigilance (liste courte), Éléments relationnels (préférences, ce qui rassure/inquiète), Actions en cours (ce qui est prévu). Ne génère aucune information qui n'est pas dans les notes. Si une section est vide, indique 'Non renseigné'. Réponds uniquement en JSON avec ces 5 clés. »
Format de sortie attendu : JSON structuré avec validation de schéma côté backend.
Modèle recommandé : LLM léger type GPT-4o-mini ou Mistral 7B (coût/performance optimal pour tâche de restructuration).

Garde-fous

- Limitation de la taille du contexte injecté (max 2000 tokens)
- Vérification systématique du format de sortie (JSON valide)
- Détection des hallucinations par comparaison source/sortie
- Affichage côte-à-côte de la source et de la restructuration pour validation humaine

1.8 Données et gouvernance

Données collectées

Lucie collecte uniquement :

- Le contenu textuel fourni par le professionnel (notes)
- L'identifiant de la personne accompagnée (pseudonymisé si possible)
- L'identifiant du professionnel qui crée/modifie la carte
- Les métadonnées (date de création, date de dernière modification)

Durée de conservation

Une carte Lucie a une logique de maintien actif tant qu'elle soutient une prise en charge. Dès que la prise en charge se termine, la carte est archivée puis supprimée selon un calendrier défini :

- Conservation active : tant que la prise en charge est en cours
- Archive : 3 mois après fin de prise en charge
- Suppression définitive : après 3 mois d'archive

Droits d'accès

Accès en lecture : tout professionnel impliqué dans la prise en charge
Accès en écriture : professionnel référent uniquement
Accès en modification/suppression : professionnel référent + responsable d'équipe

1.9 Risques et mitigation

Risque 1 : Hallucination (génération d'informations inexistantes)

Mitigation :

- Prompt strict interdisant toute génération
- Affichage côte-à-côte source/carte pour vérification
- Validation humaine obligatoire avant toute sauvegarde

Risque 2 : Perte d'information essentielle

Mitigation :

- Synthèse contrôlée (pas de suppression arbitraire)
- Conservation de la source originale accessible en 1 clic
- Possibilité d'ajouter manuellement des éléments manquants

Risque 3 : Refus d'usage par les professionnels

Mitigation :

- Co-construction dès la phase de conception
- Démonstration de valeur rapide (gain de temps mesurable)
- Formation légère et accompagnement
- Possibilité de désactiver Lucie et revenir aux notes classiques

Risque 4 : Interopérabilité difficile avec SI existants

Mitigation :

- Version autonome ne nécessitant pas d'intégration immédiate
- APIs ouvertes pour connexion future
- Export possible en PDF/Word pour partage

1.10 KPI de suivi (phase pilote)

Les indicateurs de succès suivants permettront d'évaluer l'impact de Lucie :

KPI d'usage

- Nombre de cartes créées par semaine
- Taux d'actualisation des cartes (% de cartes mises à jour régulièrement)
- Temps moyen de création d'une carte (<5 min attendu) et de prise en main (<2 min attendu)

KPI d'impact

- Stabilité du référent : nombre de changements d'intervenant sur 30 jours (avant/après)
- Incidents de continuité signalés (diminution attendue)
- Confiance accrue de la personne accompagnée (enquête qualitative)

KPI de satisfaction

- Satisfaction professionnels remplaçants (échelle Likert 1-5)
- Satisfaction personnes accompagnées (échelle Likert 1-5)
- Taux d'adoption volontaire après la phase pilote

1.11 Planning de développement

Phase 1 : Cadrage et conception (2 mois)

- Ateliers de co-construction avec 3-5 structures pilotes
- Validation de la structure de la carte Lucie
- Maquettes UI/UX
- Spécifications fonctionnelles détaillées

Phase 2 : Développement MVP (3-4 mois)

- Développement du socle technique (backend + orchestrateur IA)
- Développement des interfaces (web responsive)
- Tests internes avec données fictives
- Ajustements prompts et pipeline IA

Phase 3 : Pilote terrain (3 mois)

- Déploiement sur 2-3 structures volontaires
- Formation des équipes (1 session de 2h par structure)
- Accompagnement hebdomadaire
- Collecte de feedbacks et KPI
- Itérations rapides selon retours terrain

Phase 4 : Industrialisation (3 mois)

- Renforcement robustesse technique
- Documentation complète
- Préparation au déploiement élargi
- Formation de formateurs

1.12 Budget estimatif

Scénario 1 : Solution complète (moteur + application)

Développement :

- Phase 1 (cadrage) : 15-20k€
- Phase 2 (MVP) : 35-45k€
- Phase 3 (pilote) : 15-20k€
- Phase 4 (industrialisation) : 5-10k€

TOTAL développement : 70-95k€
Fonctionnement annuel :

- API LLM : 2-5k€/an
- Hébergement : 3-5k€/an
- Maintenance : 3-5k€/an

TOTAL fonctionnement : 8-15k€/an

Scénario 2 : Moteur IA seul (intégration SI existant)

Développement : 25-35k€
Fonctionnement annuel : 5-8k€/an

1.13 Facteurs clés de succès

- Démonstration de valeur immédiate sur cas réels dès les premiers jours du pilote
- Formation légère des équipes (2h maximum) avec documentation claire
- Interopérabilité progressive avec logiciels métiers existants
- Gouvernance éthique claire (qui accède, qui modifie, durée de conservation)
- Accompagnement au changement : valorisation des professionnels qui utilisent Lucie
- Amélioration continue basée sur feedbacks terrain réguliers

2. ÉTIENNE — L'auxiliaire situationnel

2.1 Finalité et positionnement

Étienne répond à une problématique différente de Lucie : non plus la structuration de l'information produite par les professionnels, mais la difficulté pour certaines personnes à formuler, structurer et exprimer leurs besoins.
Cette difficulté concerne particulièrement des publics vulnérables : personnes en situation de précarité, de handicap, de stress administratif, ou de surcharge cognitive. Dans ces situations, la demande exprimée est souvent partielle, implicite ou mal comprise, ce qui peut conduire à des réponses inadaptées.
Étienne est conçu comme un outil d'assistance à l'expression. Il ne parle pas à la place de la personne, mais l'aide à transformer une intention diffuse en une formulation claire, structurée et partageable.

