**Proposition d’architecture CRM pour la gestion complète d’un établissement scolaire**

**Les « espaces » (modules fonctionnels) partagent une base de données unique ; ils échangent aussi des événements (API REST + bus de messages) afin d’être réellement dynamiques et inter‐connectés.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Domaine** | **Objectifs clés** | **Principales fonctionnalités** | **Exemples d’interactions inter-modules** |
| **1. Espace Pédagogie** | Conception et pilotage de l’offre de formation | - Programmes, syllabus, référentiels  - Planning des cours & salles  - Banque de ressources pédagogiques  - Évaluations & grilles de notation | • Un formateur publie un devoir → notification instantanée dans l’Espace Étudiants et rappel dans l’Espace Parents  • Mise à jour d’un syllabus → synchronisation automatique avec le catalogue public des formations |
| **2. Espace Étudiants** | Parcours et vie scolaire | - Portail personnel (emploi du temps, notes, absences)  - Inscription aux cours/TD, choix d’options  - Portail e-learning, remise de devoirs  - Messagerie interne & agendas | • L’étudiant valide un stage → le tuteur entreprise reçoit la fiche dans l’Espace Suivi en Entreprise |
| **3. Espace Administration** | Pilotage opérationnel | - Admissions, dossiers administratifs  - Gestion RH (contrats, fiches de paie)  - Budgets, achats, inventaire  - Conformité & accréditations | • Création d’une nouvelle cohorte → génération automatique des comptes dans les Espaces Étudiants et Enseignants |
| **4. Espace Suivi en Entreprise** | Alternance, stages, projets | - Carnet de bord missions & compétences  - Convention/contrat numérique  - Évaluation tuteur‐école-étudiant  - Facturation de l’alternance | • Fin de période d’essai signalée par l’entreprise → alerte au service Pédagogie et Administration |
| **5. Espace Enseignants & Personnel** | Gestion des intervenants | - Planning de service, feuilles d’heures  - Dépôt de supports de cours  - Évaluations QDD (qualité)  - Formation continue | • Saisie des notes → visible en temps réel dans Étudiants & Parents + déclenchement des bourses dans Finance |
| **6. Espace Parents/Tuteurs** | Suivi et communication | - Accès aux résultats, absences, factures  - Rendez-vous parents/professeurs  - Autorisations & signatures numériques | • Paiement d’une facture scolarité → reçu automatique dans Finance + mise à jour du dossier Étudiant |
| **7. Espace Finance & Facturation** | Flux financiers | - Frais de scolarité, bourses, subventions  - Comptabilité analytique par programme  - Intégration bancaire & SEPA | • Attribution d’une bourse académique → révision du solde à payer côté Parents |
| **8. Espace Admissions & Recrutement** | Pipeline des candidats | - Formulaires, pièces jointes, scoring  - Sessions d’examen en ligne  - Lettres d’admission automatiques | • Candidat admis → création automatique de son compte Étudiant, déclenchement des tâches RH si alternance |
| **9. Espace Bibliothèque & Ressources** | Accès documentaire | - Catalogue, prêts, e-books  - Gestion des licences numériques  - Statistiques d’usage | • Ajout d’un nouveau manuel obligatoire dans Pédagogie → réserve automatique d’exemplaires en Bibliothèque |
| **10. Espace Alumni & Réseau** | Vie après l’école | - Annuaire, offres d’emploi  - Événements, réseautage  - Collecte de taxe d’apprentissage & dons | • Offre d’emploi publiée par un alumnus → mise en avant dans Étudiants & Enseignants (projet tutoré) |
| **11. Espace Support & Helpdesk** | Assistance IT & bâtiment | - Tickets, SLA, maintenance  - Inventaire matériel, salles  - Notifications push | • Incident réseau signalé → impact direct sur cours hybrides (Pédagogie) : bascule en asynchrone |
| **12. Espace Reporting & BI** | Décisionnel | - Tableaux de bord KPI  - Exports CSV/Power BI  - Alertes seuils | • Chute du taux de présence détectée → alerte au directeur + tuteurs parents/entreprise |

**Principes d’architecture technique**

1. **Base de données centrale** (PostgreSQL ou MySQL) avec schéma partagé : personnes, rôles, cours, pièces comptables, etc.
2. **API REST/GraphQL** + **bus d’événements** (RabbitMQ, NATS) pour déclencher des workflows temps réel.
3. **Micro-frontends** (React/Vue) isolés par espace, chargés dynamiquement via Single-SPA ; SSO OpenID Connect.
4. **Modèle d’autorisations RBAC + ABAC** : rôles (étudiant, parent, enseignant, admin, tuteur entreprise) + règles contextuelles (promo, campus).
5. **Webhooks & intégrations externes** : LMS (Moodle/Canvas), compta (Sage), signatures électroniques, plateforme de paiement.

