Phase 1.2 - Gestion des Refunds et Disputes

📋 Vue d'ensemble

Cette phase implémente la gestion complète des remboursements (refunds) et contestations (disputes) dans le contexte du modèle **Separate Charges and Transfers**, où le créateur a déjà reçu son paiement (85%) avant qu'un refund ou une dispute ne survienne.

🎯 Objectif

Gérer la récupération des fonds du créateur lorsqu'un refund ou une dispute survient après que le créateur ait déjà reçu son paiement.

🚀 Implémentation

1. Modèles Prisma

Refund (Mis à jour)

```
model Refund {
  // Champs existants...

  // ✅ Nouveaux champs pour gestion de la dette
  creatorDebt     Decimal      @default(0) @db.Decimal(10, 2)
  reconciled      Boolean      @default(false)
  reconciledAt    DateTime?
  reconciledBy    String?      // "TRANSFER_REVERSAL", "PAYOUT_DEDUCTION", "MANUAL"
  reversalId      String?      // Stripe Transfer Reversal ID
}
```

Dispute (Mis à jour)

```
model Dispute {
  // Champs existants...

  // ✅ Nouveaux champs pour gestion de la dette
  creatorDebt     Decimal      @default(0) @db.Decimal(10, 2)
  reconciled      Boolean      @default(false)
  reconciledAt    DateTime?
  reconciledBy    String?      // "TRANSFER_REVERSAL", "PAYOUT_DEDUCTION", "MANUAL"
  reversalId      String?      // Stripe Transfer Reversal ID
}
```

2. Helper Functions (`lib/creator-debt.ts`)

Fonctions principales :

1. `calculateCreatorDebt(amount: number): number`
   - Calcule la dette du créateur (85% du montant remboursé)
   - Le créateur a reçu 85%, il doit donc rembourser 85%

2. `attemptTransferReversal(transferId: string, amountInCents: number)`
   - Tente de créer un Transfer Reversal via Stripe API
   - Possible uniquement dans les **180 jours** suivant le transfer
   - Retourne `{ success: boolean, reversalId?: string, error?: string }`

3. `getCreatorUnreconciledDebt(creatorId: string)`
   - Récupère toutes les dettes non réconciliées d'un créateur
   - Retourne refunds et disputes avec montants totaux

4. `markRefundAsReconciled(refundId: string, reconciledBy: string, reversalId?: string)`
   - Marque un refund comme réconcilié
   - `reconciledBy` : TRANSFER_REVERSAL | PAYOUT_DEDUCTION | MANUAL

5. `markDisputeAsReconciled(disputeId: string, reconciledBy: string, reversalId?: string)`
   - Marque une dispute comme réconciliée
   - `reconciledBy` : TRANSFER_REVERSAL | PAYOUT_DEDUCTION | MANUAL

6. `checkAndBlockPayouts(creatorId: string, threshold: number = 100)`
   - Bloque les payouts si la dette dépasse le seuil (défaut: 100 EUR)
   - Envoie notifications au créateur et aux admins

7. `checkAndUnblockPayouts(creatorId: string)`
   - Débloque les payouts si la dette est remboursée

8. `notifyDebt(creatorId: string, type: 'REFUND' | 'DISPUTE', amount: number, reason: string)`
   - Notifie créateur et admins d'une nouvelle dette

3. Webhooks Stripe

`charge.refunded` (Mis à jour)

Flux :
1. Récupérer le payment et calculer `creatorDebt = 85% du montant remboursé`
2. Créer/mettre à jour l'enregistrement Refund avec `creatorDebt` et `reconciled = false`
3. **Tenter Transfer Reversal** si `payment.transferId` existe :
- ✅ **Succès** → Marquer `reconciled = true`, `reconciledBy = 'TRANSFER_REVERSAL'`
- ❌ **Échec** → Enregistrer dette pour déduction future
4. Vérifier si payouts doivent être bloqués (`checkAndBlockPayouts`)
5. Notifier créateur et admins

Cas d'échec du Transfer Reversal :
- Transfer > 180 jours
- Solde insuffisant sur le compte connecté
- Transfer déjà inversé

`charge.dispute.created` (Mis à jour)

Flux :
1. Récupérer le payment et calculer `creatorDebt = 85% du montant contesté`
2. Créer l'enregistrement Dispute avec `creatorDebt` et `reconciled = false`
3. **Ne pas encore tenter de Transfer Reversal** (on attend l'issue de la dispute)
4. Vérifier si payouts doivent être bloqués (`checkAndBlockPayouts`)
5. Notifier créateur et admins (alerte critique)

`charge.dispute.closed` (Mis à jour)