2.2 Problème métier traité

Contrairement à Lucie, le problème ne vient pas d'un excès d'information, mais d'une difficulté à produire une information exploitable.
Les personnes concernées ont du mal à :

- Identifier clairement ce dont elles ont besoin
- Hiérarchiser leurs priorités (tout semble urgent ou rien ne l'est)
- Formuler leur demande de manière compréhensible pour un professionnel
- Oser exprimer leurs limites ou refus (peur du jugement)
- Se préparer à un rendez-vous important (assistante sociale, MDPH, médecin)

Cette difficulté génère :

- Des rendez-vous « circulaires » (on parle sans décider)
- Des malentendus entre la personne et le professionnel
- Du découragement et de l'abandon (« je ne sais pas quoi demander »)
- Du non-recours aux droits

2.3 Proposition de valeur

Étienne permet à une personne de passer d'une expression libre, parfois confuse ou incomplète, à une « carte d'attentes » structurée, compréhensible par un professionnel.
Cette transformation repose sur trois fonctions principales :

- Reformulation progressive : clarifier ce que la personne dit
- Structuration des éléments exprimés : organiser en catégories utiles
- Explicitation des besoins implicites : rendre visible ce qui est sous-entendu

La valeur apportée est double :

- Pour la personne : se sentir comprise et capable d'exprimer sa situation
- Pour le professionnel : recevoir une information claire et directement exploitable

Spécificité clé : Étienne est disponible 24h/24 et 7j/7, ce qui permet à la personne de préparer sa réflexion quand elle le souhaite (le soir, le week-end), sans attendre la disponibilité d'un professionnel.

2.4 Périmètre fonctionnel

MVP (phase pilote)

Étienne se présente comme une interface conversationnelle guidée, orientée vers un objectif unique : produire une carte d'attentes.
Le parcours est volontairement simple. L'utilisateur exprime librement sa situation, à l'écrit ou à l'oral. Le système propose des reformulations, pose des questions de clarification si nécessaire, puis structure progressivement les éléments.
La sortie prend la forme d'une carte validée par l'utilisateur, qu'il peut ensuite partager ou utiliser dans un parcours.
Fonctionnalités incluses :

- Saisie libre (texte ou vocal simple)
- Reformulation assistée en langage accessible (FALC)
- Structuration en catégories simples (besoins, urgence, limites)
- Validation utilisateur obligatoire
- Possibilité de non-conservation (mode éphémère)
- Export en PDF ou impression pour préparer un rendez-vous

Version étendue (phase industrialisation)

Dans une phase ultérieure :

- Multilingue : traduction automatique pour personnes non francophones
- Support pictogrammes pour personnes avec handicap cognitif
- Intégration avec systèmes de prise de rendez-vous (partage direct de la carte)
- Historique des cartes pour suivre l'évolution des besoins
- Mode collaboratif : la personne peut inviter un aidant à co-construire la carte

2.5 Structure de la carte d'attentes

La carte produite par Étienne doit rester simple et compréhensible. Elle s'organise autour de cinq éléments :

1. Mes 3 besoins prioritaires

Les 3 choses les plus importantes que la personne attend. Formulation courte et claire.

2. Mon urgence

Dans quel délai la personne a besoin d'une réponse ou d'une action : 72h / 7 jours / 30 jours / Pas d'urgence.

3. Mes limites/refus

Ce que la personne ne veut pas, ce qu'elle refuse, ses lignes rouges. Cette section est essentielle pour respecter l'autonomie.

4. Mes 2 questions à poser

Les questions que la personne veut absolument poser au professionnel. Cela évite les oublis en rendez-vous.

5. La prochaine étape souhaitée

Ce que la personne attend concrètement comme suite : un rendez-vous, une orientation, une démarche, une information.
Cette structure doit être suffisamment standardisée pour être exploitable par un professionnel, tout en restant fidèle à l'expression de la personne.

2.6 Pipeline IA

Le pipeline est ici plus interactif que pour Lucie, car il repose sur un dialogue progressif.

Flux fonctionnel

- Expression initiale de l'utilisateur (texte ou voix)
- Transmission au backend via API sécurisée
- Analyse par l'orchestrateur IA : détection des éléments clés, identification des zones floues
- Génération d'une reformulation accessible (FALC)
- Interaction avec l'utilisateur : validation, correction, clarification
- Structuration progressive des informations dans les 5 catégories
- Génération d'une carte d'attentes structurée (JSON)
- Présentation à l'utilisateur pour validation finale
- Export ou partage (optionnel)

Spécificité technique : reformulation FALC

Étienne doit reformuler en Facile à Lire et à Comprendre (FALC) :

- Phrases courtes (maximum 15 mots)
- Un seul message par phrase
- Vocabulaire courant (pas de jargon administratif)
- Pas de négations complexes
- Utilisation du présent et de l'actif

Exemple de reformulation :
Input : « J'ai vraiment besoin qu'on m'aide à comprendre pourquoi ils ont refusé mon dossier RSA alors que j'ai tout envoyé comme ils m'ont dit la dernière fois. »
Output FALC : « Je veux comprendre pourquoi mon dossier RSA a été refusé. J'ai envoyé tous les documents demandés. »

2.7 Architecture spécifique

Spécificité clé

Étienne doit pouvoir fonctionner SANS persistance, afin de répondre à des enjeux de confiance et de confidentialité.
Deux modes :

- Mode éphémère : aucune donnée conservée après la session (idéal pour personnes méfiantes)
- Mode avec conservation : l'utilisateur choisit explicitement de sauvegarder sa carte

Cette option est essentielle pour l'acceptabilité auprès de publics vulnérables qui craignent la capture de données.

