République Tunisienne
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Tunis El Manar
École Nationale d'Ingénieurs de Tunis

Université Tunis El Manar

École Nationale d'Ingénieurs de Tunis

Département Technologies de l'Information et de la Communication

Rapport de stage ouvrier

Flow Signa – Parapheur électronique SaaS

Elaboré par :

El Gabsi Dhiya Eddine

Classe : 1ère année Info 2

Encadré par :

Dhia Ben Chagra

Année Universitaire : 2025/2026

Signatures

Étudiant(e)	Encadrant Professionnel

Remerciements

Je tiens à exprimer ma sincère gratitude à toutes les personnes qui ont contribué, de près ou de loin, à la réussite de ce stage d'été.

En particulier, je remercie M. Dhia Ben Chagra, mon encadrant professionnel, pour la confiance qu'il m'a accordée, sa disponibilité, ses conseils techniques et ses retours constructifs qui m'ont permis de progresser tout au long de cette expérience au sein de l'équipe de B2esis.

Je remercie également l'ensemble de l'équipe de B2esis pour son accueil chaleureux, son esprit d'entraide et les échanges enrichissants qui ont marqué mon passage.

Mes remerciements s'adressent également à l'École Nationale d'Ingénieurs de Tunis, qui m'a permis d'effectuer ce stage dans un environnement aussi stimulant et enrichissant.

Enfin, je remercie ma famille et mes amis pour leur soutien inconditionnel.

Résumé

Ce rapport présente le travail effectué dans le cadre de mon stage d'été au sein de B2esis. Il porte sur le développement du projet Flow Signa, un parapheur électronique SaaS destiné aux entreprises privées, notamment dans des contextes sensibles tels que le secteur bancaire. L'objectif du produit est de dématérialiser les circuits de validation et de signature de documents afin de réduire l'usage du papier, d'automatiser les validations, de sécuriser les signatures électroniques et d'assurer une traçabilité complète des actions réalisées.

Dans ce contexte, j'ai conçu et développé le MVP (Minimum Viable Product) complet de la plateforme : un backend Spring Boot 3 (Java 17) structuré en monolithe modulaire Maven, embarquant le moteur de workflow Flowable (BPMN 2.0) et une base de données PostgreSQL ; ainsi qu'un frontend SPA React 19 / TypeScript / Vite. Les fonctionnalités livrées couvrent l'authentification par jetons JWT (avec inscription publique validée par un administrateur), la gestion de dossiers documentaires et de leurs pièces jointes PDF, un workflow de validation puis de signature piloté par un processus BPMN, la capture de preuves de signature, un journal d'audit complet, des notifications par email, et des écrans d'administration.

L'objectif de ce stage était d'acquérir une compréhension concrète des mécanismes de dématérialisation documentaire, d'orchestration de workflow et de signature électronique, et d'appliquer ces connaissances à travers la réalisation pratique d'une plateforme complète, avec des tests d'intégration et des scénarios de bout en bout.

Mots-clés : parapheur électronique, dématérialisation, signature électronique, workflow, BPMN 2.0, Flowable, Spring Boot, React, PostgreSQL, SaaS, audit, traçabilité, JWT, sécurité.

Table des matières

Liste des figures

Liste des tableaux

Liste des acronymes

API Application Programming Interface

BDD Base de Données

BPMN Business Process Model and Notation

CI Intégration Continue

CIN Carte d'Identité Nationale

CMMN Case Management Model and Notation

CSV Comma Separated Values

DMN Decision Model and Notation

DTO Data Transfer Object

GED Gestion Électronique de Documents

HSM Hardware Security Module

IAM Identity and Access Manager

IP Internet Protocol

JPA Java Persistence API

JWT JSON Web Token

LDAP Lightweight Directory Access Protocol

MFA Authentification Multi-Facteur

MVP Minimum Viable Product

OIDC OpenID Connect

OMG Object Management Group

PAdES PDF Advanced Electronic Signature

PDF Portable Document Format

QSCD Qualified Signature Creation Device

QTSP Qualified Trust Service Provider

RBAC Role-Based Access Control

RFC Request for Comments

RGPD Règlement Général sur la Protection des Données

RLS Row Level Security

SaaS Software as a Service

SAML Security Assertion Markup Language

SI Système d'Information

SPA Single Page Application

SSO Single Sign-On

TSA Trusted Service Authority / Timestamping Authority

TS TypeScript

UI Interface Utilisateur

URL Uniform Resource Locator

UUID Universally Unique Identifier

XML eXtensible Markup Language

Introduction Générale

La transformation numérique des organisations a profondément modifié la façon dont celles-ci produisent, font circuler et conservent leurs documents. Dans les entreprises privées – et particulièrement dans les secteurs sensibles comme la banque – les circuits de validation et de signature de documents (contrats, conventions, bons de commande, factures, documents RH ou bancaires) reposent encore largement sur des processus manuels, papier, lents et difficiles à tracer. Chaque étape (dépôt, validation hiérarchique, signature, archivage) consomme du temps, génère des risques de perte ou de falsification, et rend la reconstitution d'un historique fiable quasi impossible.

C'est dans ce contexte que s'inscrit le projet Flow Signa, un parapheur électronique SaaS (Software as a Service) dont l'objectif est de dématérialiser intégralement les circuits de validation et de signature de documents PDF : réduction de l'usage du papier, automatisation des validations grâce à un moteur de workflow, sécurisation des signatures électroniques et traçabilité complète de chaque action dans un journal d'audit.

Ce rapport décrit le travail réalisé dans le cadre de mon stage au sein de B2esis, qui a consisté à concevoir et à développer le MVP de bout en bout de cette plateforme : un backend Java Spring Boot embarquant le moteur de workflow Flowable (BPMN 2.0), une base de données PostgreSQL, et un frontend React monopage, couvrant l'authentification par JWT, la gestion des dossiers et documents, le workflow validation puis signature, la capture de preuves, l'audit, les notifications email et l'administration.

Le présent rapport s'articule en quatre chapitres. Le premier chapitre présente le cadre général du stage : l'organisme d'accueil, le contexte et la problématique, puis la solution proposée et la méthodologie adoptée. Le deuxième chapitre dresse l'état de l'art des domaines mobilisés (dématérialisation et GED, workflows et BPMN, signature électronique et cadre eIDAS, moteurs de workflow, architectures web, sécurité, SaaS et multi-tenance).

Le troisième chapitre analyse les besoins puis détaille la conception du système (sécurité, dossiers, workflow Flowable, base de données). Le quatrième chapitre présente la réalisation du frontend, les tentatives d'amélioration et l'évaluation par les tests. Une conclusion générale dresse le bilan, les limites et les perspectives.

Chapitre 1

Contexte général et présentation de l'organisme d'accueil

1.1 Introduction

Ce premier chapitre présente le cadre général de mon stage. Il commence par une présentation de l'organisme d'accueil, B2esis, au sein duquel ce stage a été effectué. Nous détaillerons ensuite le contexte et la problématique ayant motivé le projet Flow Signa, avant d'exposer la solution proposée et son périmètre, puis les objectifs fixés et la méthodologie adoptée.

1.2 Présentation de l'organisme d'accueil

Dans le cadre de mon stage d'été, j'ai intégré l'entreprise B2esis. B2esis est une société qui offre des services informatiques et des solutions web et mobiles. Elle accompagne ses clients dans la conception, le développement et la mise en œuvre d'applications numériques, en s'appuyant sur une équipe technique polyvalente et un environnement de travail favorisant l'apprentissage et l'innovation. Ce stage, d'une durée de 6 semaines (juillet à mi-août), s'inscrit dans une démarche de découverte du monde de la dématérialisation et du développement logiciel à travers un projet encadré.

1.2.1 Historique

B2esis s'est développée autour des services informatiques et des solutions web et mobiles, en menant des projets pour une clientèle variée. Elle a capitalisé sur l'évolution des technologies numériques pour étendre son activité et accompagner ses clients dans leur transformation.

1.2.2 Activités

Les activités de B2esis couvrent les services informatiques et le développement de solutions numériques : applications web et mobiles, conception d'interfaces, développement backend, intégration et maintenance. C'est dans ce cadre que s'inscrit le projet Flow Signa, un parapheur électronique SaaS.

1.2.3 Présentation du département d'accueil

J'ai intégré l'équipe de développement de B2esis, en charge de la conception et de la réalisation d'applications web et mobiles. L'équipe travaille autour d'une pile technique moderne (backend Spring Boot/Java, frontend React/TypeScript), avec laquelle j'ai collaboré sur le projet Flow Signa, dans un cadre d'entraide et d'encadrement.

1.3 Présentation du projet Flow Signa

Le projet de mon stage au sein de B2esis s'inscrit dans une démarche de dématérialisation des circuits documentaires. Flow Signa est un parapheur électronique SaaS destiné aux entreprises privées, notamment dans des contextes sensibles comme le secteur bancaire. Un « parapheur » désigne, dans la terminologie métier, le circuit physique de cheminement d'un document : le dossier passe de bureau en bureau, collectant visas, validations et signatures manuscrites avant d'être finalisé et archivé. Sa dématérialisation consiste à reproduire ce circuit sous forme électronique, en y ajoutant les garanties impossibles avec le papier : automatisation des enchaînements, contrôle fin des habilitations, horodatage, preuve d'identité des signataires et journal d'audit inaltérable.

1.3.1 Contexte et enjeux

Dans les organisations où les circuits de validation restent manuels, on observe une chaîne de dysfonctionnements structurels :

1. **Lenteur et coûts :** un document doit circuler physiquement entre plusieurs intervenants (manager, direction, juridique, finance, signataire) ; chaque étape ajoute des jours de délai et un risque de perte ou d'égarement.
2. **Absence de traçabilité fiable :** une fois le dossier finalisé, il est difficile de reconstituer qui a validé quoi, quand, et dans quel ordre. Les parapheurs papier ne fournissent ni horodatage fiable ni preuve d'intégrité.
3. **Faible sécurité :** la signature manuscrite ou le scan de signature n'offrent aucune garantie cryptographique, aucune vérification d'identité forte et sont facilement falsifiables.
4. **Manque de visibilité :** le demandeur d'un document ne sait pas où en est son dossier ; les relances se font par téléphone ou email, sans file de tâches centralisée.
5. **Conservation non maîtrisée :** les documents signés sont conservés indéfiniment dans les archives physiques, sans politique de purge, ce qui pose des problèmes de conformité (RGPD, durées de conservation légales).