**Exemples de flux dynamiques entre espaces**

* **Admission ➜ Étudiants ➜ Finance**  
  Admission valide un dossier → compte étudiant créé + échéancier de paiement généré automatiquement.
* **Pédagogie ➜ Étudiants ➜ Parents**  
  Note < 10/20 enregistrée → notification immédiate à l’étudiant, copie aux parents, proposition de tutorat.
* **Suivi en Entreprise ➜ Enseignants ➜ Administration**  
  Le tuteur entreprise valide les compétences du module → l’enseignant clôture la note, l’administration édite la rémunération de l’apprenti.

**Étapes suivantes**

1. **Prioriser** les espaces selon les besoins de votre établissement (MVP = Pédagogie, Étudiants, Finance, Admissions).
2. **Cartographier** les entités de données communes (Personne, Cours, Facture, Stage…).
3. **Choisir** la pile technologique (framework, hébergement, sécurité).
4. **Prototyper** un premier workflow (ex. saisie de notes ↔ notification parents) pour valider l’architecture d’intégration.
5. **Mettre en place** une gouvernance : owners par espace, cycle DevOps CI/CD, indicateurs qualité.

N’hésitez pas à me dire si vous souhaitez un **cahier des charges détaillé**, un **schéma de base de données**, ou même un **prototype d’interface** ; je peux le préparer pour vous !

**Cahier des charges détaillé**

*(version de travail – à affiner)*

**1. Contexte & objectifs**

* **Moderniser** la gestion scolaire, administrative et financière d’un établissement accueillant ≈ 1 500 étudiants.
* **Centraliser** les données afin d’éviter les ressaisies et d’obtenir une vue 360° des personnes, cours, finances et partenariats.
* **Ouvrir** la plateforme vers des outils tiers : LMS (Moodle/Canvas), comptabilité (Sage), signature électronique, passerelles bancaires SEPA.

**2. Périmètre fonctionnel (MVP)**

|  |  |  |
| --- | --- | --- |
| **Espace** | **Fonctionnalités prioritaires** | **Jalons MVP** |
| **Pédagogie** | Gestion des programmes, planning des cours, dépôts de ressources, création d’évaluations | M1 (2 mois) |
| **Étudiants** | Portail personnel, notes, absences, remise de devoirs | M1 + M2 |
| **Finance** | Facturation frais de scolarité, échéanciers, bourses, paiements SEPA | M2 (4 mois) |
| **Admissions** | Formulaires candidature, scoring, lettres d’admission | M2 |

**3. Exigences fonctionnelles détaillées**

* **Notifications temps réel** (WS + FCM) : nouvelle note < 10/20, facture disponible, ticket support résolu, etc.
* **Workflow d’alternance** : convention → signature électronique → suivi compétences → facturation OPCO.
* **Multi-campus** : chaque entité possède ses propres calendriers, mais partage la base personnes.
* **Reporting** : indicateurs clés (taux de réussite, recouvrement, satisfaction) exposés via API GraphQL et exports CSV.

**4. Exigences non fonctionnelles**

* **Performances** : < 300 ms TTFB pour 95 % des requêtes internes.
* **Scalabilité horizontale** (pods Kubernetes).
* **Sécurité** : chiffrement TLS 1.3, SSO OpenID Connect, CSP stricte, audits OWASP A +.
* **Gouvernance données** : RGPD, historique complet (table « audit\_log »), rétention 10 ans sur les pièces comptables.

