CAHIER DES CHARGES TECHNIQUE & FONCTIONNEL : JOUAN-SUGU

Version : 2.0 (Finale)

Technologie : Laravel + MySQL

Objectif : Plateforme E-commerce & Fintech Hybride

---

1. VISION DU PROJET

Jouan-Sugu est une place de marché numérique permettant la vente de produits physiques, de services et d'actions d'entreprises. Elle intègre un système de portefeuille électronique (Wallet) et une gestion rigoureuse des licences de vente par commissions (1% pour les locaux, 2% pour les entreprises).

---

2. ARCHITECTURE DES UTILISATEURS (RÔLES)

- **Super-Admin :** Gère la plateforme, valide les licences et surveille les audits financiers.
- **Vendeur Entreprise :** Vend produits/services/actions. Commission : **2%**. Peut avoir des assistants.
- **Vendeur Local :** Particulier vendant des produits locaux. Commission : **1%**.
- **Assistant :** Employé d'un vendeur avec accès limité au stock et aux commandes.
- **Acheteur :** Client final disposant d'un portefeuille rechargeable.

---

3. CONCEPTION DE LA BASE DE DONNÉES

Schéma des Tables et Rôles

Table	Rôle en Français	Pourquoi ?
`users`	Identité de tous les membres	Centraliser les accès (Acheteur, Vendeur, Admin).
`wallets`	Comptes bancaires internes	Gérer l'argent, les dépôts et les paiements directs.
`transactions`	Journal financier	Historique de chaque centime (Brut, Commission, Net).
`products`	Catalogue des articles	Gérer les prix, descriptions et types (Produit/Service).
`stock_movements`	Gestion des stocks	Traçabilité des entrées (arrivages) et sorties (ventes).

Table	Rôle en Français	Pourquoi ?
`licences`	Statut de vente	Gérer les frais d'accès et le délai de 3 mois.
`audit_logs`	Boîte noire du système	Enregistrer qui a modifié quoi (Prix, Stock, Argent).
`orders`	Commandes clients	Suivi des achats et de l'état de livraison.

Table	Rôle en Français	Importance pour Jouan-Sugu
categories	Organisation du catalogue	Permet de filtrer par Produits, Services ou Actions d'entreprises.
coupons	Marketing & Promotions	Gestion des codes promos (ex: -10%) pour booster les ventes.
payout_requests	Demandes de retrait	Gère le passage de l'argent virtuel (Wallet) vers l'argent réel (Orange Money).
licences	Suivi des droits de vente	Contrôle la validité du compte vendeur et le délai de grâce de 3 mois.
access_attempts	Sécurité (Tentatives d'accès)	Enregistre les échecs de connexion pour bloquer les tentatives de piratage.

4. SCRIPT DE STRUCTURE (MIGRATIONS LARAVEL)

Modèle Conceptuel de Données (MCD) - Jouan-Sugu

Entités Principales et Attributs :

- **UTILISATEUR :** Identifiant, Nom, Téléphone, Mot de passe, Rôle (Superadmin, Entreprise, Local, Acheteur, Assistant).
- **PORTEFEUILLE (Wallet) :** Identifiant, Solde, Devise.
- **PRODUIT :** Identifiant, Nom, Prix, Stock actuel, Type (Physique, Service, Action).
- **CATÉGORIE :** Identifiant, Nom, Slug, Type.
- **LICENCE :** Identifiant, État (Active/Inactive), Date dernier paiement, Date expiration, Frais de maintenance.
- **COUPON :** Identifiant, Code, Type (Fixe/%), Valeur, Limite d'usage, Date expiration.

Entités de Suivi (Historique & Sécurité) :

- **TRANSACTION :** Identifiant, Montant Brut, Commission, Montant Net, Type.
- **MOUVEMENT_STOCK :** Identifiant, Quantité (+/-), Type, Raison.
- **DEMANDE_RETRAIT :** Identifiant, Montant, Méthode, Numéro compte, Statut, Note Admin.
- **LOG_AUDIT :** Identifiant, Action, Table, Anciennes valeurs, Nouvelles valeurs.
- **TENTATIVE_ACCES :** Identifiant, Identifiant utilisé, Adresse IP, Succès (Oui/Non), Navigateur.