La problématique du projet peut donc être formulée ainsi :

> *« Comment concevoir et réaliser une plateforme web SaaS qui dématérialise de bout en bout les circuits de validation et de signature de documents PDF, en automatisant les enchaînements métier par un moteur de workflow, en garantissant la sécurité des accès et des signatures, et en assurant une traçabilité complète et exploitable de chaque action, tout en restant extensible vers les niveaux de signature avancés (eIDAS) et vers un fonctionnement multi-client ? »*

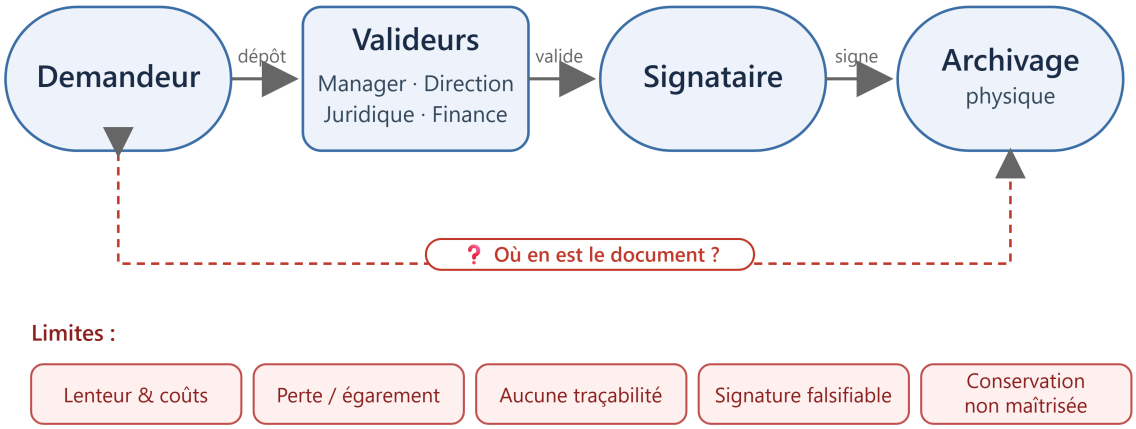

FIGURE 1.1 – Circuit de validation papier traditionnel et ses limites

1.4 Étude de l'existant

L'analyse de l'existant a porté sur deux plans : les pratiques internes des organisations (parapheurs papier ou semi-dématérialisés) et les solutions logicielles disponibles sur le marché.

1.4.1 Pratiques internes actuelles

Dans la plupart des organisations visées, le circuit de validation est soit entièrement papier, soit réalisé par email : le document est envoyé en pièce jointe, chaque intervenant répond « approuvé » ou « refusé » dans le corps du message. Ces pratiques présentent des limites majeures : aucun contrôle d'habilitation, aucune file de tâches, des enchaînements non formalisés, une traçabilité fragmentée et aucune preuve technique d'intégrité du document signé.

1.4.2 Solutions du marché

Le marché de la signature électronique est dominé par des acteurs matures, notamment : DocuSign, Adobe Sign, Yousign et Universign. Ces plateformes couvrent l'envoi de do-

cuments à signer, la signature électronique, la gestion des enchaînements de signataires et l'archivage légal. Leur comparaison met toutefois en évidence des limites pour le cas d'usage visé : coût et modèle SaaS public peu adapté à un volume interne élevé, personnalisation limitée des circuits de validation complexes, exigences d'hébergement et de souveraineté pour des clients bancaires, et intégration coûteuse au système d'information interne.

1.4.3 Enseignements

Cette étude a conduit aux choix suivants pour Flow Signa : construire une plateforme propre afin de maîtriser le modèle métier, s'appuyer sur un moteur de workflow open source éprouvé (Flowable), adopter une architecture web moderne séparée (API REST + SPA) et une pile technologique standard, et prévoir dès l'origine l'extensibilité vers la signature cryptographique et la multi-tenance.

1.5 Solution proposée

La solution proposée est une plateforme web SaaS composée de deux applications coopérantes : un backend Java (Spring Boot 3, Java 17) exposé en API REST sur le port 8080, qui embarque le moteur de workflow Flowable (formalisme BPMN 2.0), la logique métier, la sécurité et les notifications ; et un frontend SPA (React 19, TypeScript, Vite) qui offre à chaque acteur ses écrans. Le backend est structuré en monolithe modulaire Maven, persiste ses données dans PostgreSQL 17, stocke les fichiers PDF sur le filesystem et documente son API avec Swagger/OpenAPI.

Le périmètre du MVP réalisé couvre : l'authentification par jetons JWT avec inscription publique validée par un administrateur, la gestion des rôles (DEMANDEUR, VALIDEUR, SIGNATAIRE, ADMINISTRATEUR, AUDITEUR), la création de dossiers et l'upload de documents PDF, un workflow piloté par BPMN (validation puis signature avec capture de preuve), des circuits configurables, un journal d'audit, des notifications email et des écrans d'administration.

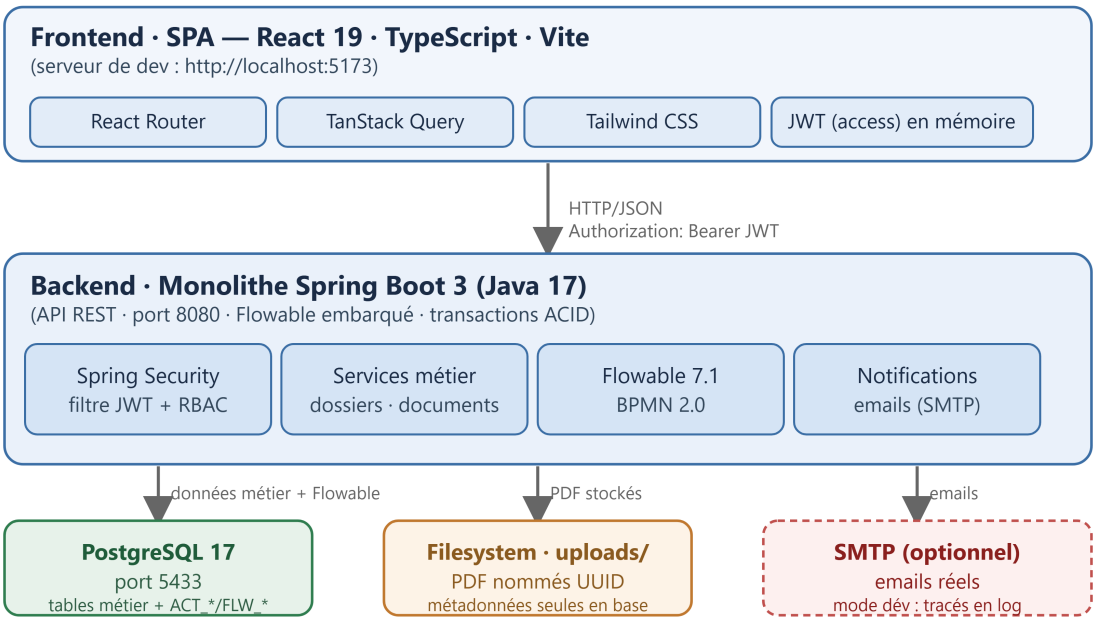

FIGURE 1.2 – Diagramme global du projet Flow Signa

1.6 Objectifs du projet

Les objectifs métier du produit, fixés par la spécification fonctionnelle de départ, sont les suivants :

- réduire l'usage du papier et les coûts associés (impression, routage, archivage physique) ;
- automatiser les validations grâce à un moteur de workflow ;
- sécuriser les signatures électroniques (d'un niveau « simple » par consentement jusqu'aux niveaux avancés et qualifiés compatibles eIDAS) ;
- assurer une traçabilité complète de chaque action réalisée sur un document.

1.7 Méthodologie adoptée

La réalisation du projet s'est appuyée sur une approche progressive, structurée en plusieurs étapes :

1. **Étude de l'existant et état de l'art :** analyse des solutions du marché, des normes (BPMN 2.0, eIDAS) et des technologies candidates.
2. **Analyse des besoins :** identification des acteurs, des besoins fonctionnels et non fonctionnels, des contraintes.
3. **Conception :** choix de l'architecture, modélisation du processus BPMN, schéma de base de données, modèle de sécurité.
4. **Implémentation :** développement du backend Spring Boot et du frontend React, intégration continue.
5. **Tests et validation :** tests d'intégration de bout en bout, évaluation des parcours nominal et secondaire, documentation.

Cette méthodologie a permis de garantir une progression cohérente du projet, en assurant une organisation claire et une traçabilité efficace des différentes étapes réalisées.

1.8 Conclusion

Ce premier chapitre a présenté le cadre général du projet mené chez B2esis, en introduisant l'organisme d'accueil, son environnement professionnel ainsi que les objectifs fixés au stage. Nous avons également décrit le contexte de la mission et souligné son importance dans le domaine de la dématérialisation documentaire. Le chapitre suivant expose les fondements théoriques et technologiques nécessaires à la compréhension de la solution.

Chapitre 2

État de l'art

2.1 Introduction

La réalisation de Flow Signa mobilise plusieurs domaines qu'il convient de maîtriser avant d'aborder la conception : la dématérialisation documentaire et les parapheurs électroniques, la modélisation et l'exécution de processus (workflows, BPMN 2.0, moteurs de workflow), la signature électronique et son cadre juridique européen eIDAS, les architectures des applications web modernes, les mécanismes d'authentification et d'autorisation, et enfin les principes du mode SaaS et de la multi-tenance. Ce chapitre passe en revue ces domaines, en privilégiant les éléments qui ont directement guidé les choix techniques du projet.

2.2 Dématérialisation et Gestion Électronique de Documents

La dématérialisation désigne le passage d'un support papier à un support numérique pour l'ensemble du cycle de vie d'un document : création, circulation, validation, signature, archivage et destruction. Elle s'appuie généralement sur une GED (Gestion Électronique de Documents), système informatique permettant d'organiser, de classer, de rechercher et de contrôler l'accès aux documents d'une organisation. Une GED apporte des béné-

fices mesurables : gain de temps de recherche, réduction des coûts d'archivage physique, sécurisation par droits d'accès, et conformité aux politiques de conservation.

Le parapheur électronique est une spécialisation de la GED centrée sur le circuit de validation : il ne s'agit pas seulement de stocker des documents, mais d'orchestrer leur cheminement entre des acteurs, avec des règles, des notifications et un historique. C'est cette spécialisation qui définit le cœur de Flow Signa.

2.3 Workflows et Modélisation de Processus (BPMN 2.0)

Un workflow (ou flux de travail) est l'automatisation d'un processus métier : il définit l'enchaînement des étapes, les acteurs responsables, les données échangées et les règles d'aiguillage. Le formalisme de référence est BPMN (Business Process Model and Notation), normalisé par l'OMG. BPMN 2.0 (2011) offre une notation graphique standard et exécutable :

- événements (start event, end event) marquant le début et la fin du processus ;
- activités : tâches humaines (user tasks) et tâches de service (service tasks, exécutées automatiquement) ;
- passerelles (gateways) exprimant les branchements conditionnels ;
- couloirs (lanes/pools) représentant les acteurs ou les systèmes ;
- variables de processus, transportées par l'instance et lues par les passerelles et les tâches.

L'intérêt majeur de BPMN 2.0 est d'être à la fois un langage de modélisation (compréhensible par les métiers) et un format exécutable (le fichier XML `.bpmn20.xml` est déployé directement dans un moteur d'exécution).

2.4 Signature Électronique

2.4.1 Les trois niveaux de signature et le cadre eIDAS

La signature électronique est encadrée en Europe par le règlement (UE) n° 910/2014, dit eIDAS (electronic IDentification, Authentication and trust Services), qui définit trois niveaux à valeur juridique croissante [1] :

1. **La signature électronique simple :** il suffit qu'un processus atteste du consentement (bouton « Signer », case de consentement). Sa valeur probante dépend des circonstances, renforcée par des éléments de contexte (identité authentifiée, horodatage, adresse IP, journal d'audit). C'est ce niveau qu'implémente le MVP de Flow Signa.
2. **La signature électronique avancée :** exige un lien exclusif avec le signataire, son identification unique, le contrôle exclusif des données de signature et la détection de toute modification ultérieure. Elle repose typiquement sur un certificat électronique et des mécanismes cryptographiques (ex. PAdES).
3. **La signature électronique qualifiée :** signature avancée créée par un dispositif qualifié de création de signature (QSCD, souvent un HSM) et apposée par un prestataire qualifié de services de confiance (QTSP).