**5. Architecture technique cible**

|  |  |  |
| --- | --- | --- |
| **Couches** | **Choix retenus** | **Raison / Notes** |
| **Base de données centrale** | PostgreSQL 15 (cluster patroni) | JSONB pour méta-données flexibles, vues matérialisées pour reporting |
| **Schéma partagé** | personnes, rôles, cours, pièces comptables… (cf. DDL ci-après) | Cohérence forte, moins de synchronisation |
| **API** | REST v1 (FastAPI) + GraphQL v2 (Apollo) | REST = services internes, GraphQL = apps front |
| **Bus d’événements** | RabbitMQ « school.events.\* » (+ NATS JetStream pour streaming) | Découple workflow & notifications |
| **Front** | Micro-frontends React (shadcn/ui) or Vue 3, chargés via *single-spa* root-config | Déploiement indépendant par espace |
| **SSO & AuthZ** | Keycloak 23 (OIDC) ; RBAC (5 rôles) + ABAC (campus, promo, cohorte) | Politiques fine-grained côté Keycloak et en DB |
| **Intégrations externes** | Webhooks HTTPS signés (HMAC) ; connecteurs : Moodle, Canvas, Sage X3, Stripe/GoCardless, Yousign | Mapping des événements → bus |
| **CI/CD** | GitHub Actions → Docker → Helm charts → K8s | Blue/Green & migrations liquibase |
| **Observabilité** | OpenTelemetry, Grafana, Loki, Sentry | Traces full-stack + alerting |

**6. Livrables & planning indicatif**

|  |  |  |  |
| --- | --- | --- | --- |
| **Phase** | **Durée** | **Livrables** | **Go/NoGo** |
| **Kick-off & cadrage** | 2 sem. | Cahier des charges validé | ✅ |
| **Sprint 0 (archi)** | 3 sem. | POC event-bus, SSO, CI/CD | ✅ |
| **M1** | 8 sem. | Pédagogie + Étudiants, schéma DB stabilisé | 🔄 |
| **M2** | 8 sem. | Finance + Admissions, reporting de base | 🔄 |
| **M3** | 6 sem. | Suivi Entreprise, Parents, Enseignants | 🔄 |
| **M4** | 6 sem. | Support, Bibliothèque, Alumni, BI avancé | 🔄 |
| **Recette & déploiement** | 4 sem. | Docs d’exploitation, formation utilisateurs | 🔄 |

**Schéma de base de données – extrait (PostgreSQL DDL)**

-- === PERSONNES & RÔLES ===

CREATE TABLE person (

id SERIAL PRIMARY KEY,

first\_name VARCHAR(80) NOT NULL,

last\_name VARCHAR(80) NOT NULL,

email CITEXT UNIQUE NOT NULL,

phone VARCHAR(30),

birth\_date DATE,

created\_at TIMESTAMPTZ DEFAULT NOW()

);

CREATE TABLE role (

code VARCHAR(32) PRIMARY KEY, -- student, parent, teacher, admin, company\_tutor

description TEXT

);

CREATE TABLE user\_role (

person\_id INT REFERENCES person(id) ON DELETE CASCADE,

role\_code VARCHAR(32) REFERENCES role(code),

campus\_id INT REFERENCES campus(id),

cohort\_id INT REFERENCES cohort(id),

PRIMARY KEY(person\_id, role\_code, campus\_id, cohort\_id) -- ABAC : contexte

);

-- === SCOLARITÉ ===

CREATE TABLE program (

id SERIAL PRIMARY KEY,

name VARCHAR(120),

level VARCHAR(20) -- Licence, Master...

);

CREATE TABLE course (

id SERIAL PRIMARY KEY,

program\_id INT REFERENCES program(id),

code VARCHAR(16),

title VARCHAR(160),

ects NUMERIC(4,1),

hours INT

);

CREATE TABLE cohort (

id SERIAL PRIMARY KEY,

program\_id INT REFERENCES program(id),

year\_start SMALLINT,

label VARCHAR(30) -- ex : 2025-A1

);

CREATE TABLE enrollment (

student\_id INT REFERENCES person(id),

cohort\_id INT REFERENCES cohort(id),

enrolled\_at TIMESTAMPTZ DEFAULT NOW(),

PRIMARY KEY(student\_id, cohort\_id)

);

