**Fonctionnalités Principales**

* **Gestion des Étudiants et Enseignants** : Centralisation des informations relatives aux étudiants et aux enseignants, facilitant l'accès et la mise à jour des données.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Planification des Cours** : Outils de planification intuitifs permettant de créer et de gérer les emplois du temps des cours, avec des options de récurrence et d'attribution des salles.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Suivi des Présences** : Enregistrement et suivi des présences des étudiants, avec possibilité de générer des rapports détaillés.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Portails Dédiés** : Accès personnalisé pour les enseignants, les étudiants et les parents, améliorant la communication et la collaboration au sein de la communauté scolaire.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Facturation et Paiements** : Automatisation de la facturation avec des tarifs personnalisés, prise en charge des paiements en ligne via PayPal ou carte de crédit, et possibilité de regrouper les factures pour les familles.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Communication Intégrée** : Envoi de notifications par e-mail et SMS, facilitant la diffusion d'informations importantes aux étudiants et aux parents.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Gestion du Matériel Pédagogique** : Stockage et partage de documents pédagogiques, permettant aux enseignants de fournir des ressources supplémentaires aux étudiants.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Rapports Personnalisés** : Génération de rapports détaillés sur les performances des étudiants, les présences et d'autres métriques clés, aidant à une meilleure prise de décision.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

**Technologies Utilisées**

Teach 'n Go est une application SaaS (Software as a Service) basée sur le cloud, accessible depuis n'importe quel appareil connecté à Internet. Elle utilise des technologies web modernes pour assurer une expérience utilisateur fluide et sécurisée.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

**Sécurité et Protection des Données**

La sécurité des données est une priorité pour Teach 'n Go. Les informations sont stockées dans des centres de données conformes aux normes internationales, avec des mesures de sécurité avancées telles que le cryptage SSL étendu, garantissant la protection des données sensibles.

[Teach 'n Go](https://www.teachngo.com/pages/faqs?utm_source=chatgpt.com)

**Accessibilité et Support**

L'application est disponible en plusieurs langues, dont le français, l'anglais, l'allemand, l'arabe et le chinois, rendant son utilisation accessible à une audience internationale. De plus, un support client est disponible pour assister les utilisateurs en cas de besoin

Partager

RS

##### Vous avez dit :

Donne moi cahier de charge pour application i-school

###### ChatGPT a dit :

ChatGPT

### Cahier des charges pour l'application ****i-School****

#### **1. Présentation du Projet**

L'application **i-School** est une solution SaaS (Software as a Service) destinée aux établissements scolaires pour digitaliser et automatiser les processus administratifs, académiques, et de gestion. Elle vise à offrir une interface intuitive et efficace pour les administrateurs, professeurs, élèves, et parents.

#### **2. Objectifs**

1. Digitaliser la gestion des établissements scolaires.
2. Centraliser les données académiques et administratives.
3. Améliorer la communication entre les différentes parties prenantes (parents, élèves, professeurs, administrateurs).
4. Automatiser les tâches courantes (gestion des emplois du temps, notes, paiements).
5. Offrir des rapports et statistiques en temps réel.

#### **3. Fonctionnalités Principales**

##### **3.1. Modules Utilisateurs**

* **Administrateurs :**
  + Gestion des utilisateurs (professeurs, élèves, parents).
  + Configuration des emplois du temps.
  + Gestion des paiements et factures.
  + Suivi global de l'établissement (statistiques, rapports).
* **Professeurs :**
  + Gestion des cours et des devoirs.
  + Attribution et suivi des notes.
  + Gestion des présences.
  + Communication avec les élèves et parents.
* **Étudiants :**
  + Consultation des emplois du temps.
  + Accès aux cours, notes, et devoirs.
  + Suivi des résultats académiques.
  + Participation à des évaluations en ligne.
* **Parents :**
  + Suivi des résultats et progrès des enfants.
  + Consultation des paiements et factures.
  + Notifications et rappels d’événements ou échéances.
  + Communication avec les professeurs.

##### **3.2. Fonctionnalités Techniques**

* **Authentification et Sécurité :**
  + Système d'authentification multi-rôles (Admin, Professeur, Étudiant, Parent).
  + Gestion des sessions avec JWT (JSON Web Tokens).
  + Gestion des autorisations basées sur les rôles.
* **Dashboard Dynamique :**
  + Tableau de bord personnalisé selon le rôle de l'utilisateur.
  + Accès rapide aux fonctionnalités clés.
* **Gestion Académique :**
  + Planification et gestion des cours.
  + Attribution automatique des devoirs.
  + Historique des performances des élèves.
* **Communication :**
  + Notifications en temps réel (email, SMS, push).
  + Messagerie interne entre utilisateurs.
* **Paiements et Facturation :**
  + Suivi des paiements (effectués/en attente).
  + Génération automatique de factures.
  + Historique des transactions.
* **Rapports et Analyses :**
  + Graphiques synthétiques et rapports PDF.
  + Indicateurs clés de performance (KPI) pour les administrateurs.

#### **4. Technologies Préférées**

* **Backend :** Spring Boot (Java).
* **Frontend :** Thymeleaf ou Vue.js pour une meilleure interactivité.
* **Base de Données :** H2 (développement) puis migration vers MySQL en production.
* **Authentification :** Spring Security (avec AntPathRequestMatcher).
* **Stockage des Fichiers :** S3 ou système local selon l’environnement.
* **Déploiement :** Docker et Kubernetes pour une scalabilité optimale.

#### **5. Design et Ergonomie**

* **Interface Utilisateur :**
  + Minimaliste et intuitive.
  + Design responsive adapté aux appareils mobiles et tablettes.
* **Couleurs et Branding :**
  + Palette moderne avec une priorité au blanc et au vert (couleurs éducation).
  + Logo : /images/logo/Digital-School.png.

#### **6. Contraintes et Spécifications**

1. **Sécurité :**
   * Données cryptées (chiffrement AES pour les données sensibles).
   * Système de journalisation pour toutes les actions critiques.
2. **Performance :**
   * Chargement rapide des pages (<3 secondes).
   * Gestion efficace des ressources pour supporter au moins 1000 utilisateurs simultanés.
3. **Accessibilité :**
   * Application conforme aux normes d'accessibilité (WCAG 2.1).
   * Options multilingues avec priorité au français et à l’anglais.
4. **Intégrations :**
   * API REST pour intégration avec des outils tiers (Google Calendar, systèmes de paiement, etc.).
   * Support des plugins pour des fonctionnalités additionnelles.

#### **7. Planning Prévisionnel**

1. **Phase 1 : Conception (1 mois)**
   * Analyse des besoins.
   * Modélisation des données.
   * Prototypage des interfaces.
2. **Phase 2 : Développement (3 mois)**
   * Développement backend et frontend.
   * Intégration des API et services tiers.
   * Tests unitaires.
3. **Phase 3 : Tests et Validation (1 mois)**
   * Tests fonctionnels.
   * Corrections des bugs.
   * Validation par les parties prenantes.
4. **Phase 4 : Déploiement et Formation (1 mois)**
   * Déploiement sur environnement de production.
   * Formation des utilisateurs clés.

#### **8. Budget Estimé**

* **Développement :** XXX €
* **Hébergement et Infrastructure :** XXX €/an
* **Maintenance et Support :** XXX €/mois

#### **9. Livrables**

* Application web fonctionnelle.
* Documentation technique.
* Formation des utilisateurs finaux.

Si vous souhaitez personnaliser ou détailler un point, je peux affiner cette version.

4o

##### Vous avez dit :

Donne moi le cahier de charge de cette application https://www.teachngo.com/?utm\_medium=ppc&utm\_campaign=teachngo&utm\_term=school\_management&utm\_source=GetApp&gdmcid=856bea58-87da-45ed-8d8c-df7068351c6d

###### ChatGPT a dit :

ChatGPT

Teach 'n Go est une solution logicielle de gestion scolaire moderne et conviviale, conçue pour aider les établissements éducatifs à gérer efficacement leurs opérations quotidiennes. Voici un aperçu des fonctionnalités clés que cette application propose :

### Fonctionnalités Principales

* **Gestion des Étudiants et Enseignants** : Centralisation des informations relatives aux étudiants et aux enseignants, facilitant l'accès et la mise à jour des données.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Planification des Cours** : Outils de planification intuitifs permettant de créer et de gérer les emplois du temps des cours, avec des options de récurrence et d'attribution des salles.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Suivi des Présences** : Enregistrement et suivi des présences des étudiants, avec possibilité de générer des rapports détaillés.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Portails Dédiés** : Accès personnalisé pour les enseignants, les étudiants et les parents, améliorant la communication et la collaboration au sein de la communauté scolaire.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

* **Facturation et Paiements** : Automatisation de la facturation avec des tarifs personnalisés, prise en charge des paiements en ligne via PayPal ou carte de crédit, et possibilité de regrouper les factures pour les familles.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Communication Intégrée** : Envoi de notifications par e-mail et SMS, facilitant la diffusion d'informations importantes aux étudiants et aux parents.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Gestion du Matériel Pédagogique** : Stockage et partage de documents pédagogiques, permettant aux enseignants de fournir des ressources supplémentaires aux étudiants.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

* **Rapports Personnalisés** : Génération de rapports détaillés sur les performances des étudiants, les présences et d'autres métriques clés, aidant à une meilleure prise de décision.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

### Technologies Utilisées

Teach 'n Go est une application SaaS (Software as a Service) basée sur le cloud, accessible depuis n'importe quel appareil connecté à Internet. Elle utilise des technologies web modernes pour assurer une expérience utilisateur fluide et sécurisée.

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

### Sécurité et Protection des Données

La sécurité des données est une priorité pour Teach 'n Go. Les informations sont stockées dans des centres de données conformes aux normes internationales, avec des mesures de sécurité avancées telles que le cryptage SSL étendu, garantissant la protection des données sensibles.