2.8 Données et gouvernance

Étienne manipule des données particulièrement sensibles, car elles sont directement issues de l'expression personnelle.

Données collectées (si conservation choisie)

- Contenu de la carte d'attentes
- Date de création
- Identifiant pseudonymisé de l'utilisateur (si compte créé)

Durée de conservation

Contrairement à Lucie, la conservation des données n'est pas nécessaire au fonctionnement du service.

- Mode éphémère : 0 jour (suppression immédiate après export)
- Mode avec conservation : 30 jours maximum, puis suppression automatique
- L'utilisateur peut supprimer sa carte à tout moment

Principe de minimisation absolue

Étienne ne collecte AUCUNE métadonnée inutile :

- Pas de géolocalisation
- Pas d'historique de navigation
- Pas de profilage comportemental
- Pas de tracking publicitaire

2.9 Risques et points de vigilance

Risque 1 : Interprétation abusive de la parole

Le principal risque d'Étienne est lié à la nature même de son usage : l'interprétation d'une parole.
Mitigation :

- L'IA clarifie ce que la personne dit, elle n'interprète jamais d'intentions cachées
- Affichage systématique de la reformulation pour validation
- Possibilité de corriger ou refuser à chaque étape
- Prompt strict : « Ne fais aucune hypothèse sur ce que la personne pense mais n'a pas dit »

Risque 2 : Enfermement dans une catégorisation

Un autre enjeu majeur est la capacité à ne pas enfermer la personne dans une catégorie, tout en proposant une structuration utile.
Mitigation :

- Structure de la carte flexible (possibilité d'ajouter des éléments libres)
- Langage ouvert (« Mes besoins » et non « Catégorie de demandeur »)
- Validation finale par la personne qui peut modifier librement

Risque 3 : Barrière technologique pour publics éloignés du numérique

Mitigation :

- Interface ultra-simple (3 clics maximum pour produire une carte)
- Support vocal pour personnes ne maîtrisant pas l'écrit
- Pictogrammes pour personnes avec difficultés cognitives
- Possibilité d'utilisation assistée (avec un aidant, un médiateur)

Risque 4 : Ton inapproprié (technocratique ou infantilisant)

Mitigation :

- Tests utilisateurs réguliers avec publics cibles
- Ajustement du prompt pour un ton respectueux et empathique
- Pas de « tu » condescendant, mais un « vous » respectueux ou neutre selon contexte

2.10 Indicateurs de succès (phase pilote)

KPI d'usage

- Nombre de cartes créées par semaine
- Taux d'utilisation en soirée/week-end (indicateur de la valeur du 24/7)
- Temps moyen d'expression (<2 min / 2-5 min / >5 min)
- Taux de complétion (% de personnes qui vont jusqu'au bout)

KPI d'impact

- Amélioration de l'expression des besoins déclarée par les utilisateurs
- Réduction du temps de rendez-vous (moins de clarifications nécessaires)
- Sentiment de pouvoir d'agir accru (échelle d'empowerment pré/post)
- Réduction de l'angoisse pré-rendez-vous

KPI de satisfaction

- Satisfaction des personnes utilisatrices (échelle Likert 1-5)
- Satisfaction des professionnels recevant les cartes (« les demandes sont plus claires »)
- Taux de recommandation (NPS)

2.11 Planning de développement

Le développement d'Étienne nécessite une attention particulière à l'UX. Le design conversationnel est ici un élément critique, plus encore que la performance technique brute.

Phase 1 : Cadrage et conception (2 mois)

- Ateliers de co-construction avec personnes en précarité + travailleurs sociaux
- Tests du parcours conversationnel (wireframes interactifs)
- Validation de la structure de la carte d'attentes
- Maquettes UI/UX accessibles (FALC, vocal, pictos)

Phase 2 : Développement MVP (4-5 mois)

- Développement interface conversationnelle
- Intégration reformulation FALC
- Développement mode éphémère vs mode avec conservation
- Tests internes avec personas
- Ajustements prompts pour ton approprié

Phase 3 : Pilote terrain (4 mois)

- Déploiement auprès de 2-3 associations/centres sociaux
- Accompagnement des médiateurs numériques
- Tests utilisateurs réguliers (sessions hebdomadaires)
- Itérations rapides selon feedbacks
- Ajustements accessibilité (vocal, pictos)

Phase 4 : Industrialisation (3 mois)

- Renforcement robustesse
- Ajout multilingue (arabe, anglais en priorité)
- Documentation complète
- Préparation déploiement élargi

2.12 Budget estimatif

Scénario 1 : Solution complète

Développement :

- Phase 1 (cadrage) : 18-25k€
- Phase 2 (MVP) : 40-50k€
- Phase 3 (pilote) : 15-20k€
- Phase 4 (industrialisation) : 8-12k€

TOTAL développement : 81-107k€

Fonctionnement annuel :

- API LLM : 3-8k€/an (selon volume)
- Hébergement : 3-5k€/an
- Maintenance : 3-6k€/an

TOTAL fonctionnement : 9-19k€/an

Scénario 2 : Moteur IA seul

Développement : 30-45k€
Fonctionnement annuel : 6-10k€/an

2.13 Facteurs clés de succès

La réussite d'Étienne repose sur un équilibre délicat entre assistance et neutralité.