---

2. Relations (Associations) entre les entités

Voici comment les données communiquent entre elles :

1. **Utilisateur ↔ Portefeuille :** Un Utilisateur **possède** un seul Portefeuille (1:1).
2. **Utilisateur ↔ Licence :** Un Vendeur (Local ou Entreprise) **détient** une Licence (1:1).
3. **Utilisateur ↔ Produit :** Un Vendeur **propose** plusieurs Produits (1:N).
4. **Catégorie ↔ Produit :** Une Catégorie **regroupe** plusieurs Produits (1:N).
5. **Portefeuille ↔ Transaction :** Un Portefeuille **enregistre** plusieurs Transactions (1:N).
6. **Produit ↔ Mouvement_Stock :** Un Produit **subit** plusieurs Mouvements de stock (1:N).
7. **Utilisateur ↔ Demande_Retrait :** Un Utilisateur **effectue** plusieurs Demandes de retrait (1:N).
8. **Utilisateur ↔ Log_Audit :** Un Utilisateur **génère** des traces d'audit (1:N).

---

3. Modèle Conceptuel de Traitement (MCT)

Le MCT explique **quand** et **comment** les données changent lors d'un événement. Voici les 3 flux majeurs de Jouan-Sugu :

Flux A : L'Achat d'un Produit

- **Événement :** L'acheteur valide son panier.
- **Action :** 1. Vérifier le solde du **Portefeuille** acheteur. 2. SI solde suffisant : * Débiter l'acheteur (Montant brut). * Calculer la commission (1% ou 2%). * Créditer le vendeur (Montant net). * Enregistrer la **Transaction**. * Créer un **Mouvement de Stock** (- Quantité).
- **Résultat :** Commande validée et stock mis à jour.

Flux B : Gestion de la Licence

- **Événement :** Arrivée au terme des 3 mois.
- **Action :**

1. Le système parcourt la table **Licences**.
2. SI aucune transaction de vente liée au vendeur n'existe depuis 90 jours :
    - Passer `is_active` à "Faux".
    - Envoyer une notification pour paiement des frais de maintenance.

- **Résultat :** Vendeur suspendu jusqu'au paiement.

Flux C : Retrait d'argent (Payout)

- **Événement :** Le vendeur demande un retrait.
- **Action :**
    1. Créer une **Demande_Retrait** (Statut : En attente).
    2. Le Superadmin vérifie les **Log_Audit** pour valider la provenance des fonds.
    3. SI OK : Marquer "Complété" et déduire le montant du **Portefeuille**.
- **Résultat :** Argent transféré sur son Mobile Money.

Voici le code que les développeurs devront exécuter pour créer une base de données cohérente.

PHP

```
// 1. UTILISATEURS ET PORTFEUILLES
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('phone')->unique();
    $table->enum('role', ['superadmin', 'business', 'local', 'buyer', 'assistant']);
    $table->string('password');
    $table->timestamps();
});

Schema::create('wallets', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->decimal('balance', 15, 2)->default(0); // Gestion monétaire précise
    $table->timestamps();
});
```

```
// 2. PRODUITS ET MOUVEMENTS DE STOCK
Schema::create('products', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained(); // Le vendeur
    $table->string('name');
    $table->decimal('price', 15, 2);
    $table->integer('current_stock')->default(0);
    $table->enum('type', ['physical', 'service', 'action']);
    $table->timestamps();
});

Schema::create('stock_movements', function (Blueprint $table) {
    $table->id();
    $table->foreignId('product_id')->constrained();
    $table->foreignId('user_id')->constrained(); // L'auteur de l'action
    $table->integer('quantity'); // Ex: +10 ou -2
    $table->string('reason'); // Ex: "Entrée nouvel arrivage" ou "Vente"
    $table->timestamps();
});

// 3. TRANSACTIONS ET COMMISSIONS
Schema::create('transactions', function (Blueprint $table) {
    $table->id();
    $table->foreignId('wallet_id')->constrained();
    $table->decimal('amount_gross', 15, 2); // Montant payé par client
    $table->decimal('commission_fee', 15, 2); // Le 1% ou 2%
    $table->decimal('amount_net', 15, 2); // Ce que le vendeur reçoit
    $table->enum('type', ['deposit', 'payment', 'commission', 'withdrawal']);
    $table->timestamps();
});

// 4. AUDIT ET SÉCURITÉ
```

```
Schema::create('audit_logs', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->nullable();
    $table->string('action'); // Ex: "Changement de prix"
    $table->string('table_name');
    $table->json('old_values')->nullable();
    $table->json('new_values')->nullable();
    $table->timestamps();
});
// --- 1. GESTION DU CATALOGUE ---
Schema::create('categories', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('slug')->unique();
    $table->enum('type', ['product', 'service', 'action']);
    $table->timestamps();
});

// --- 2. GESTION DES LICENCES ET DROITS ---
Schema::create('licences', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->boolean('is_active')->default(false);
    $table->timestamp('last_payment_at')->nullable();
    $table->timestamp('expires_at')->nullable(); // Gestion des 3 mois
    $table->decimal('maintenance_fee', 10, 2)->default(0);
    $table->timestamps();
});

// --- 3. GESTION DES PROMOTIONS (COUPONS) ---
Schema::create('coupons', function (Blueprint $table) {
    $table->id();
```

```
    $table->string('code')->unique();
    $table->enum('type', ['fixed', 'percentage']); // Montant fixe ou %
    $table->decimal('value', 10, 2);
    $table->integer('limit_usage')->default(1); // Nombre d'utilisations max
    $table->timestamp('expires_at')->nullable();
    $table->timestamps();
});

// --- 4. GESTION DES RETRAITS D'ARGENT (PAYOUTS) ---
Schema::create('payout_requests', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained();
    $table->decimal('amount', 15, 2);
    $table->string('payment_method'); // ex: Orange Money
    $table->string('account_number'); // Numéro de téléphone de réception
    $table->enum('status', ['pending', 'approved', 'rejected', 'completed'])->default('pending');
    $table->text('admin_note')->nullable(); // Raison en cas de rejet
    $table->timestamps();
});

// --- 5. SÉCURITÉ ET TENTATIVES D'ACCÈS ---
Schema::create('access_attempts', function (Blueprint $table) {
    $table->id();
    $table->string('email_or_phone');
    $table->ipAddress('ip_address');
    $table->boolean('was_successful');
    $table->string('user_agent'); // Navigateur utilisé
    $table->timestamps(); // Permet de bloquer après X tentatives en Y minutes
});
```

