

# Test technique

## «Full-Stack Senior/Confirmé(e)»

### 1. Contexte & objectif

Vous êtes chargé(e) de développer une application web permettant de composer graphiquement des workflows constitués de nœuds.

Chaque nœud référence un agent IA exposant un Endpoint au sein d'une bibliothèque organisée par familles.

L'entrée d'un workflow est un message (prompt) et la sortie est une notification e-mail indiquant l'achèvement du processus avec le détail des étapes.

### 2. Périmètre fonctionnel

#### 2.1 Dashboard + KPIs

- Liste des workflows (recherche, filtres statut : {draft, running, success, failed}).
- KPIs : Nombre de runs, Taux de succès, Durée moyenne (P50/P95), Erreurs par famille, Top agents, Distribution des durées.

#### 2.2 Bibliothèque d'agents IA

- CRUD agents (famille, version, schémas d'entrées/sorties, URL /run, secrets).
- Tri & filtres : famille, date, popularité, statut.

#### 2.3 Éditeur de workflow (drag-and-drop)

- Instanciation de nœuds depuis la bibliothèque, héritage des contrats.
- Raccordement outputs → inputs, validation de types, détection de cycles.
- Variables globales, versionnement, validation avant exécution.

#### 2.4 Playground d'exécution & test

- Playground pour lancer un run avec prompt et suivre la progression (SSE/WebSocket).
- À la fin : e-mail HTML avec statut, durée, tableau des étapes, lien vers le détail.

### 3. Contraintes techniques

- Stack :
  - Frontend : React (Vite, TS).
  - Backend : [Au choix].
  - DB : [Au choix].
- Logs en temps réel via SSE/WebSocket.
- Conteneurisation Docker (docker-compose) pour frontend, backend, DB, smtp.
- Gestion des secrets via un fichier `.env`.
- Observabilité : logs structurés corrélés (workflowId / runId).
- Tests unitaires et intégration minimaux.

### 4. Spécification des Endpoints des Agents IA

- Contrat commun : `POST /run` avec `inputs{}` et `context{}`.
- Réponses normalisées (success/error) avec métriques et logs.
- Quand un Nœud a plusieurs Endpoints, donner la main d'en choisir dans le Playground.
- Trois agents au minimum : `nlp.summarize`, `nlp.sentiment`, `utils.translate`.

Agent	Inputs	Outputs	Remarques
<b>nlp.summarize</b>	{text, max_points, language}	{summary[]}	Peut être mocké ou réel
<b>nlp.sentiment</b>	{text, language}	{label, score}	Labels standard (pos/neu/neg)
<b>utils.translate</b>	{text, toLang}	{translated}	Ex. FR→EN

### 5. Modèle de données (suggestion)

Entité	Champs (principaux)
<b>Agent</b>	id, name, family, version, schemaIn, schemaOut, endpointUrl, secrets, tags[], active, createdAt, updatedAt
<b>Workflow</b>	id, name, nodes[], edges[], version, status, createdAt, updatedAt
<b>Node</b>	id, agentId, label, config, mappingIn, mappingOut, retryPolicy
<b>Run</b>	id, workflowId, status, startedAt, endedAt, durationMs, metrics, error?
<b>RunStep</b>	id, runId, nodeId, status, startedAt, endedAt, durationMs, logs[], inputPreview, outputPreview

### 6. Règles d'exécution

- Ordonnancement : un nœud démarre quand toutes ses dépendances sont en success.
- Gestion des erreurs : fail-fast par défaut, option par nœud `onError: continue|stop`.
- Retries : `maxRetries`, `backoffMs` par nœud.
- Traçabilité : persister états, métriques, logs par step.

## 7. Sécurité & conformité

- Validation stricte des entrées (JSON Schema), limite de taille du prompt.
- Masquage des secrets en UI et logs ; bandeau de disclaimer si APIs externes.
- RBAC minimal (admin/user), CORS, rate-limiting, headers de sécurité.
- Journalisation des changements configuration et accès.

## 8. KPIs

KPI	Définition
Taux de succès	runs success / runs totaux
Durée moyenne / P95	Moyenne, percentile 95 et max par run
Top familles d'agents	Volume d'utilisation et taux d'échec
Distribution des durées	histogramme par étape
Erreurs fréquentes	agrégation par code/message

## 9. Jeux d'essai & Notification e-mail

- Seeds : 3 agents (summarize, sentiment, translate) et 2 workflows exemples.
- SMTP local MailHog ; sujet : [Workflow #id] status – durationMs. Body HTML avec tableau des étapes.

## 10. Qualité logicielle & outillage

- README clair (setup, Docker, comptes de test).
- Lint/format (ESLint/Prettier).
- Tests unitaires et intégration minimaux (ordonnancement, notification).
- Observabilité : logs JSON, correlationId.

## 11. Modalités de rendu

- Repository GitHub privé (inviter les évaluateurs).
- Inclure code frontend/back, docker-compose, README, seeds, captures d'écran, collection Postman ou OpenAPI.