- Qualité du design conversationnel : ton empathique mais jamais condescendant
- Accessibilité réelle : oral, pictos, multilingue dès la phase pilote
- Confiance dans la non-conservation des données : mode éphémère bien visible
- Coordination avec les professionnels relais : formation pour bien utiliser les cartes reçues
- Tests utilisateurs fréquents avec publics cibles (personnes en précarité réelle)
- Amélioration continue basée sur feedbacks terrain directs

L'outil doit aider sans orienter, structurer sans déformer, simplifier sans appauvrir. Il doit être accessible à des publics variés, parfois éloignés du numérique, et inspirer confiance dans son usage.
La qualité de l'expérience utilisateur, notamment dans les premières interactions, sera déterminante pour son adoption.

3. HECTOR — L'agent inclusif territorial

3.1 Finalité et positionnement

Hector répond à un problème systémique : la complexité d'accès aux droits, aux services et aux dispositifs d'accompagnement, en particulier pour les publics les plus vulnérables.
Contrairement à Lucie (information existante) et Étienne (expression du besoin), Hector intervient en aval, lorsque la personne doit agir dans un environnement institutionnel fragmenté, difficile à comprendre et à naviguer.
Les parcours sont aujourd'hui caractérisés par :

- une multiplicité d'acteurs (CAF, MDPH, Pôle emploi, CPAM, mairie, associations…)
- des règles d'éligibilité complexes et changeantes
- des démarches administratives séquencées (étape 1 bloque étape 2)
- une forte dépendance à l'accompagnement humain (services saturés)

Hector vise à jouer un rôle de guide opérationnel, capable d'orienter, structurer et accompagner la personne dans ses démarches, tout en respectant un principe fondamental : ne jamais se substituer à l'humain dans les situations complexes.

3.2 Problème métier traité

Le problème n'est pas seulement informationnel, mais systémique : il repose sur la complexité même de l'écosystème.
Les personnes sont confrontées à :

- Un système fragmenté : multiples guichets, pas de porte d'entrée unique
- Une opacité procédurale : jargon administratif, conditions d'éligibilité floues
- Des barrières d'accès : horaires contraints, procédures en ligne complexes, éloignement géographique
- Des démarches séquencées : impossible de faire l'étape 2 sans avoir terminé l'étape 1
- Un manque d'accompagnement : services sociaux saturés, délais d'attente longs

Conséquences :

- Abandon par découragement (« je ne sais pas par où commencer »)
- Accumulation d'erreurs et de retards (documents manquants, mauvais interlocuteur)
- Non-recours aux droits (estimé à 30-40% pour certains dispositifs)
- Saturation des acteurs de terrain par des demandes mal préparées

3.3 Proposition de valeur

Hector permet de transformer une situation donnée en un parcours d'action structuré, comprenant :

- les démarches pertinentes pour cette situation
- leur ordre logique
- les conditions d'accès réelles
- les actions concrètes à réaliser

Il agit comme un interprète du système administratif, en traduisant un environnement complexe en une suite d'étapes compréhensibles.
Sa valeur repose sur trois piliers :

- Simplification des parcours : transformer le maquis en chemin balisé
- Réduction des abandons : soutenir la personne à chaque étape
- Amélioration de l'accès effectif aux droits : augmenter le taux de succès des démarches

3.4 Périmètre fonctionnel

MVP (phase pilote)

Le MVP d'Hector doit rester ciblé pour éviter une complexité excessive.
Il se concentre sur :

- un nombre limité de cas d'usage (ex : accès aux aides sociales, logement, santé)
- une base de connaissances restreinte mais fiable
- des parcours guidés simples

Fonctionnement :
L'utilisateur décrit sa situation. Le système identifie les démarches pertinentes, les organise en étapes et fournit des instructions claires.
Fonctionnalités incluses :

- Identification des démarches pertinentes selon la situation
- Génération d'un parcours structuré (étapes numérotées)
- Explication simplifiée de chaque étape (langage FALC)
- Production de sorties actionnables (messages prêts à envoyer, checklists)
- Alertes en cas de situation complexe nécessitant un humain
- Redirection vers un professionnel si nécessaire

Version étendue (phase industrialisation)

Dans une phase ultérieure :

- Couverture élargie de cas d'usage (emploi, famille, justice…)
- Base de connaissances territoriale complète et actualisée
- Intégration avec systèmes partenaires (pré-remplissage de formulaires)
- Suivi de l'avancement des démarches (statut en temps réel)
- Notifications proactives (« votre dossier RSA arrive à expiration dans 30 jours »)

3.5 Structure du parcours utilisateur

Le résultat produit par Hector n'est pas une fiche, mais un parcours dynamique.
Il comprend :

1. Liste des démarches identifiées

Énumération des démarches pertinentes pour la situation (ex : demande RSA, demande APL, inscription Pôle emploi).

2. Séquence logique

Ordre dans lequel effectuer les démarches (certaines dépendent d'autres).

3. Pour chaque étape :

- Description claire de ce qu'il faut faire
- Documents nécessaires (avec aide pour les obtenir si manquants)
- Conditions d'éligibilité (vérification automatique si possible)
- Point de contact (adresse, téléphone, horaires, canal préférentiel)
- Délai estimé
- Statut d'avancement (à faire / en cours / bloqué / terminé)

4. Sorties actionnables

- Messages prêts à envoyer par mail ou formulaire en ligne
- Checklists de documents à préparer
- Scripts d'appel téléphonique si nécessaire
- Prochaine étape claire

5. Points de bascule vers l'humain

Si l'exécution est impossible sans aide (situation trop complexe, document introuvable), Hector propose 1-3 acteurs locaux qualifiés avec conditions d'accès réelles.

3.6 Pipeline IA (Hector)

Le pipeline est ici plus complexe, car il combine raisonnement procédural et accès à des données externes.