Table	Colonne (Champ)	Type de donnée	Contrainte	Rôle
users	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	phone	VARCHAR(20)	Unique	Identifiant de connexion
	role	ENUM	Not Null	superadmin, business, local, buyer, assistant
wallets	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	user_id	BIGINT (FK)	Relie à users	Propriétaire du compte
	balance	DECIMAL(15,2)	Default 0	Solde monétaire
categories	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	type	ENUM	product, service, action	Type d'offre
products	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	user_id	BIGINT (FK)	Relie à users	Le vendeur propriétaire
	category_id	BIGINT (FK)	Relie à categories	Classement
	price	DECIMAL(15,2)	Not Null	Prix de vente
transactions	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	wallet_id	BIGINT (FK)	Relie à wallets	Portefeuille concerné
	amount_gross	DECIMAL(15,2)		Montant payé par client
	commission_fee	DECIMAL(15,2)		Le 1% ou 2% prélevé
	amount_net	DECIMAL(15,2)		Montant restant au vendeur

Table	Colonne (Champ)	Type de donnée	Contrainte	Rôle
licences	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	user_id	BIGINT (FK)	Relie à users	Vendeur sous licence
	expires_at	DATETIME	Nullable	Échéance des 3 mois
stock_movements	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	product_id	BIGINT (FK)	Relie à products	Produit mouvementé
	quantity	INT		Ex: +50 (entrée), -1 (vente)
audit_logs	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	old_values	JSON	Nullable	Données avant modif
	new_values	JSON	Nullable	Données après modif

Table	Colonne (Champ)	Type de donnée	Contrainte	Rôle
coupons	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	code	VARCHAR(50)	Unique, Index	Le code à saisir (ex: SOLDES2026)
	type	ENUM	fixed, percentage	Type de remise
	value	DECIMAL(10,2)	Not Null	Valeur de la réduction
	expires_at	DATETIME	Nullable	Date de fin de validité
payout_requests	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	user_id	BIGINT (FK)	Relie à users	Demandeur du retrait
	amount	DECIMAL(15	Not Null	Somme demandée

Table	Colonne (Champ)	Type de donnée	Contrainte	Rôle
		,2)		
	status	ENUM	pending, approved, rejected, completed	État de la demande
	payment_method	VARCHAR(50)		ex: "Orange Money", "Wave"
access_attempts	id	BIGINT (PK)	Auto-incrément	Identifiant unique
	email_or_phone	VARCHAR(100)	Index	Identifiant testé
	ip_address	VARCHAR(45)		Adresse IP du visiteur
	was_successful	BOOLEAN		Réussite ou échec
	user_agent	TEXT		Navigateur/ Appareil utilisé

Focus sur les Règles Techniques Spécifiques

A. La Table `licences` (Logique de temps)

- **Champs techniques :** `last_payment_at` (Dernier paiement) et `expires_at` (Date de fin).
- **Règle technique :** Une tâche planifiée (Task Scheduling dans Laravel) doit vérifier chaque jour si `NOW() > expires_at`. Si c'est le cas, le champ `is_active` passe à `0` (False).