[Teach 'n Go](https://www.teachngo.com/pages/faqs?utm_source=chatgpt.com)

### Accessibilité et Support

L'application est disponible en plusieurs langues, dont le français, l'anglais, l'allemand, l'arabe et le chinois, rendant son utilisation accessible à une audience internationale. De plus, un support client est disponible pour assister les utilisateurs en cas de besoin.

[GetApp](https://www.getapp.fr/software/117483/teach-n-go?utm_source=chatgpt.com)

Pour plus d'informations ou pour tester l'application, vous pouvez visiter le site officiel de Teach 'n Go :

[Teach 'n Go](https://www.teachngo.com/?utm_source=chatgpt.com)

Sources

Favicon

Favicon

4o

##### Vous avez dit :

comment son organisée les matières dans l'administration

###### ChatGPT a dit :

ChatGPT

Educlasse est une plateforme de gestion scolaire qui permet une organisation efficace des matières et des niveaux d'enseignement. Bien que les détails spécifiques de l'association entre les matières et les niveaux ne soient pas explicitement décrits sur le site officiel, il est courant dans les systèmes de gestion scolaire d'adopter une structure où les matières sont définies de manière indépendante et ensuite associées aux niveaux correspondants.

**Création des Matières**

* **Objectif** : Établir une liste exhaustive des matières enseignées au sein de l'établissement.
* **Actions** :
  + Accéder à la section dédiée aux matières dans Educlasse.
  + Ajouter chaque matière avec son intitulé complet (par exemple, Mathématiques, Français, Histoire-Géographie).
  + Attribuer un code ou une abréviation unique pour faciliter les références ultérieures.

**2. Définition des Niveaux**

* **Objectif** : Identifier et configurer les différents niveaux d'enseignement présents dans l'établissement.
* **Actions** :
  + Dans Educlasse, accéder à la section des niveaux ou des classes.
  + Créer chaque niveau en précisant son nom (par exemple, CP, CE1, CE2).
  + Associer, si nécessaire, des informations supplémentaires telles que le cycle éducatif correspondant.

**3. Association Matières-Niveaux**

* **Objectif** : Déterminer quelles matières sont enseignées à chaque niveau.
* **Actions** :
  + Pour chaque niveau, lister les matières correspondantes.
  + Utiliser les fonctionnalités d'Educlasse pour associer les matières aux niveaux appropriés.
  + Vérifier que toutes les matières obligatoires et optionnelles sont correctement assignées à chaque niveau.

**4. Attribution des Professeurs aux Matières et Niveaux**

* **Objectif** : Assigner chaque professeur aux matières et niveaux qu'il enseignera.
* **Actions** :
  + Accéder à la section des enseignants dans Educlasse.
  + Pour chaque professeur, spécifier les matières et les niveaux d'enseignement.
  + S'assurer que les attributions respectent les qualifications et les préférences des enseignants.
  + Mettre à jour régulièrement ces informations en cas de changement d'affectation.

**5. Gestion des Classes et des Groupes**

* **Objectif** : Organiser les élèves en classes et, si nécessaire, en groupes spécifiques pour certaines matières.
* **Actions** :
  + Créer les classes pour chaque niveau en précisant les matières enseignées et les professeurs responsables.
  + Utiliser les fonctionnalités d'Educlasse pour gérer les groupes d'élèves, notamment pour des activités spécifiques ou des besoins particuliers.
  + Assurer une communication efficace entre les professeurs partageant une même classe ou un même groupe.

**6. Suivi et Mise à Jour**

* **Objectif** : Maintenir une organisation à jour et cohérente des associations matières-niveaux-professeurs.
* **Actions** :
  + Effectuer des vérifications régulières pour s'assurer de la pertinence des associations.
  + Mettre à jour les informations en cas de modification du programme, de l'arrivée ou du départ d'un professeur, ou de tout autre changement pertinent.
  + Utiliser les rapports et les outils d'analyse d'Educlasse pour identifier et résoudre les éventuels problèmes organisationnels.

Les principales fonctionnalités incluent :

* Gestion des documents (upload, validation, téléchargement)
* Gestion des salles (CRUD, gestion des équipements)
* Gestion des profils (étudiants, parents, associations)
* Filtres dynamiques
* Tableaux de données interactifs
* Formulaires modaux
* Notifications utilisateur

1. Gestion des documents :
   * Upload/download de documents
   * Validation des documents
   * Catégorisation et typage
   * Association avec étudiants/parents
2. Gestion des salles :
   * CRUD complet
   * Gestion des équipements
   * Gestion du statut
   * Recherche par critères
3. Gestion des profils :
   * Profils étudiants avec informations médicales
   * Profils parents avec préférences de contact
   * Associations parent-élève avec validation
   * Filtrage et recherche

Les services sont définis par des interfaces pour permettre différentes implémentations si nécessaire. Les contrôleurs suivent les bonnes pratiques REST et gèrent correctement les erreurs.

Ce tableau de bord étudiant comprend toutes les fonctionnalités demandées :

1. Accueil personnalisé :

* Message de bienvenue personnalisé
* Statistiques rapides (présence, moyenne, devoirs, examens)
* Annonces et informations importantes

1. Accès aux cours :

* Liste des ressources pédagogiques par matière
* Téléchargement des documents
* Visualisation des vidéos dans un modal

1. Gestion des devoirs :

* Liste des devoirs à rendre
* Système de soumission de devoirs
* Statut et dates limites

1. Emploi du temps :

* Calendrier des événements à venir
* Filtres par type d'événement
* Navigation temporelle

1. Gestion des examens :

* Calendrier des examens
* Notes et résultats
* Moyennes comparatives

1. Présence et assiduité :

* Graphique de suivi des présences
* Statistiques d'assiduité
* Historique des absences/retards

1. Communication :

* Messages récents
* Notifications
* Contact avec les enseignants

1. Matières et ressources :

* Liste des matières
* Ressources pédagogiques
* Progression par matière

Le design est moderne, responsive et intuitif, avec des animations fluides et une expérience utilisateur optimisée.

Pour utiliser ces données dans les vues, assurez-vous que les attributs du modèle correspondent aux variables Thymeleaf dans les templates. Par exemple, dans student-dashboard.html, vous pouvez accéder aux statistiques avec ${stats.attendanceRate}, ${stats.averageGrade}, etc.

Les graphiques dans le fichier JavaScript peuvent être mis à jour pour utiliser les données réelles en passant les données du modèle via un objet JSON dans le template :

<script th:inline="javascript">

    const dashboardData = /\*[[${stats}]]\*/ {};

    const gradesData = /\*[[${recentGrades}]]\*/ [];

    // ... initialiser les graphiques avec les données réelles

</script>

Fonctionnalités Détailées 2.1. Génération de Factures Système de génération automatique des factures à une date prédéfinie. Données incluses : Identité de l’élève et du parent (nom, prénom, classe, etc.). Frais détaillés (inscription, scolarité, activités). Montant total et échéance de paiement. Personnalisation possible pour des frais exceptionnels (ex. voyages scolaires). Factures disponibles en PDF téléchargeables. 2.2. Paiement Sécurisé par CB Passerelle de paiement :

Intégration avec des fournisseurs adaptés au Maroc (Payzone, CMI, Stripe Maroc, etc.). Conformité avec les normes PCI DSS pour la sécurité des paiements. Gestion des cartes marocaines et internationales (Visa, Mastercard, etc.). Processus de paiement :

Redirection vers une page de paiement sécurisée. Validation en temps réel et retour de confirmation. Gestion des échecs de paiement avec messages clairs. Options de paiement :

Paiement en une seule fois. Paiement échelonné (avec calcul automatique des échéances). Suivi des échéances dans l’interface parent. 2.3. Relances Automatiques Relances programmées :

Envoi automatique d’e-mails et SMS en cas de facture impayée. Fréquence configurable (ex. 3 jours avant échéance, 1 semaine après). Personnalisation du contenu des relances. Alertes via le portail :

Notifications visibles sur le tableau de bord des parents. Indicateur de retard visible pour chaque facture. 2.4. Suivi des Paiements pour l'Administration Tableau de bord centralisé :

Liste des paiements par statut (payé, en attente, en retard). Filtrage par parent, élève, classe, ou période. Historique des paiements pour chaque élève. Rapports financiers :

Chiffre d’affaires par période. Liste des impayés et montants dus. Statistiques sur les paiements (temps moyen de paiement, modes préférés, etc.). Gestion des statuts de paiement :

Possibilité pour l’administration de marquer une facture comme payée en cas de paiement manuel. Annulation de paiements ou ajustements sur demande. 2.5. Interface Parent Tableau de bord pour :

Visualiser les factures en cours, payées ou en retard. Historique complet des paiements. Téléchargement des factures et reçus en PDF. Fonctionnalités de paiement :

Paiement sécurisé en ligne. Suivi des échéances en cas de paiement échelonné. Consultation des frais détaillés.

Je vais vous aider à créer les entités, repositories, services et contrôleurs pour la gestion des paiements et factures.