Flux fonctionnel

- Saisie de la situation utilisateur (questions guidées)
- Qualification par le service métier
- Orchestration IA : interprétation de la situation, interrogation de la base de connaissances, identification des dispositifs pertinents
- Construction d'un parcours structuré (séquençage logique)
- Génération des étapes et actions pour chaque démarche

- Vérification des incohérences (dépendances circulaires, conditions contradictoires)
- Présentation à l'utilisateur
- Validation / adaptation par l'utilisateur
- Suivi du parcours (optionnel : mise à jour des statuts)

Rôle de l'IA

Hector utilise l'IA pour :

- Interpréter la situation de la personne (extraction des critères pertinents)
- Interroger une base de connaissances procédurale (« pour obtenir X, il faut faire Y puis Z »)
- Générer des textes adaptés (messages, explications en FALC)
- Détecter les situations hors périmètre nécessitant une bascule humaine

3.7 Architecture spécifique

Spécificité clé

Dépendance forte à une base de connaissances fiable et maintenue.
La qualité du système dépend de la qualité des données externes. Contrairement à Lucie et Étienne qui travaillent sur des données fournies par l'utilisateur, Hector doit s'appuyer sur :

- Un référentiel national : règles générales (RSA, APL, AAH…)
- Un référentiel territorial : acteurs locaux, horaires, conditions spécifiques
- Une logique de mise à jour continue : les règles changent régulièrement

Architecture recommandée :

- Base de connaissances structurée (graph database type Neo4j ou base relationnelle)
- API de récupération de données actualisées (scraping contrôlé ou partenariats)
- Système de vérification de cohérence (alertes si données contradictoires)

3.8 Données et gouvernance

Hector repose sur deux types de données :

1. Données utilisateur (situation)

- Informations fournies par la personne pour qualifier sa situation
- Statut d'avancement des démarches
- Blocages rencontrés

Conservation : 90 jours maximum, puis suppression automatique

2. Données système (dispositifs, règles, acteurs)

- Référentiel national des dispositifs et règles d'éligibilité
- Référentiel territorial des acteurs locaux
- Historique des mises à jour pour traçabilité

Conservation : permanente, avec versioning

Gouvernance spécifique

La gouvernance d'Hector nécessite une coordination multi-acteurs :

- Collectivités territoriales (pour référentiel local)
- Organismes nationaux (CAF, CPAM, Pôle emploi…)
- Associations de terrain (pour validation terrain)

Un comité de gouvernance doit être créé pour :

- Valider les mises à jour du référentiel
- Arbitrer en cas de données contradictoires
- Définir les priorités d'extension de périmètre

3.9 Risques et points de vigilance

Risque 1 : Données obsolètes ou incomplètes

Risque majeur : donner de mauvaises informations (adresse fermée, règle changée).
Mitigation :

- Système de vérification régulière (scraping automatisé + vérification humaine)
- Date de dernière mise à jour visible sur chaque information
- Possibilité pour utilisateurs de signaler une erreur
- Bascule vers humain en cas de doute (« cette information date de X mois, vérifiez auprès de… »)

Risque 2 : Vouloir automatiser un système fondamentalement complexe

Hector ne peut pas résoudre TOUTES les situations. Certaines sont trop complexes, ambiguës ou nécessitent un arbitrage humain.
Mitigation :

- Définir clairement le périmètre couvert (« Hector aide pour les démarches courantes, pas pour les situations exceptionnelles »)
- Détection des situations hors périmètre
- Bascule systématique vers un professionnel pour cas complexes

Risque 3 : Créer de faux espoirs

Risque de générer des attentes irréalistes (« Hector dit que j'ai droit à X mais en fait non »).
Mitigation :

- Langage prudent (« vous semblez éligible à… » et non « vous avez droit à… »)
- Toujours indiquer les conditions exactes d'éligibilité
- Rappeler systématiquement la nécessité de validation par l'organisme compétent

3.10 Indicateurs de succès (phase pilote)

KPI d'usage

- Nombre de parcours générés par semaine
- Taux de complétion (% de personnes qui suivent le parcours jusqu'au bout)
- Taux de bascule vers humain (indicateur de complexité rencontrée)

KPI d'impact

- Taux de non-recours aux dispositifs ciblés (avant/après, étude de cohorte)
- Taux d'orientation correcte (% de personnes arrivant au bon interlocuteur du premier coup)
- Temps moyen d'accès au droit (de la première demande à l'obtention)
- Charge mentale allégée (enquête qualitative)

KPI de satisfaction

- Satisfaction utilisateurs (échelle Likert 1-5)
- Satisfaction professionnels relais (« les demandes sont mieux préparées »)
- Taux de signalement d'erreurs (indicateur de qualité du référentiel)

3.11 Planning de développement

Le développement d'Hector est plus progressif et dépendant des données.

Phase 1 : Cadrage et constitution du référentiel (3-4 mois)

- Sélection de 3-5 cas d'usage prioritaires (ex : RSA, APL, aide au logement)
- Constitution du référentiel national pour ces cas (règles, conditions)
- Constitution du référentiel territorial sur 1-2 territoires pilotes
- Validation avec acteurs locaux (CAF, CPAM, associations)
- Maquettes UI/UX

Phase 2 : Développement MVP (5-6 mois)

- Développement de la base de connaissances structurée
- Développement du moteur de parcours (séquençage logique)
- Développement interface guidée
- Intégration orchestrateur IA pour génération de textes
- Tests internes avec scénarios réels

Phase 3 : Pilote terrain (4-6 mois)

- Déploiement sur 1-2 territoires pilotes
- Accompagnement des acteurs locaux (formation)
- Suivi rapproché des parcours utilisateurs
- Ajustements référentiel selon retours terrain/ Extension progressive des cas d'usage couverts

Phase 4 : Industrialisation (4-5 mois)