2.4.2 Preuve, traçabilité et non-répudiation

Toute signature doit s'appuyer sur une preuve opposable : l'identité du signataire, l'horodatage de l'acte, le consentement explicite, le contexte technique (IP, agent utilisateur), l'identification du document et le journal d'audit inaltérable. La non-répudiation n'est pleinement garantie qu'au niveau cryptographique, mais une preuve applicative complète offre déjà une valeur probante exploitable dans la plupart des circuits internes. Le MVP capture cette preuve applicative dans Flow Signa.

2.5 Moteurs de Workflow Java

L'exécution de processus BPMN nécessite un moteur de workflow. Dans l'écosystème Java, le projet Activiti (créé en 2010 par Tom Baeyens, fondateur de jBPM) a fait éclater la famille Flowable (fork maintenu par une partie de l'équipe d'origine, orienté produit complet : BPM, CMMN, DMN) et la famille Camunda (fork orienté orchestration et développeurs). Les trois moteurs partagent une héritance commune (API RuntimeService/TaskService, tables ACT_*, format BPMN 2.0).

Flowable 7.1 a été retenu pour Flow Signa : starter Spring Boot 3 officiel, embarqué dans le monolithe, partage de la même base et des mêmes transactions ACID, auto-création de son schéma, et support éprouvé des delegates Spring. Camunda 8 (architecture cloud native, plus lourde) était une alternative valable mais moins directe pour un MVP embarqué.

2.6 Évolution des Architectures des Applications Web

Le monolithe historique a été critiqué pour sa rigidité à grande échelle, au profit des microservices. Une voie intermédiaire est aujourd'hui recommandée : le monolithe modulaire – une seule unité de déploiement, mais un code strictement séparé en modules aux dépendances maîtrisées. L'architecture hexagonale (ports et adaptateurs) formalise le même souci : le domaine métier ne dépend d'aucune infrastructure.

Côté frontend, le modèle dominant est la SPA (Single Page Application) : une page unique chargée une fois, qui dialogue avec le backend par appels HTTP/JSON. React est le framework le plus répandu. Les pratiques modernes associent Vite (dev server rapide), TypeScript (typage statique), React Router (routage avec gardes), TanStack Query (gestion du cache) et Tailwind CSS.

2.7 Authentification et Sécurité des Applications Web

Pour une API REST stateless, le mécanisme d'authentification dominant est le jeton JWT (JSON Web Token, RFC 7519) [3] : un jeton signé contenant des claims (identité, rôles, dates de validité), présenté par le client dans l'en-tête `Authorization: Bearer`. La bonne

pratique SPA associe un access token de courte durée (en mémoire) et un refresh token de longue durée conservé dans un cookie httpOnly. Au-delà de l'authentification, la sécurité recouvre l'autorisation : RBAC pour les droits par rôle, et autorisation métier fine.

Pour les organisations, la gestion d'identité est souvent externalisée à un IAM (Keycloak ou un fournisseur cloud), apportant SSO, OpenID Connect, fédération LDAP/SAML et MFA.

2.8 SaaS et Multi-tenance

Un logiciel SaaS sert plusieurs clients (tenants) sur la même instance. Trois modèles d'isolation s'opposent : la base commune avec discriminant (`tenant_id`), le schéma par client, et la base dédiée par client. PostgreSQL offre nativement ces trois modèles, notamment via la Row Level Security, qui offre un bon compromis pour un parapheur bancaire.

2.9 Audit et Traçabilité

L'audit informatique consiste à enregistrer, de façon append-only, les événements significatifs du système. Les principes d'un journal d'audit solide sont : complétude (chaque action métier génère une entrée), immutabilité (le journal n'est jamais modifié), contexte (acteur, action, cible, date) et exploitabilité (consultation filtrée). Dans un contexte bancaire, la traçabilité est une exigence réglementaire. Flow Signa intègre l'audit comme fonctionnalité de premier plan.

2.10 Synthèse et Positionnement du Projet

Ce parcours de l'état de l'art peut être résumé par le tableau de correspondance suivant :

Besoin du projet	État de l'art mobilisé
Dématérialiser les circuits	Parapheur électronique, GED
Orchestrer les validations	BPMN 2.0 + moteur Flowable embarqué
Signer (MVP puis eIDAS)	Signature simple par consentement, preuve applicative ; PAdES/QTSP/HSM en perspective
Authentifier	JWT (access + refresh httpOnly), bcrypt
Autoriser	RBAC + règles métier par circuit
Tracer	Journal d'audit append-only + historique par dossier
Servir plusieurs clients	SaaS multi-tenant (`tenant_id` / schéma / base) – non implanté dans le MVP
Offrir une UX moderne	SPA React + TypeScript + Vite + TanStack Query + Tailwind
Structurer pour évoluer	Monolithe modulaire + ports/adaptateurs

TABLE 2.1 – Correspondance entre les besoins de Flow Signa et l'état de l'art mobilisé

Flow Signa ne réinvente aucun de ces fondements, mais les assemble dans une solution dédiée au cas d'usage « circuit interne de validation puis signature, avec preuve et audit », cas mal couvert par les plateformes de signature à distance et par les GED généralistes.

Chapitre 3

Analyse des besoins et conception

3.1 Introduction

Ce chapitre traduit la spécification fonctionnelle en besoins explicites, puis détaille la conception du système. Il identifie d'abord les acteurs et leurs attentes, recense les besoins fonctionnels et non fonctionnels ainsi que les contraintes, puis présente l'architecture d'ensemble et la conception technique des grands blocs de la réalisation (sécurité, dossiers, workflow, base de données).

3.2 Acteurs et Parties Prenantes

La plateforme distingue cinq rôles métier implantés, plus un rôle externe prévu par la spécification. Un utilisateur possède un rôle unique.

DEMANDEUR

Personne qui crée le dossier de signature. Il peut créer un dossier, déposer un document PDF, ajouter des pièces jointes, soumettre le dossier au workflow, suivre l'avancement et annuler un dossier non encore signé.

VALIDEUR

Intervenant du circuit de validation. Il peut consulter le dossier, le valider, ou le refuser avec un motif obligatoire.

SIGNATAIRE

Personne qui signe électroniquement le document. Il peut consulter le document, confirmer son consentement obligatoire et signer électroniquement.

ADMINISTRATEUR

Gère la configuration de la plateforme : utilisateurs, circuits, demandes d'inscription, suivi des workflows, export CSV, consultation des journaux d'audit.

AUDITEUR

Rôle de consultation et de contrôle en lecture seule : historiques, preuves de signature, conformité. Il ne peut ni créer, ni valider, ni signer.

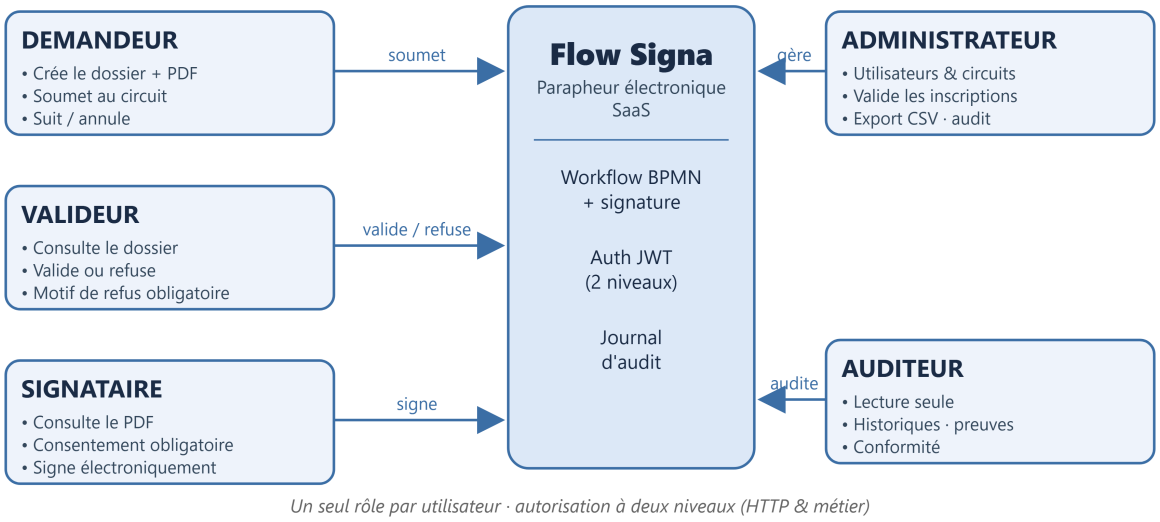

FIGURE 3.1 – Acteurs du système et leurs interactions

3.3 Besoins Fonctionnels

Les besoins fonctionnels sont regroupés par domaine et numérotés.

BF1 – Authentification et gestion des comptes

- **BF1.1** Se connecter avec un nom d'utilisateur ou un email et un mot de passe.
- **BF1.2** Se déconnecter (invalidation du jeton de rafraîchissement).
- **BF1.3** Rafraîchir automatiquement la session pendant la navigation.

- **BF1.4** S'inscrire publiquement en déposant ses pièces d'identité (CIN en PDF) ; le compte n'est créé qu'après validation par un administrateur.
- **BF1.5** Gérer les utilisateurs côté administration.

BF2 – Gestion des dossiers et des documents

- **BF2.1** Créer un dossier (titre, description, circuit).
- **BF2.2** Déposer un document PDF principal (10 Mo max).
- **BF2.3** Ajouter des pièces jointes PDF.
- **BF2.4** Consulter le détail et télécharger les documents selon les droits.
- **BF2.5** Soumettre le dossier au circuit de validation.
- **BF2.6** Annuler un dossier par son demandeur.
- **BF2.7** Suivre le statut du dossier.

BF3 – Workflow de validation et de signature

- **BF3.1** Démarrer automatiquement le processus de validation à la soumission.
- **BF3.2** Notifier les valideurs habilités.
- **BF3.3** Permettre à un valideur de valider le dossier.
- **BF3.4** Permettre à un valideur de refuser avec un motif obligatoire.
- **BF3.5** Notifier les signataires habilités.
- **BF3.6** Permettre à un signataire de signer avec consentement et capture de preuve (identité, date/heure, IP, agent utilisateur).
- **BF3.7** Notifier le demandeur des événements majeurs.

BF4 – Circuits de validation configurables

- **BF4.1** Définir des circuits (code, libellés, type de signature, options).
- **BF4.2** Affecter des valideurs et signataires par circuit, nominativement ou par groupe.
- **BF4.3** Activer/désactiver un circuit ; garantir un circuit standard par défaut.
- **BF4.4** Rattacher un dossier à un circuit à sa création.

BF5 – Audit et traçabilité

- **BF5.1** Tracer automatiquement chaque action métier dans un journal d'audit.
- **BF5.2** Consulter l'historique complet d'un dossier.
- **BF5.3** Consulter le journal d'audit global, filtré par utilisateur ou action.
- **BF5.4** Consulter la preuve de signature d'un dossier signé.

BF6 – Notifications

- **BF6.1** Notifier par email les intervenants à chaque étape.
- **BF6.2** Notifier de manière fiable sans jamais bloquer l'action métier en cas d'échec.
- **BF6.3** Respecter les préférences (utilisateurs blacklistés exclus).

BF7 – Administration et supervision

- **BF7.1** Valider ou rejeter les demandes d'inscription après consultation des pièces d'identité.
- **BF7.2** Suivre les workflows en cours.
- **BF7.3** Exporter les données de circuits en CSV.
- **BF7.4** Disposer d'une documentation interactive de l'API (Swagger/OpenAPI).