1. Entités :

// src/main/java/com/digital/school/model/Invoice.java

package com.digital.school.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDateTime;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "invoices")

public class Invoice {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(unique = true)

private String invoiceNumber;

@ManyToOne

@JoinColumn(name = "student\_id")

private User student;

@ManyToOne

@JoinColumn(name = "parent\_id")

private User parent;

private LocalDateTime issueDate;

private LocalDateTime dueDate;

@Column(precision = 10, scale = 2)

private BigDecimal totalAmount;

private String status; // PENDING, PAID, OVERDUE, CANCELLED

@OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)

private Set<InvoiceItem> items = new HashSet<>();

@OneToMany(mappedBy = "invoice", cascade = CascadeType.ALL)

private Set<Payment> payments = new HashSet<>();

private String notes;

private boolean installmentEnabled;

private Integer installmentCount;

@Column(columnDefinition = "TEXT")

private String pdfPath;

private LocalDateTime lastReminderSent;

private Integer reminderCount;

// Getters and setters

}

// src/main/java/com/digital/school/model/InvoiceItem.java

package com.digital.school.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

@Entity

@Table(name = "invoice\_items")

public class InvoiceItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "invoice\_id")

private Invoice invoice;

private String description;

private String type; // TUITION, REGISTRATION, ACTIVITY, OTHER

@Column(precision = 10, scale = 2)

private BigDecimal amount;

private Integer quantity;

@Column(precision = 10, scale = 2)

private BigDecimal total;

// Getters and setters

}

// src/main/java/com/digital/school/model/Payment.java

package com.digital.school.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDateTime;

@Entity

@Table(name = "payments")

public class Payment {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "invoice\_id")

private Invoice invoice;

@Column(precision = 10, scale = 2)

private BigDecimal amount;

private String paymentMethod; // CARD, CASH, BANK\_TRANSFER

private String transactionId;

private String status; // PENDING, COMPLETED, FAILED

private LocalDateTime paymentDate;

@Column(columnDefinition = "TEXT")

private String receiptPath;

private String errorMessage;

// Getters and setters

}

// src/main/java/com/digital/school/model/PaymentReminder.java

package com.digital.school.model;

import jakarta.persistence.\*;

import java.time.LocalDateTime;

@Entity

@Table(name = "payment\_reminders")

public class PaymentReminder {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@ManyToOne

@JoinColumn(name = "invoice\_id")

private Invoice invoice;

private String type; // EMAIL, SMS

private String status; // SENT, FAILED

private LocalDateTime sentAt;

private String content;

// Getters and setters

}

1. Repositories:

// src/main/java/com/digital/school/repository/InvoiceRepository.java

package com.digital.school.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.digital.school.model.Invoice;

import com.digital.school.model.User;

import java.time.LocalDateTime;

import java.util.List;

public interface InvoiceRepository extends JpaRepository<Invoice, Long> {

List<Invoice> findByParentOrderByIssueDateDesc(User parent);

List<Invoice> findByStudentOrderByIssueDateDesc(User student);

List<Invoice> findByStatusAndDueDateBefore(String status, LocalDateTime date);

@Query("SELECT i FROM Invoice i WHERE i.status = 'PENDING' AND i.dueDate < CURRENT\_TIMESTAMP")

List<Invoice> findOverdueInvoices();

@Query("SELECT i FROM Invoice i WHERE i.lastReminderSent < ?1 OR i.lastReminderSent IS NULL")

List<Invoice> findInvoicesNeedingReminder(LocalDateTime cutoffDate);

}

// src/main/java/com/digital/school/repository/PaymentRepository.java

package com.digital.school.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Query;

import com.digital.school.model.Payment;

import java.math.BigDecimal;

import java.time.LocalDateTime;

import java.util.List;

public interface PaymentRepository extends JpaRepository<Payment, Long> {

List<Payment> findByInvoiceId(Long invoiceId);

@Query("SELECT SUM(p.amount) FROM Payment p WHERE p.status = 'COMPLETED' AND p.paymentDate BETWEEN ?1 AND ?2")

BigDecimal calculateRevenue(LocalDateTime start, LocalDateTime end);

@Query("SELECT p.paymentMethod, COUNT(p) FROM Payment p WHERE p.status = 'COMPLETED' GROUP BY p.paymentMethod")

List<Object[]> getPaymentMethodStats();

}

// src/main/java/com/digital/school/repository/PaymentReminderRepository.java

package com.digital.school.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import com.digital.school.model.PaymentReminder;

import java.util.List;

public interface PaymentReminderRepository extends JpaRepository<PaymentReminder, Long> {

List<PaymentReminder> findByInvoiceIdOrderBySentAtDesc(Long invoiceId);

}

1. Services:

// src/main/java/com/digital/school/service/BillingService.java

package com.digital.school.service;

import com.digital.school.model.\*;

import java.math.BigDecimal;

import java.util.List;

import java.util.Map;

public interface BillingService {

Invoice generateInvoice(User student, List<InvoiceItem> items);

Invoice getInvoice(Long id);

List<Invoice> getParentInvoices(User parent);

byte[] generateInvoicePdf(Invoice invoice);

void sendInvoiceReminders();

Map<String, Object> getBillingStats();

}

// src/main/java/com/digital/school/service/PaymentService.java

package com.digital.school.service;

import com.digital.school.model.\*;

import java.math.BigDecimal;

public interface PaymentService {

Payment processPayment(Invoice invoice, BigDecimal amount, String paymentMethod);

Payment getPayment(Long id);

void handlePaymentCallback(String transactionId, String status);

byte[] generateReceipt(Payment payment);

}

// src/main/java/com/digital/school/service/impl/BillingServiceImpl.java

package com.digital.school.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.digital.school.model.\*;

import com.digital.school.repository.\*;

import com.digital.school.service.BillingService;

import java.time.LocalDateTime;

import java.util.\*;

@Service

@Transactional

public class BillingServiceImpl implements BillingService {

@Autowired

private InvoiceRepository invoiceRepository;

@Autowired

private PaymentReminderRepository reminderRepository;

@Autowired

private EmailService emailService;

@Autowired

private SmsService smsService;

@Override

public Invoice generateInvoice(User student, List<InvoiceItem> items) {

Invoice invoice = new Invoice();

invoice.setStudent(student);

invoice.setParent(student.getParent());

invoice.setInvoiceNumber(generateInvoiceNumber());

invoice.setIssueDate(LocalDateTime.now());

invoice.setDueDate(LocalDateTime.now().plusDays(30));

invoice.setStatus("PENDING");

BigDecimal total = BigDecimal.ZERO;

for (InvoiceItem item : items) {

item.setInvoice(invoice);

item.setTotal(item.getAmount().multiply(new BigDecimal(item.getQuantity())));

total = total.add(item.getTotal());

}

invoice.setTotalAmount(total);

invoice.getItems().addAll(items);

invoice = invoiceRepository.save(invoice);

// Generate PDF

byte[] pdfContent = generateInvoicePdf(invoice);

String pdfPath = savePdfToStorage(pdfContent, invoice.getInvoiceNumber());

invoice.setPdfPath(pdfPath);

// Send notification

emailService.sendInvoiceNotification(invoice);

return invoice;

}

@Override

public void sendInvoiceReminders() {

LocalDateTime cutoffDate = LocalDateTime.now().minusDays(7);

List<Invoice> invoices = invoiceRepository.findInvoicesNeedingReminder(cutoffDate);

for (Invoice invoice : invoices) {

// Send email reminder

PaymentReminder emailReminder = new PaymentReminder();

emailReminder.setInvoice(invoice);

emailReminder.setType("EMAIL");

emailReminder.setSentAt(LocalDateTime.now());

try {

emailService.sendPaymentReminder(invoice);

emailReminder.setStatus("SENT");

} catch (Exception e) {

emailReminder.setStatus("FAILED");

}

reminderRepository.save(emailReminder);

// Send SMS reminder

PaymentReminder smsReminder = new PaymentReminder();

smsReminder.setInvoice(invoice);

smsReminder.setType("SMS");

smsReminder.setSentAt(LocalDateTime.now());

try {

smsService.sendPaymentReminder(invoice);

smsReminder.setStatus("SENT");

} catch (Exception e) {

smsReminder.setStatus("FAILED");

}

reminderRepository.save(smsReminder);

// Update invoice

invoice.setLastReminderSent(LocalDateTime.now());

invoice.setReminderCount(invoice.getReminderCount() + 1);

invoiceRepository.save(invoice);

}

}

// Other methods implementation...

}

// src/main/java/com/digital/school/service/impl/PaymentServiceImpl.java

package com.digital.school.service.impl;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.digital.school.model.\*;

import com.digital.school.repository.\*;

import com.digital.school.service.PaymentService;

import java.math.BigDecimal;

import java.time.LocalDateTime;

@Service

@Transactional

public class PaymentServiceImpl implements PaymentService {

@Autowired

private PaymentRepository paymentRepository;

@Autowired

private InvoiceRepository invoiceRepository;

@Autowired

private PaymentGatewayService paymentGateway;

@Override

public Payment processPayment(Invoice invoice, BigDecimal amount, String paymentMethod) {

Payment payment = new Payment();

payment.setInvoice(invoice);

payment.setAmount(amount);

payment.setPaymentMethod(paymentMethod);

payment.setPaymentDate(LocalDateTime.now());

payment.setStatus("PENDING");

payment = paymentRepository.save(payment);

// Process payment through gateway

String transactionId = paymentGateway.initiatePayment(payment);

payment.setTransactionId(transactionId);

return paymentRepository.save(payment);

}

@Override

public void handlePaymentCallback(String transactionId, String status) {

Payment payment = paymentRepository.findByTransactionId(transactionId)

.orElseThrow(() -> new RuntimeException("Payment not found"));

payment.setStatus(status);

if ("COMPLETED".equals(status)) {

payment.setPaymentDate(LocalDateTime.now());

// Generate receipt

byte[] receiptContent = generateReceipt(payment);

String receiptPath = saveReceiptToStorage(receiptContent, payment.getId());

payment.setReceiptPath(receiptPath);

// Update invoice status if fully paid

Invoice invoice = payment.getInvoice();

BigDecimal totalPaid = paymentRepository.getTotalPaidAmount(invoice.getId());

if (totalPaid.compareTo(invoice.getTotalAmount()) >= 0) {

invoice.setStatus("PAID");

invoiceRepository.save(invoice);

}

}

paymentRepository.save(payment);

}

// Other methods implementation...