- Renforcement robustesse
- Extension à d'autres territoires
- Automatisation de la mise à jour du référentiel
- Intégration avec systèmes partenaires (API CAF, Pôle emploi…)

3.12 Budget estimatif

Scénario 1 : Solution complète

Développement :

- Phase 1 (cadrage + référentiel) : 30-40k€
- Phase 2 (MVP) : 50-70k€
- Phase 3 (pilote) : 20-30k€
- Phase 4 (industrialisation) : 15-20k€

TOTAL développement : 115-160k€

Fonctionnement annuel :

- API LLM : 3-8k€/an
- Hébergement : 5-8k€/an
- Maintenance référentiel : 8-15k€/an (crucial !)
- Maintenance technique : 5-8k€/an

TOTAL fonctionnement : 21-39k€/an

Scénario 2 : Moteur IA seul

Développement : 40-60k€
Fonctionnement annuel : 12-20k€/an (maintenance référentiel comprise)

3.13 Facteurs clés de succès

Hector est le plus ambitieux des trois compagnons et celui dont la réussite dépend le plus de l'écosystème. Sa valeur ne repose pas uniquement sur la technologie, mais sur :

- La qualité et l'actualisation continue de la base de connaissances (investissement humain important)
- La pertinence des parcours proposés (validation terrain régulière)
- La gouvernance multi-acteurs : collectivités + organismes + associations doivent travailler ensemble
- L'articulation IA / accompagnement humain : Hector oriente, l'humain accompagne les situations complexes
- La capacité à détecter les limites : ne pas masquer la complexité mais la sécuriser

Il ne doit pas être conçu comme un « remplaçant » des services d'accompagnement, mais comme un outil d'orientation et de sécurisation des parcours, capable de réduire la complexité sans la masquer.
Le succès d'Hector dépendra avant tout de la qualité de la gouvernance territoriale et de la capacité à maintenir le référentiel à jour. Sans cela, même la meilleure technologie IA sera inefficace.

PARTIE 3 : SYNTHESE ET MISE EN OEUVRE

1. Synthèse comparative des trois solutions

Les trois solutions — Lucie, Étienne et Hector — répondent à des problématiques complémentaires au sein du parcours d'accompagnement. Elles s'inscrivent dans une logique progressive et cohérente.

1.1 Positionnement dans le parcours

Chaque compagnon intervient à un moment différent du parcours :

- Lucie : en amont ou pendant l'accompagnement, pour structurer l'information produite par les professionnels
- Étienne : au moment de l'expression du besoin par les personnes accompagnées
- Hector : en aval, lors de l'activation de parcours d'action dans l'environnement institutionnel

Cette complémentarité permet d'envisager, à terme, un écosystème cohérent couvrant la chaîne complète : comprendre → exprimer → agir.

1.2 Logique d'intervention de l'IA

- Lucie : reformatage et restructuration d'information existante
- Étienne : clarification et structuration d'une expression libre
- Hector : orientation procédurale à partir d'une base de connaissances

Dans les trois cas, l'IA n'est jamais décisionnaire. Elle transforme, clarifie ou guide, mais la validation et l'action finale restent humaines.

1.3 Tableau comparatif des trois solutions

Le tableau suivant synthétise les principales caractéristiques de chaque compagnon :

Critère	Lucie	Étienne	Hector
Problème traité	Perte de continuité lors des transitions	Difficulté à formuler ses besoins	Maquis administratif complexe
Complexité	Faible	Moyenne	Élevée
Délai MVP	3-4 mois	4-5 mois	5-6 mois
Budget complet	70-90k€	75-95k€	90-120k€
Public prioritaire	Professionnels médico-sociaux	Personnes en précarité	Aidants & personnes âgées

Ce tableau met en évidence la complémentarité des trois solutions et la logique de priorisation recommandée.

1.4 Degré d'autonomie

- Lucie : Application autonome (pas de dépendance SI externe nécessaire en MVP)
- Étienne : Application autonome (mode éphémère ou conservation optionnelle)
- Hector : Forte dépendance à un référentiel externe actualisé (partenariats nécessaires)

1.5 Public prioritaire

- Lucie : Professionnels médico-sociaux (ESMS, HAD, SSR)
- Étienne : Personnes en situation de précarité + travailleurs sociaux
- Hector : Aidants familiaux + personnes âgées + publics en insertion

1.6 Logique de priorisation recommandée

Compte tenu des éléments ci-dessus, la séquence suivante est recommandée :

1. Lucie en priorité

Raisons :

- Complexité faible = développement rapide
- Valeur immédiate = adhésion rapide des professionnels
- Risques maîtrisés = peu de dépendances
- Démonstration de faisabilité = confiance dans la suite

Objectif : générer rapidement de la valeur et de l'adhésion, sécuriser le financement pour la suite.

2. Étienne en deuxième

Raisons :

- Complexité moyenne = développement maîtrisable
- Impact fort sur la qualité d'entrée dans les parcours
- Complémentarité avec Lucie (expression + structuration)
- Tests utilisateurs intensifs nécessaires (apprentissage utile pour Hector)

Objectif : améliorer la qualité de la demande initiale, alléger la charge des professionnels.

3. Hector en troisième

Raisons :

- Complexité élevée = développement long
- Dépendance au référentiel = risque de blocage
- Nécessite coordination multi-acteurs = gouvernance lourde
- Bénéficie des apprentissages de Lucie et Étienne

Objectif : traiter la complexité systémique après avoir sécurisé les briques de base.
Cette séquence permet de sécuriser les usages, limiter les risques et construire progressivement un écosystème cohérent.

2. Hypothèses budgétaires et modèle de coûts

Les estimations budgétaires présentées ci-après reposent sur des hypothèses de développement en France, avec une équipe technique de taille réduite (2-3 développeurs + 1 chef de projet + 1 UX/UI designer).