BF8 – Tableau de bord et expérience utilisateur

- **BF8.1** Tableau de bord synthétique (dossiers par statut, tâches à traiter).
- **BF8.2** Lister les dossiers selon le contexte.
- **BF8.3** Visualiser le PDF directement (visionneuse intégrée).
- **BF8.4** Restreindre l'affichage des actions selon le rôle et le statut.

BF9 – Purge automatique des documents (prévu par la spécification, non inclus dans le MVP)

- **BF9.1** Purger les documents après une durée de conservation configurable.
- **BF9.2** Conserver les métadonnées et traces d'audit au-delà de la purge.

3.4 Besoins Non Fonctionnels

BNF1 – Sécurité

- Authentification robuste : mots de passe hashés (bcrypt), jetons signés (HS256), refresh token en cookie httpOnly.
- Autorisation à deux niveaux : règles HTTP par rôle + règles métier.
- Contrôle des uploads (PDF uniquement, taille limitée) ; journalisation des échecs.

BNF2 – Traçabilité et intégrité

- Journal d'audit complet, chronologique, non modifiable ; reconstitution possible de l'historique.

BNF3 – Performance et volumétrie

- Temps de réponse compatible avec une utilisation interactive ; upload jusqu'à 10 Mo.

BNF4 – Disponibilité et résilience

- Un échec d'envoi d'email ne bloque jamais l'action métier (échec doux) ; base embarquée pour le dev.

BNF5 – Maintenabilité et évolutivité

- Code en modules aux responsabilités distinctes ; schéma de base explicite et versionné ; tests automatisés.

BNF6 – Utilisabilité

- Interface claire et orientée métier, en français ; navigation adaptée au rôle ; documentation API.

3.5 Contraintes et Règles à Respecter

Contraintes fonctionnelles :

- seule l'extension PDF est acceptée en v1 ; taille maximale des uploads : 10 Mo ;
- un utilisateur possède un seul rôle ; le motif de refus est obligatoire ;
- le consentement de signature est obligatoire ; la soumission exige au moins un PDF attaché ;

- l'annulation n'est possible qu'en brouillon ou en validation, par le demandeur.

Contraintes techniques :

- backend Spring Boot, moteur de workflow Flowable, base PostgreSQL ;
- architecture séparée API REST / SPA, communication HTTP/JSON ;
- fonctionnement local complet sous Windows, avec alternative embarquée H2 ;
- gestion des versions par Git avec intégration continue GitHub Actions.

Contraintes réglementaires et de sécurité (cible) :

- respect du règlement eIDAS pour la feuille de route de la signature ;
- durées de conservation configurables et purge des documents (cible) ;
- séparation des données des différents clients (multi-tenance).

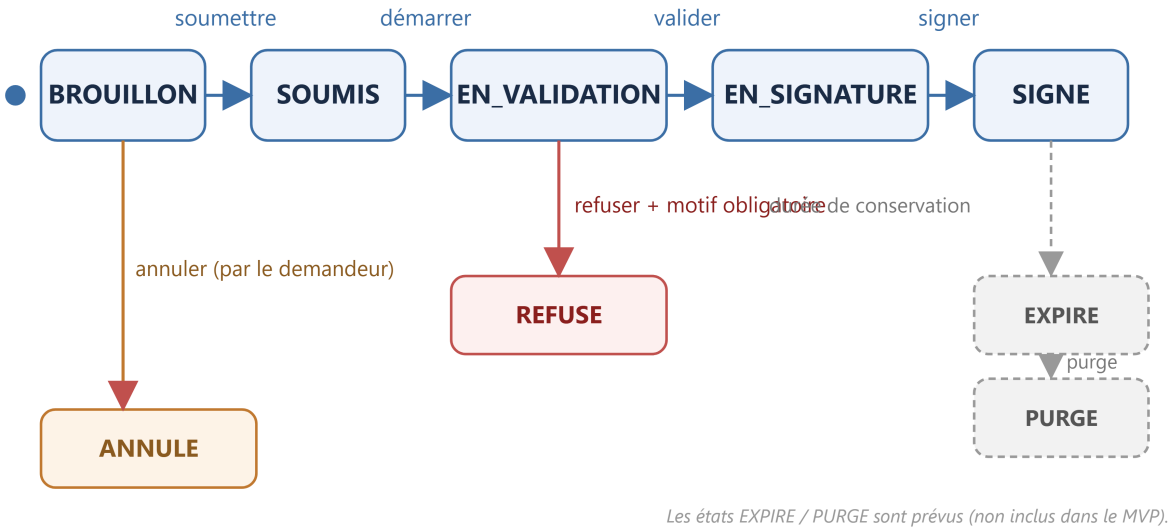

FIGURE 3.2 – Cycle de vie d'un dossier (machine à états)

3.6 Architecture d'ensemble

L'architecture réalisée est une architecture séparée classique : la SPA React/Vite tourne dans le navigateur et communique avec l'API en HTTP/JSON ; le monolithe Spring Boot expose l'API REST sur le port 8080 et embarque Spring Security, les services métier, le moteur Flowable et le service de notifications email ; PostgreSQL 17 héberge les données métier et le schéma du moteur de workflow ; les fichiers PDF sont stockés sur le filesystem, leurs métadonnées étant seules en base.

Les points structurants de cette architecture sont : un seul processus backend (Flowable embarqué, transactions ACID partagées) ; le lien entre un dossier et son instance établi par la clé métier (UUID) ; et une signature MVP fondée sur une preuve applicative.

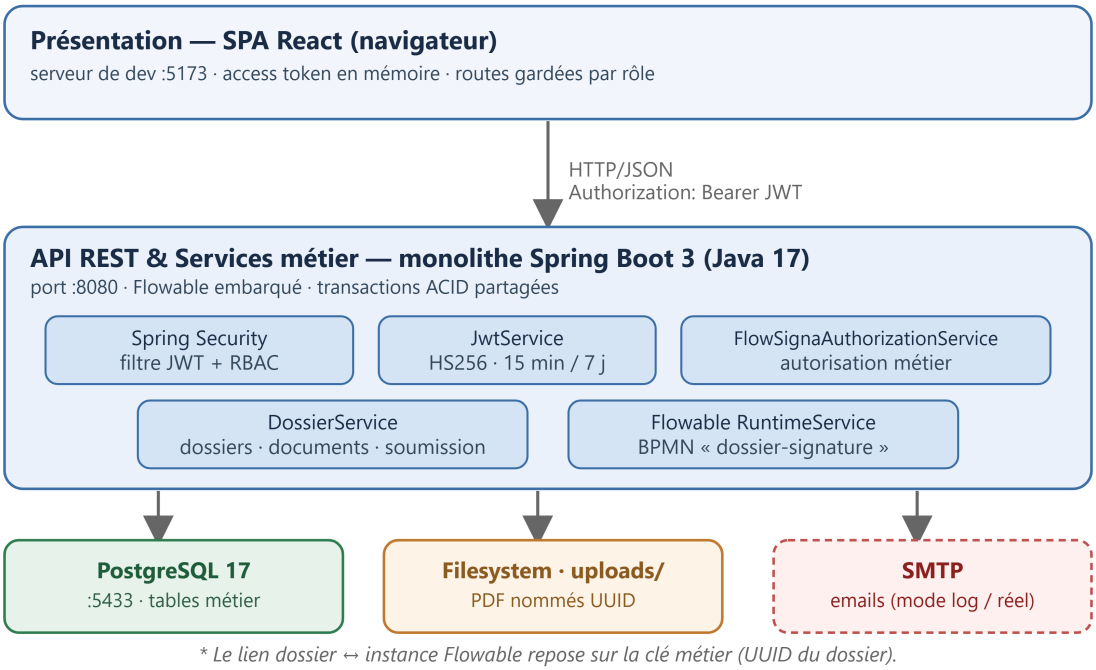

FIGURE 3.3 – Architecture d'ensemble du système

3.7 Conception du bloc sécurité et authentification

L'authentification est stateless, basée sur des jetons JWT générés et vérifiés par le service `JwtService` du module `flowsigna-security`. La signature HS256 est implémentée directement avec `javax.crypto.Mac` et Jackson, sans bibliothèque externe. Les jetons portent des claims au format Keycloak (`sub`, `preferred_username`, `email`, `roles`, `iat`, `exp`, `type`), d'une durée de 15 minutes pour l'access et 7 jours pour le refresh.

Extrait du code source (`JwtService.generateToken`) :

```
1 Map<String, Object> header = Map.of("alg", "HS256", "typ",
    "JWT");
2 Map<String, Object> claims = new LinkedHashMap<>();
3 claims.put("sub", userDetails.getUsername());
4 claims.put("preferred_username", userDetails.getUsername());
```

```
claims.put("email", userDetails.getUsername()
                                + "@flowsigna.local");
claims.put("roles", roles);           // sans prefixe ROLE_
claims.put("iat", now);
claims.put("exp", now + ttlSeconds);
claims.put("type", type);             // "access" ou "refresh"
```

L'autorisation est appliquée à deux niveaux indépendants : la couche HTTP (Spring Security, filtre JWT puis règles par endpoint) et la couche métier (`FlowSignaAuthorizationService`) qui vérifie qui peut agir sur quel dossier. Les règles HTTP sont centralisées dans `SecurityConfig` :

```
.requestMatchers("/api/admin/**").hasRole("ADMINISTRATEUR")
.requestMatchers(HttpMethod.GET, "/api/audit/**")
    .hasAnyRole("ADMINISTRATEUR", "AUDITEUR")
.requestMatchers(HttpMethod.POST, "/api/dossiers/*/valider",
        "/api/dossiers/*/refuser")
    .hasAnyRole("VALIDEUR", "SIGNATAIRE", "ADMINISTRATEUR")
.requestMatchers(HttpMethod.POST, "/api/dossiers/*/signer")
    .hasAnyRole("SIGNATAIRE", "ADMINISTRATEUR")
.requestMatchers(HttpMethod.POST, "/api/dossiers/*/annuler")
    .hasRole("DEMANDEUR")
.requestMatchers("/api/dossiers/**").authenticated()
.anyRequest().authenticated()
```