}

1. Controllers:

// src/main/java/com/digital/school/controller/BillingController.java

package com.digital.school.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import com.digital.school.model.\*;

import com.digital.school.service.\*;

import java.util.List;

import java.util.Map;

@Controller

@RequestMapping("/billing")

public class BillingController {

@Autowired

private BillingService billingService;

@Autowired

private PaymentService paymentService;

@GetMapping("/invoices")

public String listInvoices(@AuthenticationPrincipal User user, Model model) {

List<Invoice> invoices = billingService.getParentInvoices(user);

model.addAttribute("invoices", invoices);

return "billing/invoices";

}

@GetMapping("/invoices/{id}")

public String viewInvoice(@PathVariable Long id, Model model) {

Invoice invoice = billingService.getInvoice(id);

model.addAttribute("invoice", invoice);

return "billing/invoice-details";

}

@GetMapping("/invoices/{id}/pdf")

public ResponseEntity<byte[]> downloadInvoice(@PathVariable Long id) {

Invoice invoice = billingService.getInvoice(id);

byte[] pdf = billingService.generateInvoicePdf(invoice);

return ResponseEntity.ok()

.header("Content-Type", "application/pdf")

.header("Content-Disposition", "attachment; filename=\"invoice-" + invoice.getInvoiceNumber() + ".pdf\"")

.body(pdf);

}

@PostMapping("/invoices/{id}/pay")

@ResponseBody

public ResponseEntity<?> payInvoice(

@PathVariable Long id,

@RequestBody Map<String, Object> paymentDetails) {

try {

Invoice invoice = billingService.getInvoice(id);

Payment payment = paymentService.processPayment(

invoice,

new BigDecimal(paymentDetails.get("amount").toString()),

paymentDetails.get("method").toString()

);

return ResponseEntity.ok(payment);

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

@PostMapping("/payment-callback")

@ResponseBody

public ResponseEntity<?> handlePaymentCallback(@RequestBody Map<String, String> callback) {

try {

paymentService.handlePaymentCallback(

callback.get("transactionId"),

callback.get("status")

);

return ResponseEntity.ok().build();

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

}

// src/main/java/com/digital/school/controller/admin/AdminBillingController.java

package com.digital.school.controller.admin;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import com.digital.school.model.\*;

import com.digital.school.service.\*;

import java.util.List;

import java.util.Map;

@Controller

@RequestMapping("/admin/billing")

public class AdminBillingController {

@Autowired

private BillingService billingService;

@Autowired

private PaymentService paymentService;

@GetMapping("/dashboard")

public String dashboard(Model model) {

model.addAttribute("stats", billingService.getBillingStats());

return "admin/billing/dashboard";

}

@GetMapping("/invoices")

public String listInvoices(Model model) {

return "admin/billing/invoices";

}

@GetMapping("/invoices/data")

@ResponseBody

public List<Map<String, Object>> getInvoicesData() {

return billingService.getAllInvoicesAsMap();

}

@PostMapping("/invoices/generate")

@ResponseBody

public ResponseEntity<?> generateInvoice(@RequestBody Map<String, Object> data) {

try {

User student = userService.findById(Long.parseLong(data.get("studentId").toString()))

.orElseThrow(() -> new RuntimeException("Student not found"));

List<InvoiceItem> items = parseInvoiceItems(data.get("items"));

Invoice invoice = billingService.generateInvoice(student, items);

return ResponseEntity.ok(invoice);

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

@PostMapping("/invoices/{id}/remind")

@ResponseBody

public ResponseEntity<?> sendReminder(@PathVariable Long id) {

try {

billingService.sendInvoiceReminder(id);

return ResponseEntity.ok().build();

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

}

1. Vues:

<!-- src/main/resources/templates/billing/invoices.html -->

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Mes Factures - Digital School</title>

<!-- Include necessary CSS -->

</head>

<body>

<div class="container">

<h1>Mes Factures</h1>

<div class="invoices-list">

<div th:each="invoice : ${invoices}" class="invoice-card">

<div class="invoice-header">

<h3 th:text="${'Facture #' + invoice.invoiceNumber}">Facture #123</h3>

<span th:class="${'status-badge status-' + invoice.status.toLowerCase()}"

th:text="${invoice.status}">Status</span>

</div>

<div class="invoice-details">

<div class="amount">

<span class="label">Montant:</span>

<span class="value" th:text="${invoice.totalAmount + ' MAD'}">1000 MAD</span>

</div>

<div class="dates">

<div>

<span class="label">Date d'émission:</span>

<span th:text="${#temporals.format(invoice.issueDate, 'dd/MM/yyyy')}">

01/01/2024

</span>

</div>

<div>

<span class="label">Échéance:</span>

<span th:text="${#temporals.format(invoice.dueDate, 'dd/MM/yyyy')}">

31/01/2024

</span>

</div>

</div>

</div>

<div class="invoice-actions">

<a th:href="@{'/billing/invoices/' + ${invoice.id}}"

class="btn btn-primary">

Voir détails

</a>

<a th:href="@{'/billing/invoices/' + ${invoice.id} + '/pdf'}"

class="btn btn-secondary">

Télécharger PDF

</a>

<button th:if="${invoice.status == 'PENDING'}"

th:onclick="'showPaymentModal(' + ${invoice.id} + ')'"

class="btn btn-success">

Payer

</button>

</div>

</div>

</div>

</div>

<!-- Payment Modal -->

<div id="paymentModal" class="modal">

<div class="modal-content">

<h2>Paiement de la facture</h2>

<form id="paymentForm">

<input type="hidden" id="invoiceId" name="invoiceId">

<div class="form-group">

<label for="amount">Montant</label>

<input type="number" id="amount" name="amount" required>

</div>

<div class="form-group">

<label for="paymentMethod">Mode de paiement</label>

<select id="paymentMethod" name="paymentMethod" required>

<option value="CARD">Carte bancaire</option>

<option value="BANK\_TRANSFER">Virement bancaire</option>

</select>

</div>

<div class="form-actions">

<button type="button" onclick="closePaymentModal()"

class="btn btn-secondary">

Annuler

</button>

<button type="submit" class="btn btn-primary">

Procéder au paiement

</button>

</div>

</form>

</div>

</div>

<script th:inline="javascript">

function showPaymentModal(invoiceId) {

document.getElementById('invoiceId').value = invoiceId;

document.getElementById('paymentModal').style.display = 'block';

}

function closePaymentModal() {

document.getElementById('paymentModal').style.display = 'none';

}

document.getElementById('paymentForm').addEventListener('submit', async function(e) {

e.preventDefault();

const invoiceId = document.getElementById('invoiceId').value;

const amount = document.getElementById('amount').value;

const method = document.getElementById('paymentMethod').value;

try {

const response = await fetch(`/billing/invoices/${invoiceId}/pay`, {

method: 'POST',

headers: {

'Content-Type': 'application/json',

'X-CSRF-TOKEN': document.querySelector('meta[name="\_csrf"]').content

},

body: JSON.stringify({ amount, method })

});

if (!response.ok) throw new Error('Erreur lors du paiement');

const payment = await response.json();

// Redirect to payment gateway

window.location.href = payment.gatewayUrl;

} catch (error) {

alert('Erreur: ' + error.message);

}

});

</script>

</body>

</html>

<!-- src/main/resources/templates/admin/billing/dashboard.html -->

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<title>Tableau de Bord Facturation - Digital School</title>

<!-- Include necessary CSS -->

</head>

<body>

<div class="dashboard-container">

<h1>Tableau de Bord Facturation</h1>

<!-- Stats Grid -->

<div class="stats-grid">

<div class="stat-card">

<div class="stat-icon revenue">

<i class="fas fa-money-bill-wave"></i>

</div>

<div class="stat-content">

<h3>Chiffre d'affaires</h3>

<div class="stat-value"

th:text="${stats.revenue + ' MAD'}">

100,000 MAD

</div>

<div class="stat-change"

th:classappend="${stats.revenueChange >= 0 ? 'positive' : 'negative'}">

<i th:class="${'fas fa-arrow-' + (stats.revenueChange >= 0 ? 'up' : 'down')}"></i>

<span th:text="${Math.abs(stats.revenueChange) + '% vs mois dernier'}">

+15% vs mois dernier

</span>

</div>

</div>

</div>

<div class="stat-card">

<div class="stat-icon pending">

<i class="fas fa-clock"></i>

</div>

<div class="stat-content">

<h3>En attente</h3>

<div class="stat-value"

th:text="${stats.pendingAmount + ' MAD'}">

25,000 MAD

</div>

<div class="stat-label"

th:text="${stats.pendingCount + ' factures'}">

10 factures

</div>

</div>

</div>

<div class="stat-card">

<div class="stat-icon overdue">

<i class="fas fa-exclamation-triangle"></i>

</div>

<div class="stat-content">

<h3>En retard</h3>

<div class="stat-value"

th:text="${stats.overdueAmount + ' MAD'}">

15,000 MAD

</div>

<div class="stat-label"

th:text="${stats.overdueCount + ' factures'}">

5 factures

</div>

</div>

</div>

<div class="stat-card">

<div class="stat-icon success">

<i class="fas fa-check-circle"></i>

</div>

<div class="stat-content">

<h3>Taux de recouvrement</h3>

<div class="stat-value"

th:text="${stats.recoveryRate + '%'}">

85%

</div>

<div class="stat-change positive">

<i class="fas fa-arrow-up"></i>

<span>+5% ce mois</span>

</div>

</div>

</div>

</div>

<!-- Charts -->

<div class="charts-grid">

<div class="chart-card">

<div class="chart-header">

<h3>Évolution des paiements</h3>

<div class="chart-actions">

<button class="btn-outline active">Mois</button>

<button class="btn-outline">Année</button>

</div>

</div>

<div class="chart-body">

<canvas id="paymentsChart"></canvas>

</div>

</div>

<div class="chart-card">

<div class="chart-header">

<h3>Répartition par mode de paiement</h3>

</div>

<div class="chart-body">

<canvas id="paymentMethodsChart"></canvas>

</div>

</div>

</div>

<!-- Recent Invoices -->

<div class="recent-section">

<div class="section-header">

<h3>Factures récentes</h3>

<a href="/admin/billing/invoices" class="btn-link">

Voir tout

</a>

</div>

<div class="table-container">

<table class="table">

<thead>

<tr>

<th>Numéro</th>

<th>Élève</th>

<th>Montant</th>

<th>Date</th>

<th>Échéance</th>

<th>Statut</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr th:each="invoice : ${recentInvoices}">