2.1 Coûts de développement — Scénario solution complète

Ce scénario correspond au développement d'une application complète, incluant frontend, backend, orchestrateur IA, base de données et déploiement en environnement de production.
Hypothèse : équipe mixée (interne + prestataire externe), 4 phases de développement par solution.

Phase	Lucie	Étienne	Hector
Cadrage et conception	15-20k€	18-25k€	30-40k€
Développement MVP	35-45k€	40-50k€	50-70k€
Pilote terrain	15-20k€	15-20k€	20-30k€
Industrialisation	5-10k€	8-12k€	15-20k€
TOTAL	70-95k€	81-107k€	115-160k€

TOTAL développement complet des 3 solutions : 235-305k€

2.2 Coûts de développement — Scénario moteur IA seul

Ce scénario correspond au développement du moteur IA (orchestrateur + pipeline) uniquement, pour intégration dans un SI existant. L'interface utilisateur et le backend métier sont à la charge du partenaire.

Lucie (moteur seul) : 25-35k€

Étienne (moteur seul) : 25-40k€

Hector (moteur seul) : 30-50k€

TOTAL moteur IA seul : 80-125k€

2.3 Coûts de fonctionnement annuels (mutualisés)

Les coûts de fonctionnement regroupent :

- API LLM (OpenAI, Mistral ou autre)
- Hébergement et infrastructure (cloud AWS/Azure/GCP)
- Maintenance technique et évolutions

API LLM : 5-15k€/an

Hypothèse : 50 000 à 200 000 appels/mois, modèle type GPT-4o-mini ou Mistral 7B.
Répartition indicative :

- Lucie : 2-5k€/an (restructuration simple)
- Étienne : 3-8k€/an (reformulation FALC + dialogue)
- Hector : 3-8k€/an (génération de parcours)

Hébergement et infrastructure : 8-15k€/an

Hypothèse : ECS Fargate ou EKS, RDS PostgreSQL, S3, Redis, CloudWatch.
Inclut : compute, stockage, bases de données, cache, monitoring.

Maintenance et évolutions : 10-20k€/an

Hypothèse : 5-10 jours/mois de maintenance technique, corrections de bugs, petites évolutions.
Inclut : ajustements prompts, mise à jour dépendances, sécurité, support utilisateurs.
TOTAL fonctionnement annuel (3 solutions) : 23-50k€/an

3. Modalités de mise en œuvre

3.1 Organisation projet

La réussite du projet repose sur une organisation claire, avec des instances de pilotage adaptées.

Comité de pilotage (mensuel)

Rôle : suivi stratégique, validation des orientations, arbitrages budgétaires.
Composition :

- Porteur du projet (sponsor)
- Représentants des structures pilotes
- Représentants des usagers (via associations)
- Partenaires financeurs
- Responsable technique

Comité opérationnel (hebdomadaire)

Rôle : suivi de l'avancement, gestion des priorités, coordination des équipes.
Composition :

- Chef de projet
- Responsable développement
- Responsable UX/UI
- Référents métier (professionnels terrain)

Ateliers métiers (ponctuels)

Rôle : co-construction, retours utilisateurs, validation des évolutions.
Composition :

- Professionnels utilisateurs
- Personnes accompagnées (usagers finaux)
- Équipe produit

3.2 Méthodologie de développement

Le projet doit être conduit selon une méthodologie agile, avec cycles courts et validation terrain régulière.

Principes clés

- Cycles de 2 semaines (sprints)
- Livraisons fréquentes et testables
- Tests en conditions réelles dès que possible
- Amélioration continue basée sur les retours d'usage
- Validation humaine systématique des fonctionnalités sensibles

Gestion des risques

- Identification précoce des blocages techniques
- Validation des hypothèses métier avec le terrain
- Tests de charge et de sécurité dès la phase MVP
- Plan de contingence en cas de dérive budgétaire ou calendaire

3.3 Stratégie de déploiement

Le déploiement doit être progressif pour limiter les risques organisationnels et techniques.

Phase 1 : Pilotes ciblés (3-6 mois)

Déploiement sur 2-3 structures volontaires, avec accompagnement rapproché.
Objectifs :

- Valider l'adéquation produit/besoin
- Identifier les bugs et irritants d'usage
- Mesurer les KPI de succès et ajuster les fonctionnalités selon retours terrain

Phase 2 : Extension contrôlée (6-12 mois)

Déploiement sur 10-15 structures supplémentaires, avec formation standardisée.
Objectifs :

- Valider la scalabilité technique et tester le modèle de formation et d'accompagnement
- Affiner le modèle économique
- Préparer la documentation et les supports

Phase 3 : Généralisation progressive (12-24 mois)

Déploiement élargi à l'échelle d'un territoire, d'une région ou national.
Objectifs :

- Industrialisation du processus de déploiement
- Support utilisateurs à distance
- Intégration avec systèmes partenaires

3.4 Accompagnement au changement

La technologie n'est qu'un levier. Le véritable enjeu est l'appropriation par les utilisateurs.

Formation initiale

- Session de formation de 2h par solution (Lucie, Étienne, Hector)
- Format mixte : théorie (30 min) + pratique (90 min)
- Supports : vidéos, guides PDF, FAQ

Support technique

- Hotline accessible (mail + téléphone)
- Temps de réponse : <24h pour questions bloquantes
- Base de connaissances en ligne (documentation, tutoriels)

Communication transparente

- Explication claire du rôle de l'IA (ce qu'elle fait / ne fait pas)
- Publication des KPI de succès régulièrement/ Partage des retours d'expérience entre structures

Valorisation des utilisateurs

- Reconnaissance des « ambassadeurs » (professionnels moteurs)
- Partage de bonnes pratiques
- Participation aux évolutions du produit (ateliers métiers)

L'enjeu principal n'est pas la technologie, mais la confiance. Sans adhésion des professionnels et des usagers, même le meilleur outil échouera.