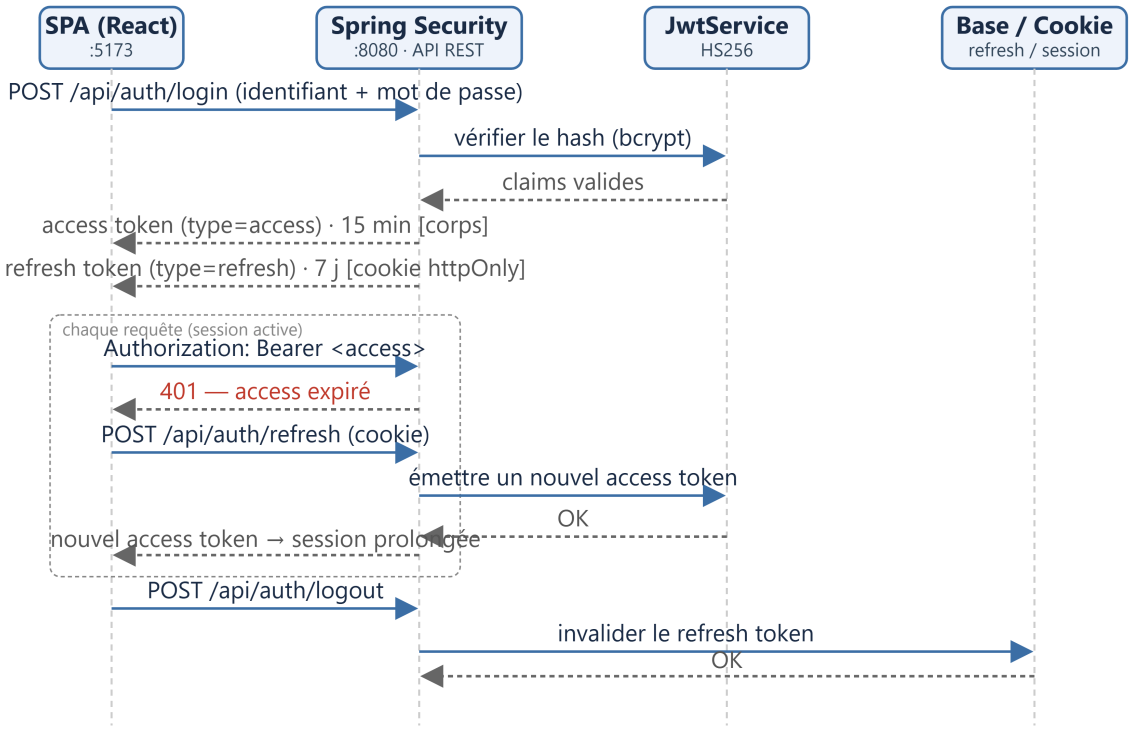

FIGURE 3.4 – Cycle login / refresh / logout (JWT et cookies)

3.8 Conception de la gestion des dossiers et documents

Le dossier (entité `Dossier`) est l'unité métier centrale : titre, description, demandeur, statut (enum à 9 valeurs), rattachement à un circuit et horodatage. Sa clé primaire est un UUID, qui sert également de clé métier auprès du moteur de workflow. Le stockage des documents s'appuie sur le filesystem : l'upload impose l'extension `.pdf`, une taille maximale de 10 Mo, et l'écriture du fichier sous un nom UUID, les métadonnées seules étant enregistrées en base.

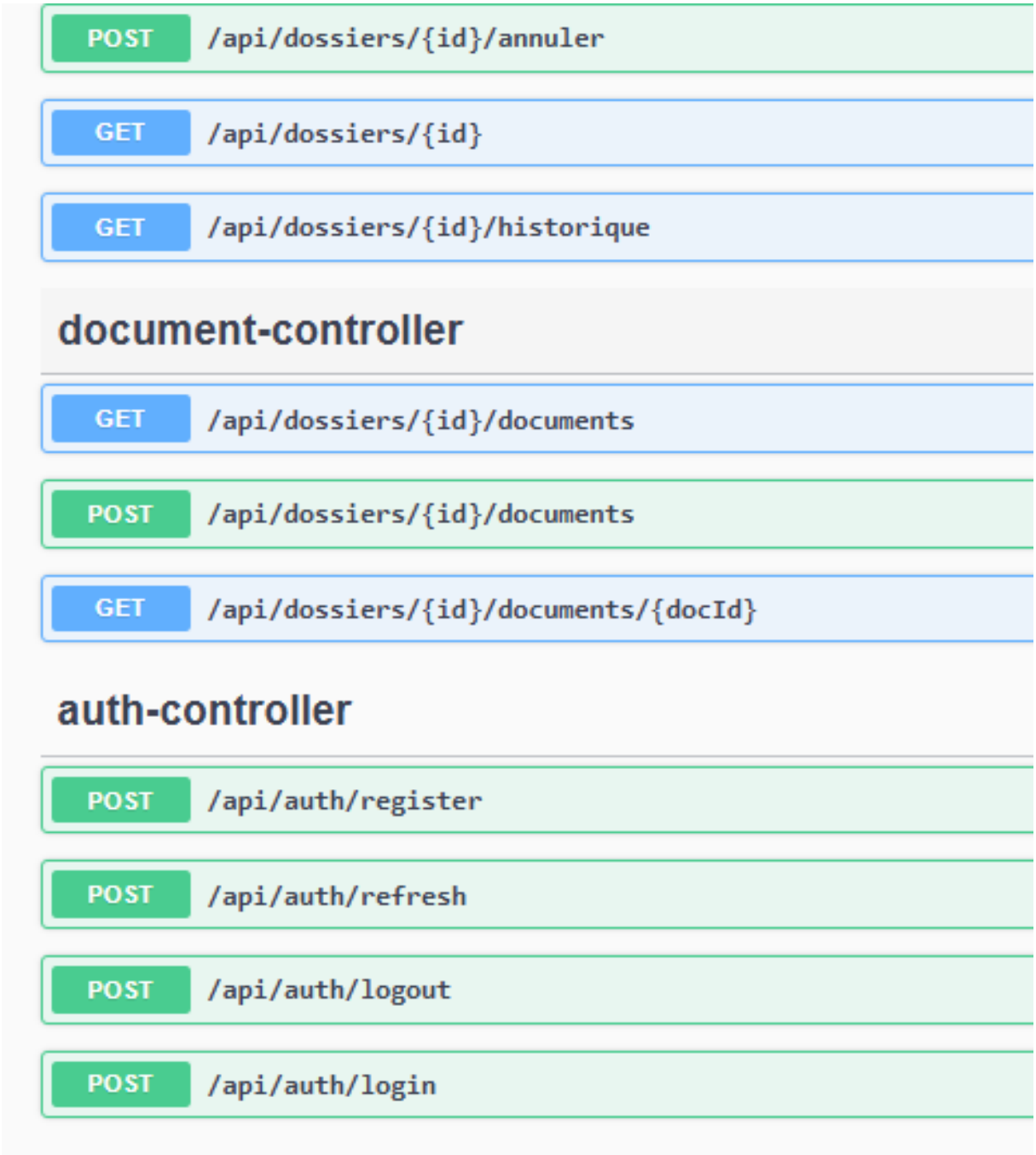

FIGURE 3.5 – Documentation interactive Swagger UI de l'API

3.9 Conception du moteur de workflow Flowable

Le moteur Flowable 7.1.0 est embarqué dans le monolithe. Au démarrage, il crée son schéma et déploie le processus défini dans `processes/dossier-signature.bpmn20.xml`. La cle du processus est « dossier-signature ». La topologie comprend : `startSoumission`, `workflowStarted` (EN_VALIDATION), `notifyValidationTask`, `taskValidation` (user

task), `gatewayValidation` (passerelle exclusive), puis les branches refus et approbation, jusqu'à `taskSignature`, `captureSignatureProof`, `markSigned`, `notifySignedToRequester`, `generateProofAndAudit` et `endSigned`.

Extrait du fichier réel (`dossier-signature.bpmn20.xml`) :

```
<userTask id="taskValidation"
          name="6. Consulter le dossier et decider"
          flowable:candidateGroups="VALIDEUR,SIGNATAIRE"/>

<exclusiveGateway id="gatewayValidation"
                  name="7. Approuver ou refuser ?"/>

<sequenceFlow id="flowRejected" sourceRef="gatewayValidation"
              targetRef="markRejected">
  <conditionExpression xsi:type="tFormalExpression">
    ${approuve == false}
  </conditionExpression>
</sequenceFlow>
```

Le couplage API-moteur est réalisé dans `DossierService` : la tâche active est retrouvée par business key, puis complétée avec les variables de preuve lors de la signature.

```
// soumettre(id) : demarrage du processus,
// business key = UUID du dossier
runtimeService.startProcessInstanceByKey(
    PROCESS_KEY,                        // "dossier-signature"
    dossier.getId().toString(),
    Map.of());

// signer(...) : completion avec les variables de preuve
completeTask(id, TASK_SIGNATURE, Map.of(
    "signe", true,
    "consentAccepted", true,
    "signedBy", signedBy,
    "signatureIp", ipAddress,
    "signatureUserAgent", userAgent,
```

```
15        "signedAt", Instant.now().toString()));
```

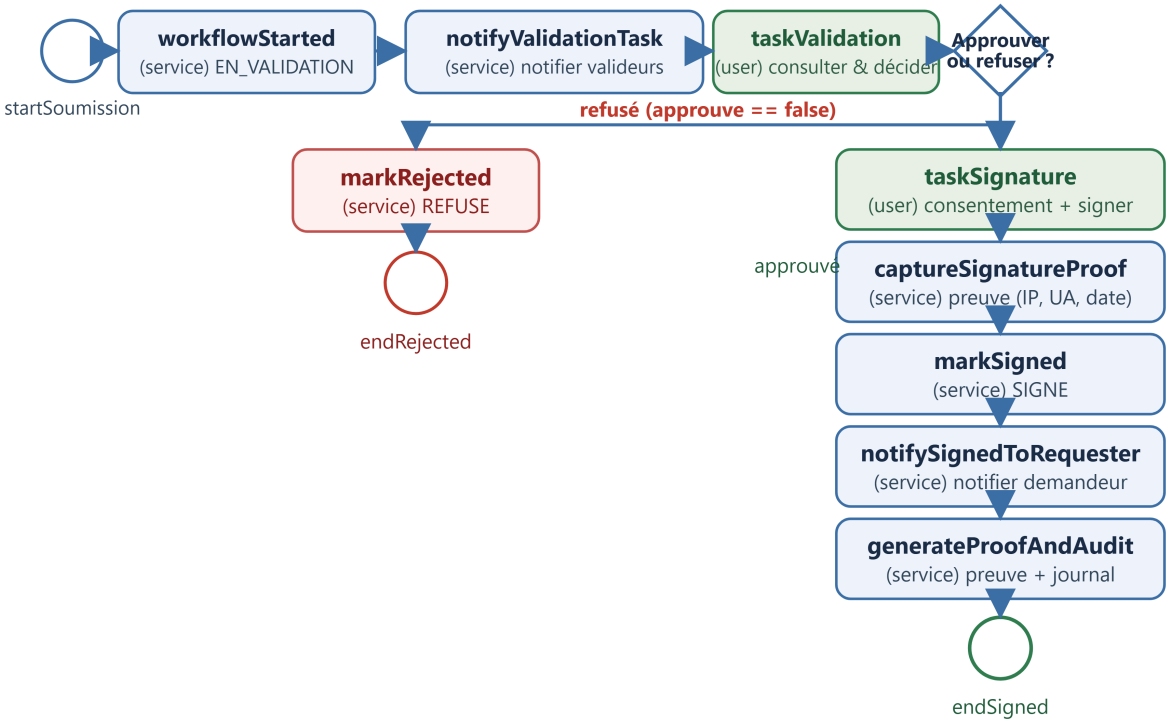

FIGURE 3.6 – Processus BPMN « dossier-signature »

3.10 Conception de la base de données

La base héberge huit tables métier créées par `schema.sql` et pilotées par huit entités JPA :

`dossiers`

id (UUID, PK, = business key), title, description, status (enum), circuit_id, requester_username, requester_email, created_at, updated_at.

`documents`

id (UUID, PK, sert aussi de nom de fichier), dossier_id, filename, content_type, file_path, uploaded_at.

`user_accounts`

username (PK), email (unique), first_name, last_name, password_hash, role (enum), blacklisted, notifications_enabled, created_at.

`circuits / circuit_assignments`

configuration des circuits (code, libellés, type de signature, options) et affectations par circuit (type_acteur, principal_type, principal_id).