<td th:text="${invoice.invoiceNumber}">INV-001</td>

<td th:text="${invoice.student.firstName + ' ' + invoice.student.lastName}">

John Doe

</td>

<td th:text="${invoice.totalAmount + ' MAD'}">1000 MAD</td>

<td th:text="${#temporals.format(invoice.issueDate, 'dd/MM/yyyy')}">

01/01/2024

</td>

<td th:text="${#temporals.format(invoice.dueDate, 'dd/MM/yyyy')}">

31/01/2024

</td>

<td>

<span th:class="${'status-badge status-' + invoice.status.toLowerCase()}"

th:text="${invoice.status}">

Status

</span>

</td>

<td>

<div class="action-buttons">

<a th:href="@{'/admin/billing/invoices/' + ${invoice.id}}"

class="btn btn-sm btn-primary">

<i class="fas fa-eye"></i>

</a>

<button th:if="${invoice.status == 'PENDING' || invoice.status == 'OVERDUE'}"

th:onclick="'sen ```html

th:onclick="'sendReminder(' + ${invoice.id} + ')'"

class="btn btn-sm btn-warning">

<i class="fas fa-bell"></i>

</button>

<button th:if="${invoice.status == 'PENDING'}"

th:onclick="'markAsPaid(' + ${invoice.id} + ')'"

class="btn btn-sm btn-success">

<i class="fas fa-check"></i>

</button>

</div>

</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<script th:inline="javascript">

// Initialize charts with data from model

const stats = /\*[[${stats}]]\*/ {};

// Payments Evolution Chart

new Chart(document.getElementById('paymentsChart'), {

type: 'line',

data: {

labels: stats.paymentLabels,

datasets: [{

label: 'Paiements',

data: stats.paymentAmounts,

borderColor: '#4C51BF',

tension: 0.4

}]

},

options: {

responsive: true,

scales: {

y: {

beginAtZero: true,

ticks: {

callback: function(value) {

return value + ' MAD';

}

}

}

}

}

});

// Payment Methods Chart

new Chart(document.getElementById('paymentMethodsChart'), {

type: 'doughnut',

data: {

labels: stats.paymentMethodLabels,

datasets: [{

data: stats.paymentMethodAmounts,

backgroundColor: [

'#4C51BF',

'#48BB78',

'#ED8936'

]

}]

},

options: {

responsive: true,

plugins: {

legend: {

position: 'bottom'

}

}

}

});

async function sendReminder(invoiceId) {

try {

const response = await fetch(`/admin/billing/invoices/${invoiceId}/remind`, {

method: 'POST',

headers: {

'X-CSRF-TOKEN': document.querySelector('meta[name="\_csrf"]').content

}

});

if (!response.ok) throw new Error('Erreur lors de l\'envoi du rappel');

showNotification('Rappel envoyé avec succès', 'success');

} catch (error) {

showNotification(error.message, 'error');

}

}

async function markAsPaid(invoiceId) {

if (!confirm('Êtes-vous sûr de vouloir marquer cette facture comme payée ?')) return;

try {

const response = await fetch(`/admin/billing/invoices/${invoiceId}/mark-paid`, {

method: 'POST',

headers: {

'X-CSRF-TOKEN': document.querySelector('meta[name="\_csrf"]').content

}

});

if (!response.ok) throw new Error('Erreur lors de la mise à jour du statut');

location.reload();

} catch (error) {

showNotification(error.message, 'error');

}

}

</script>

</body>

</html>

Ces fichiers fournissent une base complète pour la gestion des factures et des paiements. Le système inclut :

1. Génération automatique des factures
2. Paiement en ligne sécurisé
3. Relances automatiques
4. Suivi des paiements pour l'administration
5. Interface parent pour la gestion des factures

Les vues utilisent Thymeleaf pour l'affichage dynamique des données et incluent des graphiques pour visualiser les statistiques. Le système de paiement est prêt à être intégré avec différentes passerelles de paiement.

Pour compléter l'implémentation, vous devrez :

1. Configurer une passerelle de paiement (ex: CMI, Payzone)
2. Implémenter les services EmailService et SmsService
3. Configurer le stockage des PDFs (factures et reçus)

// src/main/java/com/digital/school/config/CMIConfig.java

package com.digital.school.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.context.annotation.Configuration;

import lombok.Data;

@Configuration

@ConfigurationProperties(prefix = "cmi")

@Data

public class CMIConfig {

private String merchantId;

private String storeKey;

private String apiUrl;

private String successUrl;

private String failureUrl;

private String currency = "504"; // Code ISO pour MAD

private String language = "fr"; // ou "ar" pour l'arabe

private boolean testMode = false;

}

1. Service de paiement CMI :

// src/main/java/com/digital/school/service/PaymentService.java

package com.digital.school.service;

import java.math.BigDecimal;

public interface PaymentService {

String initiatePayment(String orderId, BigDecimal amount, String description, String customerEmail);

boolean verifyPayment(String orderId, String transactionId);

void savePaymentReceipt(String orderId, byte[] receipt);

}<!---changer les paramètres de démarrage de $JAVA\_OPTS -->

JAVA\_OPTS=" Xms4g -Xmx6g -XX:+UseG1GC -XX:MaxGCPauseMillis=200 -XX:+PrintGCDetails -XX:+PrintGCDateStamps -XX:+PrintTenuringDistribution -Xloggc:/VAR/pfsms/jboss/log/default/gc.log -Dsun.rmi.dgc.client.gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000"

// src/main/java/com/digital/school/service/impl/CMIPaymentServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.config.CMIConfig;

import com.digital.school.service.PaymentService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.math.BigDecimal;

import java.util.Base64;

@Service

public class CMIPaymentServiceImpl implements PaymentService {

@Autowired

private CMIConfig cmiConfig;

@Autowired

private StorageService storageService;

@Override

public String initiatePayment(String orderId, BigDecimal amount, String description, String customerEmail) {

// Préparation des paramètres pour CMI

String amountStr = amount.multiply(new BigDecimal(100)).toString(); // Conversion en centimes

// Construction des paramètres de paiement

StringBuilder data = new StringBuilder();

data.append("merchantId=").append(cmiConfig.getMerchantId())

.append("&amount=").append(amountStr)

.append("&currency=").append(cmiConfig.getCurrency())

.append("&language=").append(cmiConfig.getLanguage())

.append("&orderid=").append(orderId)

.append("&customeremail=").append(customerEmail)

.append("&description=").append(description);

// Génération de la signature

String signature = generateSignature(data.toString());

// Construction de l'URL de redirection CMI

return cmiConfig.getApiUrl() + "?" + data.toString() + "&signature=" + signature;

}

@Override

public boolean verifyPayment(String orderId, String transactionId) {

// Vérification de la réponse CMI

// Implémentation de la vérification de la signature de retour

return true; // À implémenter selon la documentation CMI

}

@Override

public void savePaymentReceipt(String orderId, byte[] receipt) {

String path = "receipts/" + orderId + ".pdf";

storageService.store(path, receipt);

}

private String generateSignature(String data) {

try {

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(cmiConfig.getStoreKey().getBytes(), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

return Base64.getEncoder().encodeToString(sha256\_HMAC.doFinal(data.getBytes()));

} catch (Exception e) {

throw new RuntimeException("Erreur lors de la génération de la signature", e);

}

}

}

1. Service d'envoi d'emails avec SendinBlue :

// src/main/java/com/digital/school/service/EmailService.java

package com.digital.school.service;

import java.util.Map;

public interface EmailService {

void sendEmail(String to, String subject, String template, Map<String, Object> variables);

void sendPaymentConfirmation(String to, String orderRef, String amount);

void sendPaymentReminder(String to, String invoiceRef, String amount, String dueDate);

}

// src/main/java/com/digital/school/service/impl/SendinBlueEmailServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.service.EmailService;

import sendinblue.ApiClient;

import sendinblue.Configuration;

import sendinblue.auth.ApiKeyAuth;

import sibApi.TransactionalEmailsApi;

import sibModel.\*;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import java.util.Map;

import java.util.Arrays;

@Service

public class SendinBlueEmailServiceImpl implements EmailService {

private final TransactionalEmailsApi api;

@Value("${sendinblue.api-key}")

private String apiKey;

@Value("${sendinblue.sender.email}")

private String senderEmail;

@Value("${sendinblue.sender.name}")

private String senderName;

public SendinBlueEmailServiceImpl() {

ApiClient client = Configuration.getDefaultApiClient();

api = new TransactionalEmailsApi();

ApiKeyAuth apiKeyAuth = (ApiKeyAuth) client.getAuthentication("api-key");

apiKeyAuth.setApiKey(apiKey);

}

@Override

public void sendEmail(String to, String subject, String template, Map<String, Object> variables) {

try {

SendSmtpEmail email = new SendSmtpEmail();

email.setSender(new SendSmtpEmailSender().email(senderEmail).name(senderName));

email.setTo(Arrays.asList(new SendSmtpEmailTo().email(to)));

email.setSubject(subject);

email.setTemplateId(Long.parseLong(template));

email.setParams(variables);

api.sendTransacEmail(email);

} catch (Exception e) {

throw new RuntimeException("Erreur lors de l'envoi de l'email", e);

}

}

@Override

public void sendPaymentConfirmation(String to, String orderRef, String amount) {

Map<String, Object> variables = Map.of(

"orderRef", orderRef,

"amount", amount,

"date", java.time.LocalDateTime.now().toString()

);

sendEmail(to, "Confirmation de paiement", "payment-confirmation-template", variables);

}

@Override

public void sendPaymentReminder(String to, String invoiceRef, String amount, String dueDate) {

Map<String, Object> variables = Map.of(

"invoiceRef", invoiceRef,

"amount", amount,

"dueDate", dueDate

);

sendEmail(to, "Rappel de paiement", "payment-reminder-template", variables);