4. Évolutions et trajectoire cible

4.1 Logique d'évolution

Le projet repose sur une logique en trois temps :

Étape 1 : Solutions unitaires autonomes

Chaque compagnon (Lucie, Étienne, Hector) est développé et testé indépendamment, avec son propre périmètre fonctionnel. Cette approche permet de :

- Limiter les risques techniques
- Valider la valeur métier de chaque solution
- Apprendre rapidement ce qui fonctionne / ne fonctionne pas
- Sécuriser les financements étape par étape

Étape 2 : Enrichissement fonctionnel

Chaque compagnon est amélioré à partir des retours terrain :

- Ajout de fonctionnalités secondaires validées par l'usage
- Amélioration de l'ergonomie et de l'accessibilité
- Renforcement de la robustesse technique
- Extension du périmètre métier (nouveaux cas d'usage)

Étape 3 : Convergence progressive

Les trois compagnons sont progressivement articulés pour couvrir la chaîne complète : comprendre → exprimer → agir.
Cette convergence peut prendre plusieurs formes :

- Interopérabilité : les solutions échangent des données entre elles
- Point d'accès unique : l'utilisateur accède aux 3 compagnons depuis une interface commune
- Socle technique partagé : mutualisation des composants (auth, logs, orchestrateur IA)
- Fusion partielle : certaines fonctionnalités peuvent être regroupées si cela fait sens

4.2 Vision cible (horizon 3-5 ans)

À terme, les trois compagnons pourraient former un écosystème cohérent, caractérisé par :

Mutualisation des briques techniques

- Orchestrateur IA commun (gestion des prompts, des modèles, des coûts)
- Socle de sécurité partagé (authentification, autorisation, chiffrement)
- Système de logs et d'observabilité unifié

Expérience utilisateur cohérente

- Interface unifiée avec accès aux 3 compagnons selon le besoin
- Charte graphique et ton éditorial cohérents
- Parcours fluide entre les solutions

Intégration progressive dans les environnements existants

- APIs ouvertes pour connexion aux logiciels métiers (DPI, CRM)
- Exports standardisés (PDF, CSV, JSON)
- SSO (Single Sign-On) avec systèmes de gestion des identités

Articulation fluide avec les professionnels

- Les compagnons ne remplacent pas l'humain, ils lui fournissent de meilleurs outils
- Les professionnels restent maîtres des décisions, les compagnons allégeant les tâches à faible valeur ajoutée

4.3 Enjeu stratégique

La valeur du projet ne réside pas uniquement dans chaque solution prise isolément, mais dans leur capacité à fonctionner ensemble pour :

- Améliorer la qualité des parcours d'accompagnement
- Réduire les ruptures de prise en charge
- Faciliter l'accès aux droits et aux services
- Alléger la charge des professionnels sur les tâches périphériques
- Renforcer le pouvoir d'agir des personnes accompagnées

Cette ambition ne peut se réaliser que par étapes, en commençant par des solutions simples, utiles et acceptées, puis en les articulant progressivement.

5. Synthèse finale

Les solutions proposées — Lucie, Étienne et Hector — apportent une réponse cohérente et pragmatique à des besoins récurrents du secteur social et médico-social : difficulté à structurer l'information, à exprimer une demande, à naviguer dans un système complexe.
Leur complémentarité permet d'envisager, à terme, un écosystème d'outils capable d'améliorer significativement la qualité des interactions, la fluidité des parcours et l'accès effectif aux droits, tout en respectant les principes éthiques et opérationnels fondamentaux du secteur.

5.1 Principes structurants

La démarche retenue repose sur des principes structurants :

- Progressivité : partir de solutions simples et évoluer par étapes
- Co-construction : associer les professionnels et les usagers à toutes les phases
- Maîtrise des risques : identifier et mitiger les risques techniques, éthiques, organisationnels
- Éthique by design : intégrer les garde-fous dès la conception
- Transparence : expliquer clairement ce que l'IA fait et ne fait pas

5.2 Trajectoire opérationnelle

Sur le plan opérationnel, la trajectoire proposée privilégie :

- Des déploiements ciblés sur des structures volontaires
- Des cycles courts de développement et d'amélioration
- Des indicateurs de succès mesurables et partagés
- Une validation terrain systématique avant extension

Cette approche permet de démontrer la valeur dès les premières phases, tout en limitant les risques d'échec.

5.3 Socle technique mutualisé

D'un point de vue technique, le projet s'appuie sur :

- Un socle mutualisé garantissant cohérence et scalabilité
- Des modules métier distincts pour chaque compagnon
- Une architecture préparée pour l'interopérabilité future
- Une maîtrise des coûts par mutualisation des composants

5.4 Facteurs de succès

La réussite du projet reposera avant tout sur sa capacité à :

- S'inscrire dans les pratiques existantes sans les bouleverser
- Inspirer confiance par la transparence et le respect de l'autonomie
- Démontrer un bénéfice concret pour les utilisateurs (professionnels et usagers)
- Être porté par une gouvernance claire, multi-acteurs et légitime

5.5 Conclusion

Ce projet ne vise pas uniquement à introduire des solutions technologiques dans le secteur social et médico-social. Il propose une approche structurée, progressive et éthique pour améliorer durablement la qualité des parcours d'accompagnement.

En combinant innovation, pragmatisme et exigence éthique, il offre une réponse concrète à des enjeux majeurs du système — continuité relationnelle, expression des besoins, accès aux droits — tout en restant ancré dans les réalités de terrain.

L'enjeu, au-delà de la technologie, est de contribuer à un accompagnement plus fluide, plus respectueux et plus efficace, au service des personnes en situation de vulnérabilité.