`registration_requests / registration_request_documents`

demandes d'inscription publiques : identités, entreprise, mot de passe, statut, pièces d'identité PDF.

`audit_logs`

id, dossier_id, dossier_title, circuit, action, actor, details, created_at. Journal append-only.

La stratégie de gestion du schéma est explicite et versionnée : `hibernate ddl-auto: validate`, `schema.sql` exécuté au démarrage, migrations manuelles dans `migrations/`, et profil `local-h2` pour un démarrage sans infrastructure.

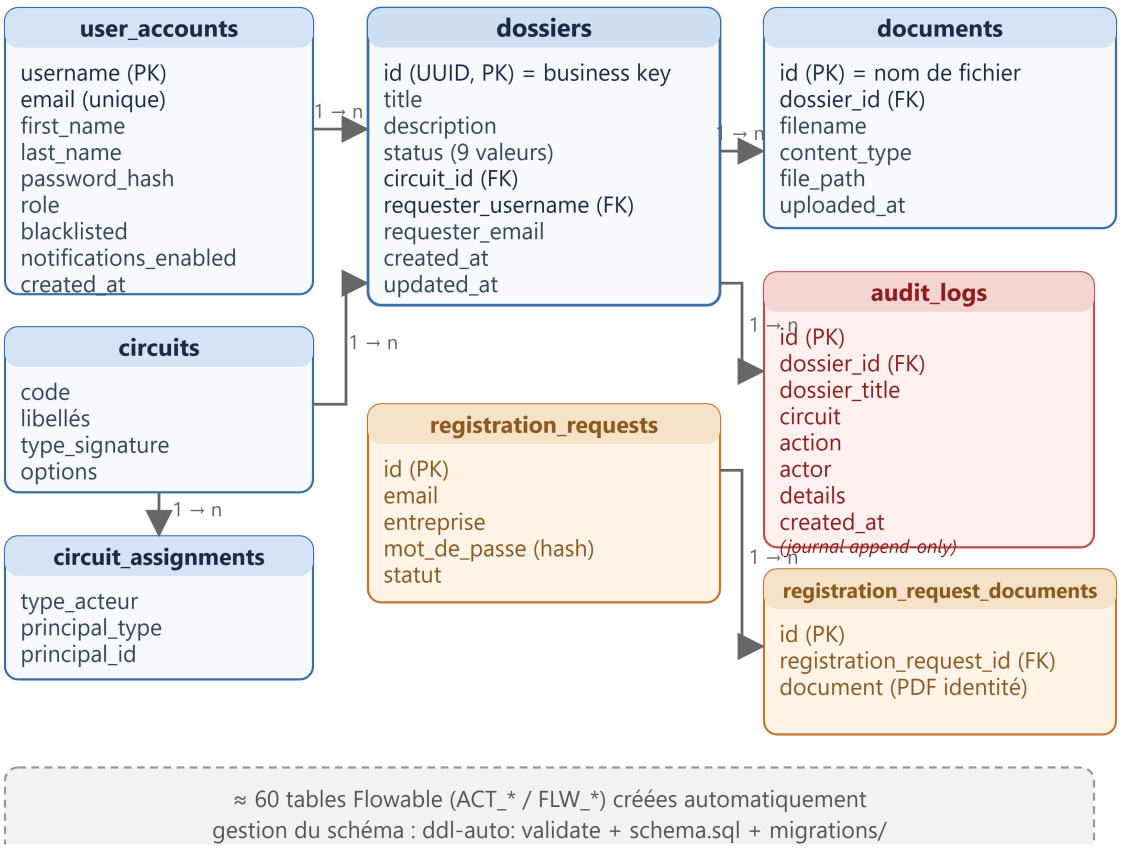

FIGURE 3.7 – Schéma relationnel simplifié de la base de données

3.11 Conclusion

Ce chapitre a présenté l'analyse des besoins (acteurs, besoins fonctionnels et non fonctionnels, contraintes) et la conception du système : architecture d'ensemble, sécurité à deux couches, gestion des dossiers et documents, moteur de workflow Flowable et base de données au schéma explicite. Le chapitre suivant détaille la réalisation du frontend et l'évaluation du système.

Chapitre 4

Réalisation, tests et évaluation

4.1 Introduction

Ce chapitre détaille la réalisation technique du système et son évaluation. Il présente l'environnement de développement, puis le frontend React, les tentatives d'amélioration menées en cours de projet, et enfin la stratégie de test et les résultats des parcours nominal et secondaire.

4.2 Environnement et prérequis de réalisation

Le développement a été réalisé sous Windows, avec l'outillage suivant.

Poste de développement

- Windows (PowerShell comme shell de référence) ;
- JDK 17+ pour le backend ; Node.js 20+ et npm pour le frontend ;
- IntelliJ IDEA (backend) et Visual Studio Code (frontend) ;
- Git + GitHub (dépôts, branches, pull requests).

Infrastructure locale

- PostgreSQL 17 sur le port 5433 (non standard) ;
- alternative embarquée H2 (profil « local-h2 »).

Commandes de référence :

```
.\mvnw clean install -DskipTests
.\mvnw spring-boot:run -pl flowsigna-api/flowsigna-app -am `
  "-Dspring-boot.run.profiles=local-h2"
npm install
npm run dev        # dev server sur http://localhost:5173
```

4.3 Le backend : monolithe modulaire Maven

Le backend est un monolithe multi-module Maven (groupe `org.flowsigna`) organisé en trois modules porteurs de code et deux agrégateurs :

```
flowsigna (pom racine)
|-- flowsigna-api (agregateur, packaging pom)
|   |-- flowsigna-app        -> point d'entree, controleurs,
|   |                           DTOs, config, tests, schema.sql
|   '-- flowsigna-security   -> JWT, authentification,
|                               inscription, CORS, emails
'-- flowsigna-core (agregateur, packaging pom)
    '-- flowsigna-model      -> entites JPA, repositories,
                                services metier, Flowable, BPMN
```

La répartition des responsabilités est la suivante :

Module	Responsabilité
`flowsigna-app`	Couche d'application : 8 contrôleurs REST, 13 DTOs, gestionnaire global d'exceptions, configuration OpenAPI/Web, initialisation du circuit standard.
`flowsigna-security`	Couche de sécurité : authentification JWT (`SecurityConfig`, `JwtService`, `JwtAuthenticationFilter`), contrôleur d'auth, inscription utilisateur, notifications par e-mail, configuration CORS.
`flowsigna-model`	Couche métier et domaine : 8 entités JPA, 8 repositories, services métier, `FlowableConfig`, 12 délégués BPMN, règles d'autorisation métier, port de notification.

TABLE 4.1 – Répartition des responsabilités entre les modules Maven

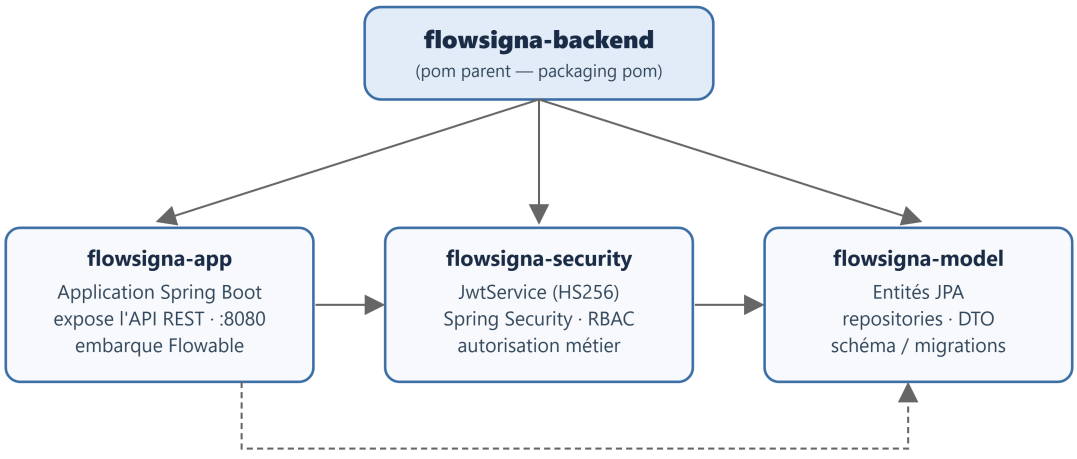

FIGURE 4.1 – Organisation en modules Maven du backend

4.4 Le frontend React

Le frontend est une SPA React 19 + TypeScript (mode strict) + Vite, organisée par couches : `api/` (un client Axios unique + modules par domaine), `app/` (routage React Router 7 + layouts + garde `ProtectedRoute`), `lib/auth/` (contexte d'authentification et magasin de token en mémoire), `lib/query-keys.ts` (clés TanStack Query) et `components/` (composants de structure, métier et document).

Extrait du code source (`src/api/client.ts`) :

```
const apiClient = axios.create({
  baseURL: import.meta.env.VITE_API_BASE_URL
      ?? 'http://localhost:8080/api',
  withCredentials: true,        // cookie de refresh
     auto-envoye
})

let refreshPromise: Promise<string | null> | null = null
// (promesse partagee : un seul appel de refresh simultane)

if (error.response?.status !== 401
    || !originalRequest || originalRequest._retry) {
  return Promise.reject(error)
}
originalRequest._retry = true
const token = await getRefreshedToken()
if (!token) {
  window.location.replace('/login')   // refresh impossible
}
setAuthorizationHeader(originalRequest, token)
return apiClient(originalRequest)     // rejeu de la requete
```

Les routes principales et leurs gardes sont les suivantes :

Route	Description / garde
`/login`, `/inscription`	publics (connexion, inscription avec dépôt des pièces CIN)
`/`	tableau de bord (authentifié)
`/dossiers/crees`	mes dossiers créés
`/dossiers/nouveau`	création (assistant)
`/dossiers/a-valider`	dossiers à valider (VALIDEUR+)
`/dossiers/a-signer`	dossiers à signer (SIGNATAIRE+)
`/dossiers/termines`	dossiers terminés (authentifié)
`/dossiers/:id`	détail du dossier
`/dossiers/:id/historique`	historique (timeline d'audit)
`/admin/utilisateurs`	administration (ADMINISTRATEUR)
`/admin/demandes-inscription`	demandes d'inscription (ADMIN)
`/admin/workflows`	suivi des parapheurs + circuits (ADMIN)
`/audit/logs`	journal d'audit (AUDITEUR, ADMIN)

TABLE 4.2 – Routes de la SPA React et leurs gardes d'accès

FIGURE 4.2 – Page de connexion du frontend

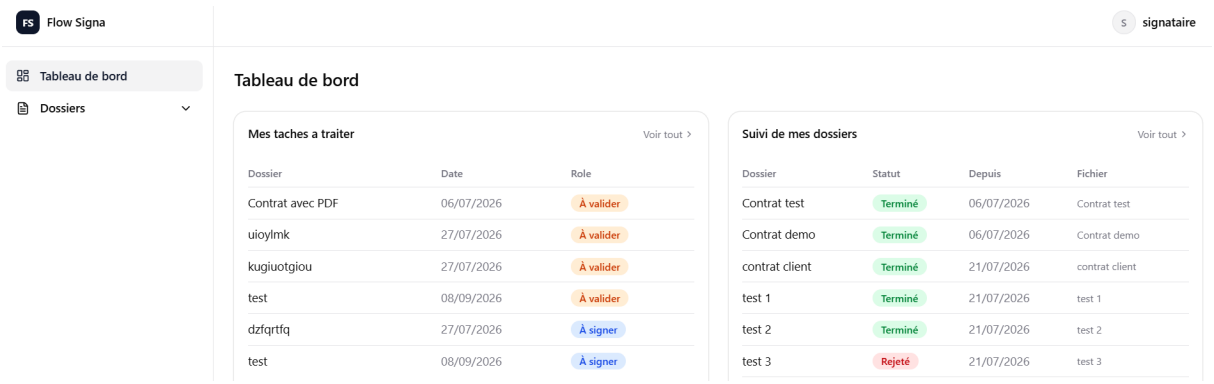

FIGURE 4.3 – Tableau de bord utilisateur

La page de détail du dossier est le cœur de l'expérience : visionneuse PDF intégrée (`PdfViewer`), liste des pièces jointes, bandeau de statut, et actions conditionnées au rôle et au statut. La signature passe obligatoirement par une modale de consentement (`SignatureConfirmModal`).