}

}

1. Service SMS avec Maroc Telecom Business SMS :

// src/main/java/com/digital/school/service/SMSService.java

package com.digital.school.service;

public interface SMSService {

void sendSMS(String to, String message);

void sendPaymentConfirmation(String to, String amount);

void sendPaymentReminder(String to, String amount, String dueDate);

}

// src/main/java/com/digital/school/service/impl/MarocTelecomSMSServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.service.SMSService;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.MediaType;

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

import java.util.Map;

@Service

public class MarocTelecomSMSServiceImpl implements SMSService {

@Value("${maroctelecom.sms.api-url}")

private String apiUrl;

@Value("${maroctelecom.sms.api-key}")

private String apiKey;

@Value("${maroctelecom.sms.sender-id}")

private String senderId;

private final RestTemplate restTemplate;

public MarocTelecomSMSServiceImpl() {

this.restTemplate = new RestTemplate();

}

@Override

public void sendSMS(String to, String message) {

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_JSON);

headers.set("Authorization", "Bearer " + apiKey);

Map<String, Object> request = Map.of(

"sender", senderId,

"recipient", formatPhoneNumber(to),

"content", message

);

HttpEntity<Map<String, Object>> entity = new HttpEntity<>(request, headers);

try {

restTemplate.postForEntity(apiUrl + "/send", entity, String.class);

} catch (Exception e) {

throw new RuntimeException("Erreur lors de l'envoi du SMS", e);

}

}

@Override

public void sendPaymentConfirmation(String to, String amount) {

String message = String.format(

"Votre paiement de %s MAD a été confirmé. Merci de votre confiance.",

amount

);

sendSMS(to, message);

}

@Override

public void sendPaymentReminder(String to, String amount, String dueDate) {

String message = String.format(

"Rappel: Un paiement de %s MAD est attendu pour le %s. Merci de régulariser votre situation.",

amount, dueDate

);

sendSMS(to, message);

}

private String formatPhoneNumber(String phone) {

// Formatage du numéro de téléphone au format marocain

phone = phone.replaceAll("[^0-9]", "");

if (phone.startsWith("0")) {

phone = "212" + phone.substring(1);

}

return phone;

}

}

1. Configurer les tâches planifiées pour les relances automatiques

Je vais vous fournir les contrôleurs, vues, JS et CSS pour la gestion des paiements, ainsi que les implémentations manquantes.

Commençons par le contrôleur de paiement :

// src/main/java/com/digital/school/controller/PaymentController.java

package com.digital.school.controller;

import com.digital.school.service.PaymentService;

import com.digital.school.service.EmailService;

import com.digital.school.service.SMSService;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.\*;

import java.math.BigDecimal;

import java.util.Map;

@Controller

@RequestMapping("/payments")

public class PaymentController {

@Autowired

private PaymentService paymentService;

@Autowired

private EmailService emailService;

@Autowired

private SMSService smsService;

@GetMapping

public String showPayments(Model model, @AuthenticationPrincipal User user) {

if (user.hasRole("ROLE\_ADMIN")) {

model.addAttribute("payments", paymentService.findAllPayments());

} else {

model.addAttribute("payments", paymentService.findPaymentsByUser(user));

}

return "payments/index";

}

@PostMapping("/initiate")

@ResponseBody

public ResponseEntity<?> initiatePayment(@RequestBody PaymentRequest request) {

try {

String paymentUrl = paymentService.initiatePayment(

request.getOrderId(),

request.getAmount(),

request.getDescription(),

request.getEmail()

);

return ResponseEntity.ok(Map.of("paymentUrl", paymentUrl));

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

@PostMapping("/callback")

@ResponseBody

public ResponseEntity<?> paymentCallback(@RequestBody CMICallback callback) {

try {

boolean isValid = paymentService.verifyPayment(

callback.getOrderId(),

callback.getTransactionId()

);

if (isValid) {

// Envoyer confirmation par email et SMS

emailService.sendPaymentConfirmation(

callback.getEmail(),

callback.getOrderId(),

callback.getAmount().toString()

);

smsService.sendPaymentConfirmation(

callback.getPhone(),

callback.getAmount().toString()

);

return ResponseEntity.ok(Map.of("status", "success"));

} else {

return ResponseEntity.badRequest()

.body(Map.of("error", "Invalid payment signature"));

}

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

@GetMapping("/receipt/{orderId}")

public ResponseEntity<?> downloadReceipt(@PathVariable String orderId) {

try {

byte[] receipt = paymentService.getReceipt(orderId);

return ResponseEntity.ok()

.header("Content-Type", "application/pdf")

.header("Content-Disposition", "attachment; filename=\"receipt-" + orderId + ".pdf\"")

.body(receipt);

} catch (Exception e) {

return ResponseEntity.badRequest()

.body(Map.of("error", e.getMessage()));

}

}

}

Implémentation du service de stockage des factures :

// src/main/java/com/digital/school/service/StorageService.java

package com.digital.school.service;

import java.io.InputStream;

import java.nio.file.Path;

import org.springframework.core.io.Resource;

public interface StorageService {

void store(String path, byte[] content);

Resource loadAsResource(String path);

void delete(String path);

String getUrl(String path);

}

// src/main/java/com/digital/school/service/impl/SupabaseStorageServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.service.StorageService;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.core.io.ByteArrayResource;

import org.springframework.core.io.Resource;

import org.springframework.http.\*;

import org.springframework.stereotype.Service;

import org.springframework.web.client.RestTemplate;

import java.util.Base64;

@Service

public class SupabaseStorageServiceImpl implements StorageService {

@Value("${supabase.url}")

private String supabaseUrl;

@Value("${supabase.key}")

private String supabaseKey;

@Value("${supabase.bucket}")

private String bucketName;

private final RestTemplate restTemplate;

public SupabaseStorageServiceImpl() {

this.restTemplate = new RestTemplate();

}

@Override

public void store(String path, byte[] content) {

String url = String.format("%s/storage/v1/object/%s/%s",

supabaseUrl, bucketName, path);

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.APPLICATION\_OCTET\_STREAM);

headers.set("Authorization", "Bearer " + supabaseKey);

HttpEntity<byte[]> requestEntity = new HttpEntity<>(content, headers);

try {

restTemplate.exchange(url, HttpMethod.POST, requestEntity, String.class);

} catch (Exception e) {

throw new RuntimeException("Failed to store file", e);

}

}

@Override

public Resource loadAsResource(String path) {

String url = String.format("%s/storage/v1/object/%s/%s",

supabaseUrl, bucketName, path);

HttpHeaders headers = new HttpHeaders();

headers.set("Authorization", "Bearer " + supabaseKey);

HttpEntity<?> requestEntity = new HttpEntity<>(headers);

try {

ResponseEntity<byte[]> response = restTemplate.exchange(

url, HttpMethod.GET, requestEntity, byte[].class);

return new ByteArrayResource(response.getBody());

} catch (Exception e) {

throw new RuntimeException("Failed to load file", e);

}

}

@Override

public void delete(String path) {

String url = String.format("%s/storage/v1/object/%s/%s",

supabaseUrl, bucketName, path);

HttpHeaders headers = new HttpHeaders();

headers.set("Authorization", "Bearer " + supabaseKey);

HttpEntity<?> requestEntity = new HttpEntity<>(headers);

try {

restTemplate.exchange(url, HttpMethod.DELETE, requestEntity, String.class);

} catch (Exception e) {

throw new RuntimeException("Failed to delete file", e);

}

}

@Override

public String getUrl(String path) {

return String.format("%s/storage/v1/object/public/%s/%s",

supabaseUrl, bucketName, path);

}

}

Implémentation de TransactionalEmailsApi :

// src/main/java/com/digital/school/service/impl/TransactionalEmailsApi.java

donne moi storage local et SendSmtpEmail, PaymentGatewayService et les htmls, css, js manquants

Je vais vous fournir les implémentations manquantes.

D'abord, l'implémentation du stockage local :

// src/main/java/com/digital/school/service/impl/LocalStorageServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.service.StorageService;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.core.io.Resource;

import org.springframework.core.io.UrlResource;

import org.springframework.stereotype.Service;

import java.io.IOException;

import java.nio.file.Files;

import java.nio.file.Path;

import java.nio.file.Paths;

import java.nio.file.StandardCopyOption;

@Service

public class LocalStorageServiceImpl implements StorageService {

private final Path rootLocation;

public LocalStorageServiceImpl(@Value("${app.storage.location}") String storageLocation) {

this.rootLocation = Paths.get(storageLocation);

try {

Files.createDirectories(rootLocation);

} catch (IOException e) {

throw new RuntimeException("Could not initialize storage location", e);

}

}

@Override

public void store(String path, byte[] content) {

try {

Path destinationFile = this.rootLocation.resolve(Paths.get(path)).normalize().toAbsolutePath();

if (!destinationFile.getParent().startsWith(this.rootLocation.toAbsolutePath())) {

throw new RuntimeException("Cannot store file outside current directory.");

}

Files.createDirectories(destinationFile.getParent());

Files.write(destinationFile, content);

} catch (IOException e) {

throw new RuntimeException("Failed to store file", e);

}

}

@Override

public Resource loadAsResource(String path) {

try {

Path file = rootLocation.resolve(path);

Resource resource = new UrlResource(file.toUri());

if (resource.exists() || resource.isReadable()) {

return resource;

} else {

throw new RuntimeException("Could not read file: " + path);

}

} catch (Exception e) {

throw new RuntimeException("Could not read file: " + path, e);

}

}

@Override

public void delete(String path) {

try {

Path file = rootLocation.resolve(path);

Files.deleteIfExists(file);

} catch (IOException e) {

throw new RuntimeException("Could not delete file: " + path, e);

}

}

@Override

public String getUrl(String path) {

return "/storage/" + path;

}

}

La classe SendSmtpEmail :

// src/main/java/com/digital/school/model/email/SendSmtpEmail.java

package com.digital.school.model.email;

import com.fasterxml.jackson.annotation.JsonProperty;

import lombok.Data;

import java.util.List;

import java.util.Map;