B. La Table `access_attempts` (Sécurité brute)

- **Indexation :** Le champ `ip_address` et `email_or_phone` doivent être indexés.
- **Utilité :** Cela permet aux développeurs de faire une requête ultra-rapide du type : *"Compter le nombre d'échecs pour cette IP dans les 15 dernières minutes"*. Si le compte est > 5, on bloque l'accès.

C. La Table `payout_requests` (Double validation)

- **Règle métier :** Le montant `amount` ne peut pas être supérieur au `balance` (solde) disponible dans la table `wallets` de l'utilisateur au moment de la demande.

- **Champ `admin_note` :** Indispensable pour que le Super-Admin explique au vendeur pourquoi son retrait a été refusé (ex: "Documents d'entreprise non conformes").

---

3. Résumé des Clés Étrangères (Relations Physiques) restantes

- **Lien Utilisateur-Licence :** `users.id` (1) <---> (1) `licences.user_id`
- **Lien Utilisateur-Retrait :** `users.id` (1) <---> (N) `payout_requests.user_id`
- **Lien Utilisateur-Audit :** `users.id` (1) <---> (N) `audit_logs.user_id`

5. FONCTIONNALITÉS CLÉS DÉTAILLÉES

A. Système de Maintenance de Licence

Pour maintenir le droit de vendre, le système vérifie :

- **Frais initiaux :** Payés lors de l'inscription.
- **Délai de 3 mois :** Si aucune vente n'est faite, le vendeur doit payer des frais de maintenance.
- **Prélèvement automatique :** Lors de chaque vente via Wallet, Laravel retire automatiquement les **1% (Local)** ou **2% (Entreprise)**.

B. Gestion des Stocks "Clairvoyante"

Chaque fois qu'un vendeur reçoit de la marchandise, il doit enregistrer une **Entrée de Stock**.

- Le système crée automatiquement une ligne dans `stock_movements`.
- Le Super-Admin peut ainsi voir si un vendeur gonfle ses stocks artificiellement.

C. Assistants IA

- **IA Vendeur :** Aide à rédiger les descriptions et suggère des catégories en fonction de l'image du produit.
- **IA Support :** Répond aux acheteurs sur l'état de leur portefeuille ou de leur livraison.

D. Audit Multi-niveaux

- **Audit Admin :** Surveillance des flux financiers totaux.
- **Audit Vendeur :** Historique des accès des assistants et des modifications de prix.

Flux de Retrait d'Argent (Payout Workflow)

1. Le vendeur demande un retrait depuis son Wallet.
2. Le montant est "gelé" (en attente) dans le système.
3. Le Super-Admin reçoit une notification, vérifie l'Audit du vendeur pour s'assurer que l'argent provient de ventes réelles.

4. Une fois validé, le transfert est effectué et le statut passe à `completed`.

Gestion Dynamique des Licences

- **Activation :** Le vendeur paie un montant fixe initial.
- **Maintenance :** Le système vérifie chaque mois si le vendeur a réalisé des ventes. Si aucune vente n'est faite pendant 3 mois, la licence est suspendue jusqu'au paiement des frais de maintenance.

Sécurité Anti-Intrusion

- La table `access_attempts` permet de mettre en place un **Throttling** (limitation) : après 5 tentatives infructueuses sur un même compte ou une même IP, l'accès est bloqué pendant 30 minutes.

Marketing : Système de Coupons

- Les coupons peuvent être créés par le Super-Admin (pour toute la plateforme) ou par les Entreprises (pour leurs propres produits uniquement).
- Le système calcule la réduction avant d'appliquer la commission de 1% ou 2%.

---

6. OUTILS DE DÉVELOPPEMENT

- **Framework :** Laravel 11 (Stabilité et Sécurité).
- **Base de données :** MySQL 8 (Gestion des transactions ACID).
- **Interface :** Tailwind CSS + Livewire (Expérience fluide sans rechargement).
- **Paiement :** API Mobile Money (Orange/Moov/Wave).