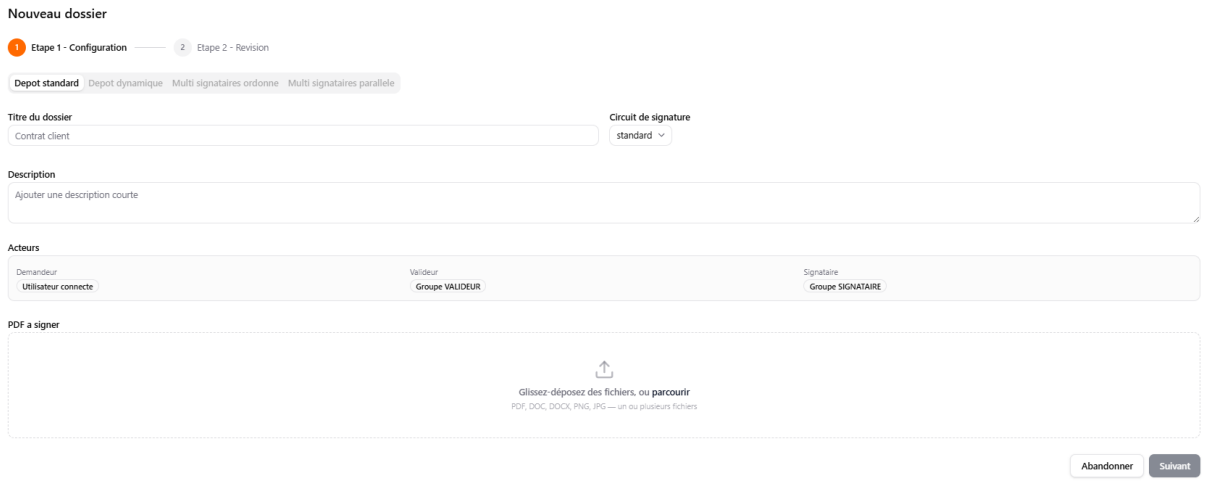

FIGURE 4.4 – Création d'un dossier avec dépôt de PDF

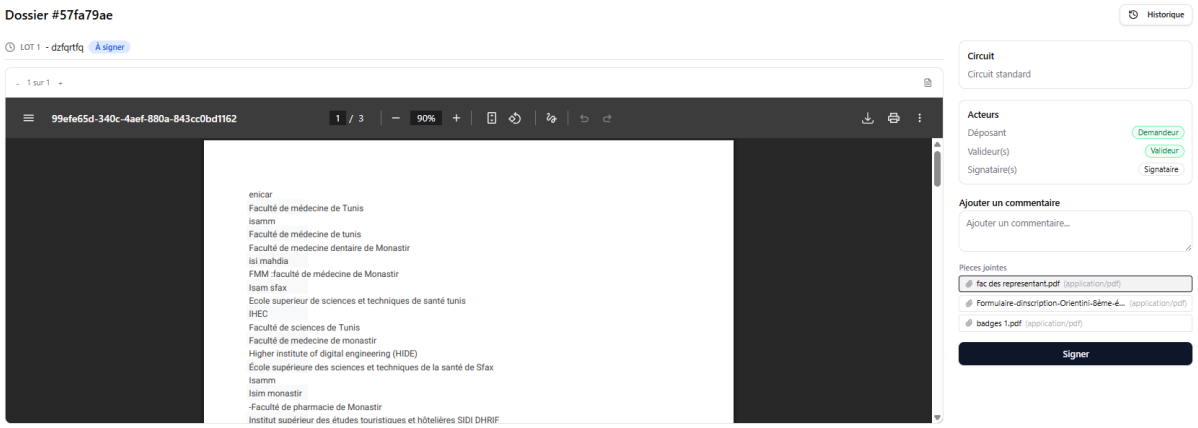

FIGURE 4.5 – Détail d'un dossier, visionneuse PDF et consentement

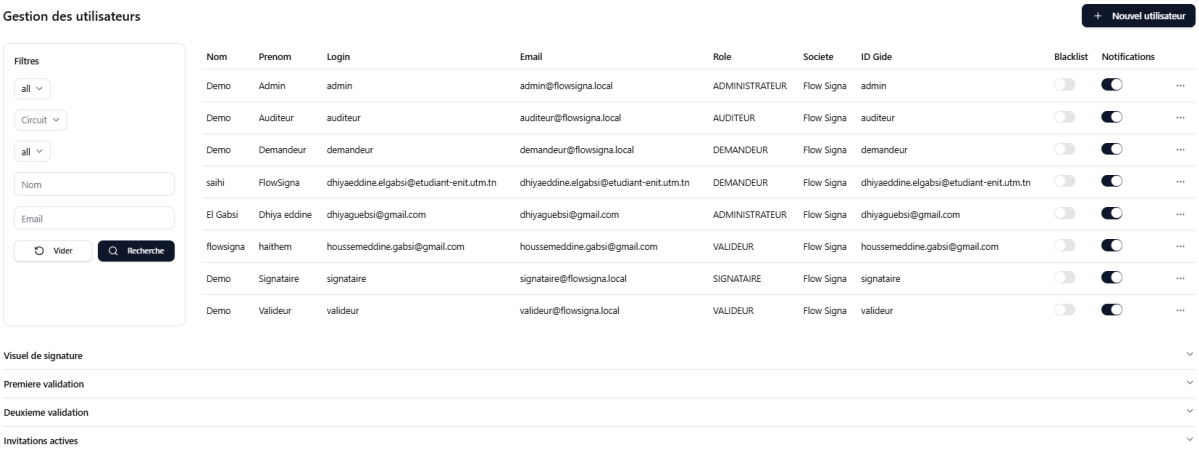

FIGURE 4.6 – Administration des circuits et des utilisateurs

4.5 Tentatives d'amélioration et d'optimisation

Plusieurs chantiers d'amélioration ont été menés ou étudiés en cours de projet :

1. **Circuits configurables.** Le modèle Circuit/CircuitAssignment a été introduit pour rendre configurables les habilitations et les destinataires des emails. Limite documentée : la topologie BPMN reste unique (pas de multi-validation ni d'escalade dans le MVP).
2. **Notifications fiables.** Les destinataires sont résolus selon le circuit avec repli, et l'action métier n'est jamais interrompue par un échec de notification.

3. **Sécurisation des échanges SPA/backend.** Le flux de rafraîchissement du jeton a été rendu robuste : déduplication des appels de refresh concurrents et rejeu automatique de la requête originelle.
4. **Optimisations identifiées mais non réalisées** (dette technique) : DTOs de réponse, contrôle magic-number des uploads, secret JWT obligatoire hors dev, Flyway, tests frontend (Vitest).

4.6 Stratégie et environnement de test

La stratégie retenue privilégie les tests d'intégration de bout en bout : les tests démarrent le contexte Spring complet sur une base H2 en mémoire et vérifient les parcours métier réels. Les tests automatisés backend (JUnit 5) couvrent quatre classes :

- `DossierWorkflowTest` : le chemin nominal et le chemin de refus, et le déploiement du BPMN ;
- `DossierAuthorizationTest` : les autorisations métier sur les dossiers ;
- `AuthSecurityTest` : authentification HTTP, jetons, règles d'accès par endpoint ;
- `AdminCircuitTest` : API d'administration des circuits.

Ces tests sont exécutés par la CI GitHub Actions (workflow backend-ci), en complément du scan de secrets (gitleaks).