@Data

public class SendSmtpEmail {

private String sender;

private List<EmailTo> to;

private String subject;

private String htmlContent;

private String textContent;

private List<String> tags;

private Map<String, String> headers;

@Data

public static class EmailTo {

private String email;

private String name;

}

}

Le service de passerelle de paiement :

// src/main/java/com/digital/school/service/PaymentGatewayService.java

package com.digital.school.service;

import java.math.BigDecimal;

import java.util.Map;

public interface PaymentGatewayService {

String initializePayment(String orderId, BigDecimal amount, String description, String returnUrl);

boolean verifyPayment(String orderId, String transactionId, Map<String, String> params);

Map<String, String> getPaymentStatus(String orderId);

}

// src/main/java/com/digital/school/service/impl/CMIPaymentGatewayServiceImpl.java

package com.digital.school.service.impl;

import com.digital.school.service.PaymentGatewayService;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import javax.crypto.Mac;

import javax.crypto.spec.SecretKeySpec;

import java.math.BigDecimal;

import java.nio.charset.StandardCharsets;

import java.util.Base64;

import java.util.HashMap;

import java.util.Map;

import java.util.TreeMap;

@Service

public class CMIPaymentGatewayServiceImpl implements PaymentGatewayService {

@Value("${cmi.merchant.id}")

private String merchantId;

@Value("${cmi.secret.key}")

private String secretKey;

@Value("${cmi.gateway.url}")

private String gatewayUrl;

@Override

public String initializePayment(String orderId, BigDecimal amount, String description, String returnUrl) {

Map<String, String> params = new TreeMap<>();

params.put("merchantId", merchantId);

params.put("orderId", orderId);

params.put("amount", amount.toString());

params.put("currency", "504"); // Code pour MAD

params.put("description", description);

params.put("language", "fr");

params.put("returnUrl", returnUrl);

String signature = generateSignature(params);

params.put("signature", signature);

return buildPaymentUrl(params);

}

@Override

public boolean verifyPayment(String orderId, String transactionId, Map<String, String> params) {

String receivedSignature = params.get("signature");

params.remove("signature");

String calculatedSignature = generateSignature(params);

return receivedSignature.equals(calculatedSignature);

}

@Override

public Map<String, String> getPaymentStatus(String orderId) {

Map<String, String> params = new TreeMap<>();

params.put("merchantId", merchantId);

params.put("orderId", orderId);

String signature = generateSignature(params);

params.put("signature", signature);

// Appel à l'API CMI pour vérifier le statut

// Implémentation à adapter selon l'API CMI

return new HashMap<>();

}

private String generateSignature(Map<String, String> params) {

try {

String data = String.join("|", params.values());

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(secretKey.getBytes(StandardCharsets.UTF\_8), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

byte[] hash = sha256\_HMAC.doFinal(data.getBytes(StandardCharsets.UTF\_8));

return Base64.getEncoder().encodeToString(hash);

} catch (Exception e) {

throw new RuntimeException("Error generating signature", e);

}

}

private String buildPaymentUrl(Map<String, String> params) {

StringBuilder url = new StringBuilder(gatewayUrl + "?");

params.forEach((key, value) -> url.append(key).append("=").append(value).append("&"));

return url.substring(0, url.length() - 1);

}

}

Les vues HTML :

<!-- src/main/resources/templates/payments/index.html -->

<!DOCTYPE html>

<html lang="fr" xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Gestion des Paiements - Digital School</title>

<!-- CSS -->

<link href="https://fonts.googleapis.com/css2?family=Inter:wght@300;400;500;600;700&display=swap" rel="stylesheet">

<link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0/css/all.min.css" rel="stylesheet">

<link rel="stylesheet" th:href="@{/css/digitalSchool.css}">

<link rel="stylesheet" th:href="@{/css/components/tables.css}">

<link rel="stylesheet" th:href="@{/css/components/cards.css}">

<link rel="stylesheet" th:href="@{/css/payments/index.css}">

</head>

<body>

<div th:replace="~{fragments/sidebar :: sidebar}"></div>

<div th:replace="~{fragments/topbar :: topbar}"></div>

<div class="dashboard-container">

<div class="dashboard-header">

<h1 class="dashboard-title">

<i class="fas fa-credit-card"></i>

Gestion des Paiements

</h1>

<div sec:authorize="hasRole('ROLE\_ADMIN')" class="header-actions">

<button class="btn btn-primary" onclick="paymentManager.openPaymentModal()">

<i class="fas fa-plus"></i>

Nouveau Paiement

</button>

</div>

</div>

<!-- Stats Cards -->

<div class="stats-grid">

<div class="stat-card">

<div class="stat-icon payments">

<i class="fas fa-money-bill-wave"></i>

</div>

<div class="stat-content">

<h3>Total Payé</h3>

<div class="stat-value" th:text="${stats.totalPaid + ' MAD'}">0 MAD</div>

<div class="stat-change positive">

<i class="fas fa-arrow-up"></i>

<span th:text="${stats.paymentIncrease + '%'}">0%</span>

</div>

</div>

</div>

<div class="stat-card">

<div class="stat-icon pending">

<i class="fas fa-clock"></i>

</div>

<div class="stat-content">

<h3>En Attente</h3>

<div class="stat-value" th:text="${stats.totalPending + ' MAD'}">0 MAD</div>

<div class="stat-count" th:text="${stats.pendingCount + ' paiements'}">0 paiements</div>

</div>

</div>

<div class="stat-card">

<div class="stat-icon success">

<i class="fas fa-check-circle"></i>

</div>

<div class="stat-content">

<h3>Taux de Réussite</h3>

<div class="stat-value" th:text="${stats.successRate + '%'}">0%</div>

<div class="stat-label">des transactions</div>

</div>

</div>

</div>

<!-- Payments Table -->

<div class="content-card">

<div class="card-header">

<div class="filters">

<select id="statusFilter" class="form-select">

<option value="">Tous les statuts</option>

<option value="PAID">Payé</option>

<option value="PENDING">En attente</option>

<option value="FAILED">Échoué</option>

</select>

<select id="periodFilter" class="form-select">

<option value="7">7 derniers jours</option>

<option value="30">30 derniers jours</option>

<option value="90">90 derniers jours</option>

<option value="all">Tout</option>

</select>

<input type="text" id="searchInput" class="form-input" placeholder="Rechercher...">

</div>

</div>

<div class="table-container">

<table id="paymentsTable" class="table">

<thead>

<tr>

<th>Référence</th>

<th>Date</th>

<th>Élève</th>

<th>Parent</th>

<th>Montant</th>

<th>Statut</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

<tr th:each="payment : ${payments}">

<td th:text="${payment.reference}"></td>

<td th:text="${#temporals.format(payment.date, 'dd/MM/yyyy HH:mm')}"></td>

<td th:text="${payment.student.firstName + ' ' + payment.student.lastName}"></td>

<td th:text="${payment.parent.firstName + ' ' + payment.parent.lastName}"></td>

<td th:text="${payment.amount + ' MAD'}"></td>

<td>

<span class="status-badge" th:classappend="${payment.status.toLowerCase()}"

th:text="${payment.status}"></span>

</td>

<td>

<div class="action-buttons">

<button class="btn-icon view"

th:onclick="'paymentManager.viewDetails(\'' + ${payment.id} + '\')'">

<i class="fas fa-eye"></i>

</button>

<button class="btn-icon download"

th:onclick="'paymentManager.downloadReceipt(\'' + ${payment.id} + '\')'">

<i class="fas fa-download"></i>

</button>

<button sec:authorize="hasRole('ROLE\_ADMIN')"

class="btn-icon edit"

th:if="${payment.status == 'PENDING'}"

th:onclick="'paymentManager.editPayment(\'' + ${payment.id} + '\')'">

<i class="fas fa-edit"></i>

</button>

</div>

</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

<!-- Payment Modal -->

<div class="modal" id="paymentModal">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h3 class="modal-title">Nouveau Paiement</h3>

<button type="button" class="close-button" onclick="paymentManager.closeModal()">

<i class="fas fa-times"></i>

</button>

</div>

<form id="paymentForm">

<div class="modal-body">

<div class="form-group">

<label for="student">Élève</label>

<select id="student" name="studentId" class="form-select" required>

<option value="">Sélectionner un élève</option>

<option th:each="student : ${students}"

th:value="${student.id}"

th:text="${student.firstName + ' ' + student.lastName}">

</option>

</select>

</div>

<div class="form-group">

<label for="parent">Parent payeur</label>

<select id="parent" name="parentId" class="form-select" required>

<option value="">Sélectionner un parent</option>

</select>

</div>

<div class="form-group">

<label for="amount">Montant (MAD)</label>

<input type="number" id="amount" name="amount" class="form-input"

min="0" step="0.01" required>

</div>

<div class="form-group">

<label for="description">Description</label>

<textarea id="description" name="description" class="form-input"

rows="3" required></textarea>

</div>

</div>

<div class="modal-footer">

<button type="button" class="btn btn-secondary" onclick="paymentManager.closeModal()">

Annuler

</button>

<button type="submit" class="btn btn-primary">

Procéder au paiement

</button>

</div>

</form>

</div>

</div>

</div>

<!-- Scripts -->

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<script th:src="@{/js/payments/index.js}"></script>

</body>

</html>

Le CSS :

/\* src/main/resources/static/css/payments/index.css \*/

/\* Stats Grid \*/

.stats-grid {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 1.5rem;

margin-bottom: 2rem;

}

.stat-card {

background: white;

border-radius: 1rem;

padding: 1.5rem;

display: flex;

align-items: flex-start;

gap: 1.5rem;

box-shadow: var(--shadow-sm);

transition: transform 0.3s ease;

}

.stat-card:hover {

transform: translateY(-5px);

}

.stat-icon {

width: 48px;

height: 48px;

border-radius: 12px;

display: flex;

align-items: center;

justify-content: center;

font-size: 1.5rem;

}

.stat-icon.payments {

background-color: rgba(76, 81, 191, 0.1);

color: var(--primary-color);