CREATE TABLE grade (

id SERIAL PRIMARY KEY,

student\_id INT REFERENCES person(id),

course\_id INT REFERENCES course(id),

value NUMERIC(4,1) CHECK (value BETWEEN 0 AND 20),

graded\_at TIMESTAMPTZ DEFAULT NOW()

);

-- === ENTREPRISE & ALTERNANCE ===

CREATE TABLE company (

id SERIAL PRIMARY KEY,

name VARCHAR(200),

siret CHAR(14),

address TEXT

);

CREATE TABLE apprenticeship\_contract (

id SERIAL PRIMARY KEY,

student\_id INT REFERENCES person(id),

company\_id INT REFERENCES company(id),

tutor\_id INT REFERENCES person(id), -- tuteur entreprise

start\_date DATE,

end\_date DATE,

status VARCHAR(20) -- draft, signed, archived

);

-- === FINANCE ===

CREATE TABLE invoice (

id SERIAL PRIMARY KEY,

person\_id INT REFERENCES person(id),

amount NUMERIC(10,2),

due\_date DATE,

status VARCHAR(20) DEFAULT 'pending' -- pending, paid, cancelled

);

CREATE TABLE payment (

id SERIAL PRIMARY KEY,

invoice\_id INT REFERENCES invoice(id),

paid\_amount NUMERIC(10,2),

paid\_at TIMESTAMPTZ,

method VARCHAR(30) -- sepa, card, cash

);

-- === GESTION DES ÉVÉNEMENTS ===

CREATE TABLE audit\_log (

id BIGSERIAL PRIMARY KEY,

event\_type VARCHAR(80),

payload JSONB,

emitted\_at TIMESTAMPTZ DEFAULT NOW()

);

-- TRIGGER : après INSERT/UPDATE d'une note, publier vers RabbitMQ

CREATE OR REPLACE FUNCTION notify\_grade() RETURNS TRIGGER AS $$

BEGIN

INSERT INTO audit\_log(event\_type, payload)

VALUES ('grade.created', jsonb\_build\_object('grade\_id', NEW.id));

-- (Worker Python lit audit\_log → échange « school.events.grade »)

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_grade\_notify AFTER INSERT ON grade

FOR EACH ROW EXECUTE FUNCTION notify\_grade();

*Indices, contraintes de blocs, schéma « library », « support », etc., sont omis pour garder ce snippet lisible.*

**Modèle d’API & bus d’événements**

| **Couche** | **Exemple d’endpoint / sujet** | **Payload clé** |
| --- | --- | --- |
| **REST** | POST /v1/cohorts/{id}/grades | { value: 13.5 } |
| **GraphQL** | subscription { gradeAdded(cohortId:1) { id value student { id name } } } | SSE ou WebSocket |
| **RabbitMQ** | school.events.grade.created | { grade\_id: 42 } |
| **NATS JetStream** | stream notifications subject pattern school.\* (durable consumers) | dédupliqué, replayable |

**Patterns** : commande → événement ; sagas (ex. inscription ↔ génération facture) orchestrées par *workflow-service* (Temporal.io possible).

**Stratégie micro-frontends & SSO**

1. **Root config** (Single-SPA) chargé sous portal.school.edu.
2. Chaque espace empaqueté en **parcel** React/Vue livré sur son sous-domaine (pedago. , etu. …) et déployé indépendamment.
3. **Keycloak** publie un *access token* JWT contenant :
   * rôle(s) RBAC dans realm\_access.roles
   * claims de contexte (promo, campus) ↦ policies ABAC.
4. Les parcels ne stockent pas l’état : app-shell interroge GraphQL pour le menu, puis hydrate chaque micro-app.

**Webhooks & intégrations externes**

| **Système tiers** | **Direction** | **Exemple d’événement** | **Action côté CRM** |
| --- | --- | --- | --- |
| **Moodle/Canvas** | ↔︎ | gradebook.changed | Mettre à jour table grade |
| **Sage X3** | ↔︎ | invoice.paid | Marquer facture paid + déclencher reçu PDF |
| **Stripe / GoCardless** | ← | payment\_intent.succeeded | INSERT payment + bus payment.completed |
| **Yousign** | ↔︎ | procedure.finished | Changer apprenticeship\_contract.status = 'signed' |