Commande d'exécution :

```
.\mvnw test -pl flowsigna-api/flowsigna-app -am
```

```
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 6.234 s -- in org.flowsigna.app.DossierWorkflowTe
st
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 36, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for Flow Signa 0.0.1-SNAPSHOT:
[INFO]
[INFO] Flow Signa ......................................... SUCCESS [  0.028 s]
[INFO] flowsigna-core ..................................... SUCCESS [  0.004 s]
[INFO] flowsigna-model .................................... SUCCESS [  1.673 s]
[INFO] flowsigna-api ...................................... SUCCESS [  0.000 s]
[INFO] flowsigna-security ................................. SUCCESS [  0.314 s]
[INFO] flowsigna-app ...................................... SUCCESS [ 29.937 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

FIGURE 4.7 – Exécution des tests backend (mvnw test)

4.7 Évaluation du flux principal (parcours nominal)

Cette section évalue le flux principal de bout en bout : authentification, création et upload, soumission, validation, signature, notifications et audit.

4.7.1 Authentification et contrôle d'accès

Scénario : appel de `POST /api/auth/login` avec les identifiants du compte « demandeur » ; puis appel d'un endpoint protégé avec et sans jeton ; tentative d'une action non autorisée.

Résultats observés :

- login correct : 200, access token dans le corps, cookie `flowsigna_refresh` posé (httpOnly) ; login erroné : 401 ;
- appel protégé sans jeton : 401 ; avec jeton valide : 200 ;
- validation d'un dossier par un DEMANDEUR : 403 « Accès interdit » (couche HTTP : rôle insuffisant).

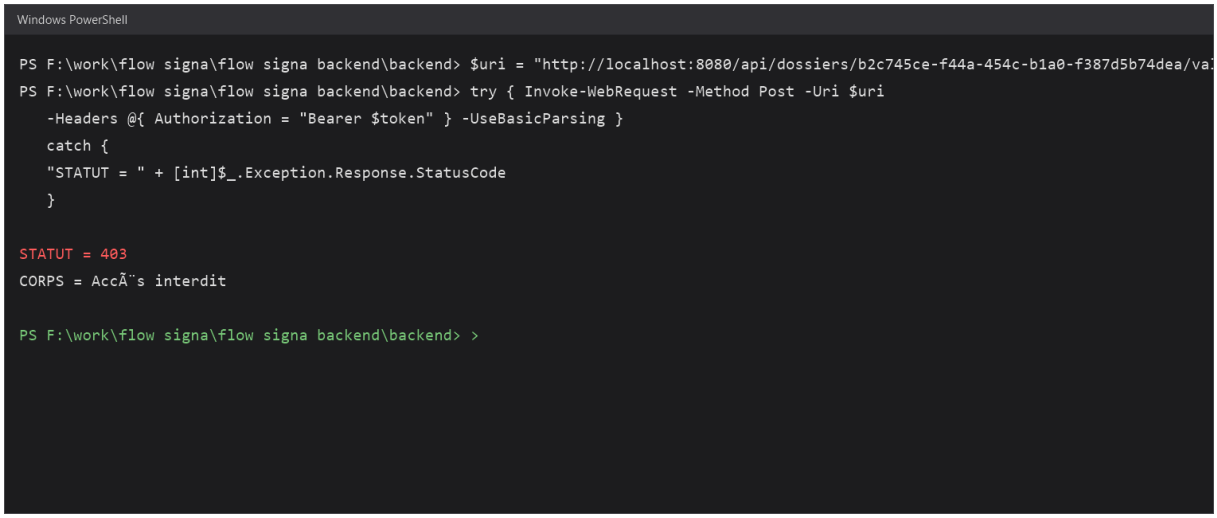

FIGURE 4.8 – Contrôle d'accès – refus 403 d'une action non autorisée

4.7.2 Création d'un dossier et upload du PDF

Résultats observés : création 201 (statut BROUILLON, entrée d'audit « depose ») ; upload PDF 201 (fichier sous nom UUID dans `uploads/`) ; upload non-PDF 400 ; upload supérieur à 10 Mo rejeté ; listing et téléchargement du PDF 200.

4.7.3 Soumission et validation

Résultats observés : soumission validée 200 (statut SOUMIS puis EN_VALIDATION, démarrage de l'instance Flowable, email aux valideurs) ; re-soumission 409 ; soumission d'un dossier vierge 409 ; validation 200 (taskValidation complétée avec `approuve=true`, statut EN_SIGNATURE).

4.7.4 Signature avec consentement et capture de preuve

Résultats observés : signature sans `consentAccepted` 400 ; avec consentement 200 (preuve capturée : signataire, IP, agent utilisateur, instant ; statut SIGNE) ; le processus atteint `endSigned` ; double signature impossible 409.

4.7.5 Notifications

Résultats observés : chaque action du workflow (soumission, validation, refus, signature) déclenche les notifications prévues ; les destinataires sont correctement résolus selon le circuit et les habilitations, et un utilisateur blacklisté est exclu des notifications.

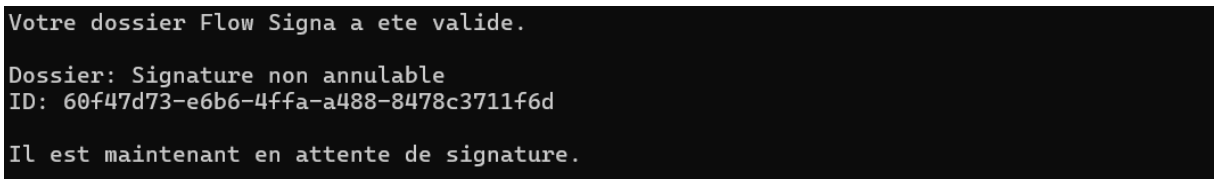

FIGURE 4.9 – Notifications déclenchées lors du workflow

4.7.6 Audit et historique

Résultats observés : `GET /api/dossiers/{id}/historique` 200 (timeline complète) ; `GET /api/audit/logs` par l'AUDITEUR 200 ; par un VALIDEUR 403 ; lecture d'un dossier par un utilisateur non concerné 403 ; aucune interface ne permet de modifier les entrées d'audit.

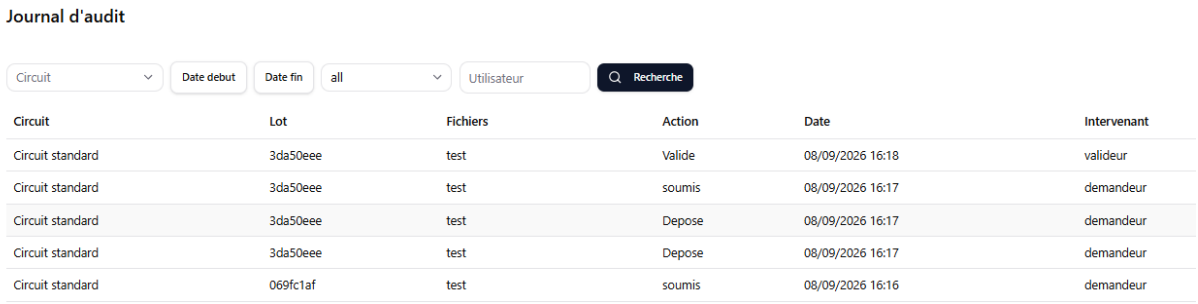

FIGURE 4.10 – Historique / timeline d'audit d'un dossier

4.8 Évaluation du flux secondaire

4.8.1 Refus d'un dossier avec motif

Résultats observés : refus sans motif 400 (motif obligatoire) ; avec motif 200 (taskValidation complétée avec `approuve=false`, statut REFUSE, notifications au demandeur, processus terminé `endRefuse`).

4.8.2 Annulation d'un dossier

Résultats observés : annulation d'un brouillon 200 (statut ANNULE) ; annulation en validation 200 (instance du processus supprimée) ; annulation d'un dossier SIGNE 409 ; annulation par un autre utilisateur que le demandeur 403.

4.8.3 Administration et autorisations

Résultats observés : création de circuits et affectations 201/200 (circuit par défaut garanti) ; seul un valideur affecté au circuit peut valider ; approbation d'une inscription : compte DEMANDEUR créé ; suivi des workflows et export CSV ; tentatives non administrateur sur `/api/admin/**` 403.

FIGURE 4.11 – Suivi des workflows et export CSV côté administration

4.9 Conclusion

Au terme de l'évaluation, le flux nominal et le flux de refus sont vérifiés de bout en bout, à la fois par les tests automatisés et par les parcours manuels ; les règles d'accès sont vérifiées aux deux couches (401/403 conformes) ; l'administration fonctionne et est réservée à l'ADMINISTRATEUR ; la résilience des notifications est vérifiée. Les limites documentées : absence de tests frontend automatisés, pas de suite E2E, pas de test de charge, et périmètre non implémenté du MVP (purge, multi-tenant, signature cryptographique).

Conclusion Générale

Ce rapport a présenté le projet Flow Signa, un parapheur électronique SaaS destiné à dématérialiser les circuits de validation et de signature de documents au sein des entreprises privées, notamment bancaires. Partis d'une problématique claire – automatiser, sécuriser et tracer des circuits jusqu'ici papier ou informels – nous avons construit, de bout en bout, le MVP complet de la plateforme.

Bilan des travaux réalisés

Le bilan est tangible : la plateforme est opérationnelle de bout en bout. Côté backend, un monolithe modulaire Spring Boot 3 (Java 17) embarque le moteur de workflow Flowable et expose une API REST documentée d'une quarantaine d'endpoints, sécurisée par des jetons JWT et un modèle d'autorisation à deux couches. Le cœur du produit – le processus BPMN « dossier-signature » et ses douze délégués – orchestre le cycle complet : soumission, validation ou refus motivé, signature avec consentement, capture de preuve, notifications et audit. Côté frontend, une SPA React 19 / TypeScript / Vite offre à chaque acteur ses écrans. L'ensemble est vérifié par une suite de tests d'intégration exécutée en intégration continue.

Au-delà de la fonctionnalité, le stage a permis de mettre en pratique une démarche d'ingénierie complète : analyse de l'existant, spécification des besoins, choix d'architecture argumentés, développement avec discipline Git/CI, tests d'intégration et documentation technique de référence.

Limites

Les limites de la version actuelle sont connues et documentées :

- la signature du MVP est une preuve applicative, sans valeur cryptographique : les niveaux avancé et qualifié eIDAS restent à intégrer ;
- la multi-tenance n'est pas implantée ;
- la purge automatique des documents n'est pas développée ;
- l'authentification repose sur un JWT maison : la migration vers Keycloak/OIDC est recommandée ;
- la topologie du workflow est unique (pas de multi-validation ni d'escalade) ;
- l'industrialisation reste à faire : Docker, HTTPS, rate limiting, Flyway, tests frontend.

Perspectives

La feuille de route de reprise s'articule en trois horizons. **Court terme – consolidation :** adoption de Flyway pour versionner le schéma, durcissement de la sécurité, nettoyage des dépôts, ajout de tests frontend. **Moyen terme – valeur métier :** purge automatique des documents, migration vers Keycloak, multi-tenance, première signature cryptographique (PAdES) et stockage objet S3/MinIO. **Long terme – conformité et industrialisation :** signature avancée puis qualifiée via un prestataire de confiance, HSM, horodatage TSA, circuits BPMN dynamiques et déploiement Docker.

En définitive, Flow Signa démontre qu'un assemblage maîtrisé de technologies éprouvées – BPMN 2.0, Flowable, Spring Boot, React, PostgreSQL – permet de répondre efficacement à un besoin métier précis et sensible. Ce projet constitue pour moi une expérience formatrice complète : de la problématique métier au produit livré et testé, en passant par les arbitrages d'architecture et la rigueur d'une démarche d'ingénierie logicielle documentée.

Bibliographie

[1] Parlement européen et Conseil de l'UE, "Règlement (UE) n° 910/2014 sur l'identification électronique et les services de confiance pour les transactions électroniques au sein du marché intérieur (eIDAS)", *Journal officiel de l'Union européenne*, 2014.

[2] OMG (Object Management Group), "Business Process Model and Notation (BPMN) – Version 2.0", spécification formelle, 2011.

[3] IETF, M. Jones, J. Bradley, N. Sakimura, "RFC 7519 : JSON Web Token (JWT)", *Request for Comments*, 2015.

[4] Cockburn, Alistair, "Hexagonal Architecture (Ports and Adapters)", article de référence, 2005.

[5] Fowler, Martin, "Microservices", article, `https://martinfowler.com`, 2014.

[6] Newman, Sam, *Building Microservices*, O'Reilly Media, 2015.

[7] Richards, Mark, Ford, Neal, *Fundamentals of Software Architecture*, O'Reilly Media, 2020.

[8] VMware/Tanzu, "Spring Boot Reference Documentation" (version 3.4), `https://spring.io/projects/spring-boot`, 2025.

[9] VMware/Tanzu, "Spring Security Reference Documentation", `https://spring.io/projects/spring-security`, 2025.

[10] Flowable, "Flowable Open Source Documentation" (version 7.1), `https://flowable.com/open-source/docs`, 2025.

[11] PostgreSQL Global Development Group, "PostgreSQL 17 Documentation", `https://www.postgresql.org`, 2024.

[12] Keycloak, "Keycloak Documentation", `https://www.keycloak.org`, 2025.

[13] Meta / communauté React, "React – The Library for Web and Native User Interfaces", `https://react.dev`, 2025.

[14] TanStack, “TanStack Query Documentation” (v5), `https://tanstack.com/query`, 2025.

[15] Vite, “Vite – Next Generation Frontend Tooling”, `https://vite.dev`, 2025.

[16] TypeScript, “TypeScript Documentation”, `https://www.typescriptlang.org`, 2025.

[17] DocuSign, “Electronic Signature Agreement Cloud”, `https://www.docusign.com`, consulté en 2026.

[18] Adobe, “Adobe Acrobat Sign”, `https://www.adobe.com/sign`, consulté en 2026.

[19] Yousign, “Solution européenne de signature électronique”, `https://www.yousign.com`, consulté en 2026.

[20] Universign, “Signature électronique et horodatage qualifiés”, `https://www.universign.com`, consulté en 2026.

[21] Spécification fonctionnelle d’origine du projet Flow Signa (description.txt, 573 lignes). Documentation interne du projet.

[22] Documentation technique de référence du projet Flow Signa : architecture, API, workflow BPMN, base de données, sécurité, tests. Documentation interne du dépôt.

[23] Code source du projet Flow Signa : dépôts “flow signa backend” et “flow signa front” (branche dev). Source première citée tout au long du chapitre 4.