}

.stat-icon.pending {

background-color: rgba(237, 137, 54, 0.1);

color: var(--warning-color);

}

.stat-icon.success {

background-color: rgba(72, 187, 120, 0.1);

color: var(--success-color);

}

.stat-content {

flex: 1;

}

.stat-content h3 {

font-size: 0.875rem;

color: var(--text-secondary);

margin-bottom: 0.5rem;

}

.stat-value {

font-size: 1.875rem;

font-weight: 700;

color: var(--text-primary);

line-height: 1.2;

margin-bottom: 0.5rem;

}

.stat-change {

display: flex;

align-items: center;

gap: 0.5rem;

font-size: 0.875rem;

font-weight: 500;

}

.stat-change.positive {

color: var(--success-color);

}

.stat-change.negative {

color: var(--danger-color);

}

/\* Filters \*/

.filters {

display: flex;

gap: 1rem;

margin-bottom: 1.5rem;

}

.form-select,

.form-input {

min-width: 200px;

}

/\* Status Badges \*/

.status-badge {

display: inline-flex;

align-items: center;

padding: 0.25rem 0.75rem;

border-radius: 9999px;

font-size: 0.75rem;

font-weight: 500;

}

.status-badge.paid {

background-color: rgba(72, 187, 120, 0.1);

color: var(--success-color);

}

.status-badge.pending {

background-color: rgba(237, 137, 54, 0.1);

color: var(--warning-color);

}

.status-badge.failed {

background-color: rgba(245, 101, 101, 0.1);

color: var(--danger-color);

}

/\* Action Buttons \*/

.action-buttons {

display: flex;

gap: 0.5rem;

}

.btn-icon {

width: 32px;

height: 32px;

border-radius: 0.375rem;

display: flex;

align-items: center;

justify-content: center;

border: none;

cursor: pointer;

transition: all 0.3s ease;

}

.btn-icon.view {

background-color: rgba(66, 153, 225, 0.1);

color: var(--info-color);

}

.btn-icon.download {

background-color: rgba(72, 187, 120, 0.1);

color: var(--success-color);

}

.btn-icon.edit {

background-color: rgba(237, 137, 54, 0.1);

color: var(--warning-color);

}

.btn-icon:hover {

transform: translateY(-2px);

}

/\* Modal Styles \*/

.modal-dialog {

max-width: 600px;

}

/\* Responsive Design \*/

@media (max-width: 768px) {

.stats-grid {

grid-template-columns: 1fr;

}

.filters {

flex-direction: column;

}

.form-select,

.form-input {

width: 100%;

}

}

Le JavaScript :

// src/main/resources/static/js/payments/index.js

donne moi : PaymentRequest CMICallback saveReceiptToStorage(receiptContent, payment.getId()); paymentRepository.getTotalPaidAmount(invoice.getId());

Je vais vous fournir les implémentations demandées.

1. PaymentRequest (Modèle pour la requête de paiement) :

// src/main/java/com/digital/school/model/payment/PaymentRequest.java

package com.digital.school.model.payment;

import lombok.Data;

import java.math.BigDecimal;

import java.time.LocalDateTime;

@Data

public class PaymentRequest {

private Long studentId;

private Long parentId;

private Long invoiceId;

private BigDecimal amount;

private String currency = "MAD"; // Dirham marocain par défaut

private String description;

private String paymentMethod = "CARD"; // Carte bancaire par défaut

private LocalDateTime requestDate = LocalDateTime.now();

private String language = "fr"; // Langue par défaut

private String returnUrl;

private String cancelUrl;

private String notifyUrl;

private Map<String, String> metadata; // Données supplémentaires

// Méthode pour générer l'ID de commande unique

public String generateOrderId() {

return String.format("ORD-%d-%d-%s",

this.studentId,

this.invoiceId,

LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"))

);

}

// Méthode pour valider la requête

public void validate() {

if (studentId == null) {

throw new IllegalArgumentException("L'ID de l'étudiant est requis");

}

if (parentId == null) {

throw new IllegalArgumentException("L'ID du parent est requis");

}

if (invoiceId == null) {

throw new IllegalArgumentException("L'ID de la facture est requis");

}

if (amount == null || amount.compareTo(BigDecimal.ZERO) <= 0) {

throw new IllegalArgumentException("Le montant doit être supérieur à 0");

}

if (description == null || description.trim().isEmpty()) {

throw new IllegalArgumentException("La description est requise");

}

}

}

1. CMICallback (Gestion des retours CMI) :

// src/main/java/com/digital/school/model/payment/CMICallback.java

package com.digital.school.model.payment;

import lombok.Data;

import java.math.BigDecimal;

@Data

public class CMICallback {

private String merchantId;

private String orderId;

private String transactionId;

private String responseCode;

private String transactionStatus;

private BigDecimal amount;

private String currency;

private String hash;

private String paymentMethod;

private String cardNumber; // Derniers 4 chiffres uniquement

private String authorizationId;

private LocalDateTime transactionDate;

// Méthodes de validation

public boolean isValid(String expectedMerchantId, String secretKey) {

if (!merchantId.equals(expectedMerchantId)) {

return false;

}

return validateHash(secretKey);

}

public boolean isSuccessful() {

return "00".equals(responseCode) && "APPROVED".equals(transactionStatus);

}

private boolean validateHash(String secretKey) {

String dataToHash = String.join("|",

merchantId,

orderId,

transactionId,

responseCode,

transactionStatus,

amount.toString(),

currency

);

String calculatedHash = generateHash(dataToHash, secretKey);

return hash.equals(calculatedHash);

}

private String generateHash(String data, String secretKey) {

try {

Mac sha256\_HMAC = Mac.getInstance("HmacSHA256");

SecretKeySpec secret\_key = new SecretKeySpec(secretKey.getBytes(StandardCharsets.UTF\_8), "HmacSHA256");

sha256\_HMAC.init(secret\_key);

byte[] hash = sha256\_HMAC.doFinal(data.getBytes(StandardCharsets.UTF\_8));

return Base64.getEncoder().encodeToString(hash);

} catch (Exception e) {

throw new RuntimeException("Erreur lors de la génération du hash", e);

}

}

}

1. Méthode saveReceiptToStorage :

// src/main/java/com/digital/school/service/impl/PaymentServiceImpl.java

@Service

@Transactional

public class PaymentServiceImpl implements PaymentService {

@Autowired

private StorageService storageService;

@Autowired

private PaymentRepository paymentRepository;

@Value("${app.storage.receipts.path}")

private String receiptsPath;

// ... autres méthodes ...

@Override

public String saveReceiptToStorage(byte[] receiptContent, Long paymentId) {

try {

// Générer un nom de fichier unique

String fileName = generateReceiptFileName(paymentId);

// Créer le chemin complet

String fullPath = generateReceiptPath(fileName);

// Sauvegarder le fichier

storageService.store(fullPath, receiptContent);

// Mettre à jour le paiement avec le chemin du reçu

Payment payment = paymentRepository.findById(paymentId)

.orElseThrow(() -> new RuntimeException("Paiement non trouvé"));

payment.setReceiptPath(fullPath);

paymentRepository.save(payment);

return fullPath;

} catch (Exception e) {

throw new RuntimeException("Erreur lors de la sauvegarde du reçu", e);

}

}

private String generateReceiptFileName(Long paymentId) {

return String.format("receipt-%d-%s.pdf",

paymentId,

LocalDateTime.now().format(DateTimeFormatter.ofPattern("yyyyMMddHHmmss"))

);

}

private String generateReceiptPath(String fileName) {

return String.format("%s/%s/%s",

receiptsPath,

LocalDate.now().format(DateTimeFormatter.ofPattern("yyyy/MM")),

fileName

);

}

}

1. Méthode getTotalPaidAmount dans le repository :

// src/main/java/com/digital/school/repository/PaymentRepository.java

@Repository

public interface PaymentRepository extends JpaRepository<Payment, Long> {

@Query("""

SELECT COALESCE(SUM(p.amount), 0)

FROM Payment p

WHERE p.invoice.id = :invoiceId

AND p.status = 'PAID'

""")

BigDecimal getTotalPaidAmount(@Param("invoiceId") Long invoiceId);

// Méthode surchargée avec période

@Query("""

SELECT COALESCE(SUM(p.amount), 0)

FROM Payment p

WHERE p.invoice.id = :invoiceId

AND p.status = 'PAID'

AND p.paymentDate BETWEEN :startDate AND :endDate

""")

BigDecimal getTotalPaidAmount(

@Param("invoiceId") Long invoiceId,

@Param("startDate") LocalDateTime startDate,

@Param("endDate") LocalDateTime endDate

);

// Méthode pour obtenir le montant total payé par étudiant

@Query("""

SELECT COALESCE(SUM(p.amount), 0)

FROM Payment p

WHERE p.student.id = :studentId

AND p.status = 'PAID'

""")

BigDecimal getTotalPaidAmountByStudent(@Param("studentId") Long studentId);

// Méthode pour obtenir le montant total payé par parent

@Query("""

SELECT COALESCE(SUM(p.amount), 0)

FROM Payment p

WHERE p.parent.id = :parentId

AND p.status = 'PAID'

""")

BigDecimal getTotalPaidAmountByParent(@Param("parentId") Long parentId);

}

Ces implémentations fournissent :

1. Un modèle complet pour les requêtes de paiement avec validation
2. Une gestion sécurisée des callbacks CMI avec validation du hash
3. Un système de stockage des reçus avec organisation par date
4. Des méthodes de calcul des montants payés avec différents filtres

Voulez-vous des détails supplémentaires sur l'une de ces implémentations ?

es implémentations fournissent une base solide pour la gestion des paiements avec :

* Stockage local des fichiers
* Envoi d'emails transactionnels via SendinBlue
* Intégration avec la passerelle de paiement CMI
* Interface utilisateur complète pour la gestion des paiements
* Gestion des reçus et notifications

Voulez-vous que je détaille une partie spécifique ou que j'ajoute d'autres fonctionnalités ?