Flux si dispute WON (gagnée) :
1. Marquer `reconciled = true`, `reconciledBy = 'MANUAL'`
2. Débloquer payouts si dette totale = 0
3. Notifier créateur et admins

Flux si dispute LOST (perdue) :
1. **Tenter Transfer Reversal** si `payment.transferId` existe :
- ✅ **Succès** → Marquer `reconciled = true`, `reconciledBy = 'TRANSFER_REVERSAL'`
- ❌ **Échec** → Enregistrer dette pour déduction future
2. Vérifier si payouts doivent être bloqués (`checkAndBlockPayouts`)
3. Notifier créateur et admins

4. Déduction Automatique sur Payouts

Fichier : `app/api/payouts/request/route.ts`

Logique :
1. **Avant de créer un payout**, vérifier les dettes non réconciliées
2. **Déduire automatiquement** :
- Déduire d'abord des refunds non réconciliés
- Puis des disputes non réconciliés
- Gestion de **déductions partielles** si payout < dette
3. **Marquer comme réconcilié** :
- Si dette totalement déduite → `reconciledBy = 'PAYOUT_DEDUCTION'`
- Si partiellement déduite → Mettre à jour `creatorDebt` restant
4. **Débloquer payouts** si dette totale = 0
5. **Notifier le créateur** du montant déduit

Exemple :

```
Payout demandé: 100 EUR
Dette refund: 30 EUR
Dette dispute: 20 EUR
→ Payout final: 50 EUR
→ Refund et dispute marqués comme réconciliés
```

5. UI Admin

Page : `/dashboard/admin/refunds-disputes`

Fonctionnalités :
- ✅ **Vue d'ensemble** :
- Total refunds/disputes
- Nombre de non réconciliés
- Dette totale non réconciliée
- ✅ **Filtres** :
- Tous / Non réconciliés / Réconciliés
- ✅ **Tableau refunds** :
- Date, Créateur, Client, Offre
- Montant, Dette créateur
- Statut de réconciliation
- Méthode de réconciliation (Transfer Reversal, Déduction payout, Manuel)

- ✅ **Tableau disputes** :
- Mêmes colonnes que refunds
- Raison de la dispute
- ✅ **Action admin** :
- Bouton “Marquer réconcilié” manuellement si nécessaire

API Endpoint :
- `GET /api/admin/refunds-disputes` - Récupérer tous les refunds/disputes
- `PATCH /api/admin/refunds-disputes` - Marquer comme réconcilié manuellement

🔄 Flux complets

Refund complet (100 EUR)

Situation initiale :
- Client paie 100 EUR
- Créateur reçoit 85 EUR via Transfer (immédiat)
- Plateforme garde 15 EUR de commission

Refund demandé :
1. Client demande refund de 100 EUR
2. Webhook `charge.refunded` reçu
3. **Dette créateur calculée** : 85 EUR (85% de 100 EUR)
4. **Transfer Reversal tenté** :
- ✅ Si < 180 jours → Reversal réussit → Dette réconciliée
- ❌ Si > 180 jours → Reversal échoue → Dette enregistrée
5. **Si dette enregistrée** :
- Créateur reçoit notification
- Payouts bloqués si dette > 100 EUR
- Dette sera déduite du prochain payout

Dispute perdue

Situation initiale :
- Client paie 100 EUR
- Créateur reçoit 85 EUR via Transfer (immédiat)

Dispute créée :
1. Client conteste le paiement
2. Webhook `charge.dispute.created` reçu
3. **Dette potentielle calculée** : 85 EUR
4. **Payouts bloqués** si dette > 100 EUR
5. Notifications envoyées (alerte critique)

Dispute perdue :
1. Dispute résolue en faveur du client
2. Webhook `charge.dispute.closed` reçu (status = lost)
3. **Transfer Reversal tenté** :
- ✅ Si succès → Dette réconciliée
- ❌ Si échec → Dette enregistrée pour déduction
4. Notifications envoyées

📊 Stratégies de réconciliation

1. Transfer Reversal (Priorité 1)

- **Quand** : Dans les 180 jours du transfer original
- **Comment** : `stripe.transfers.createReversal()`
- **Avantage** : Récupération immédiate
- **Limitation** : Fenêtre de 180 jours

2. Payout Deduction (Priorité 2)

- **Quand** : Transfer Reversal impossible ou échoué
- **Comment** : Déduction automatique du prochain payout
- **Avantage** : Récupération garantie (si le créateur continue)
- **Limitation** : Peut prendre du temps

3. Manual (Priorité 3)

- **Quand** : Intervention admin nécessaire
- **Comment** : Admin marque manuellement comme réconcilié
- **Avantage** : Flexibilité pour cas exceptionnels
- **Limitation** : Requiert intervention manuelle

🔒 Sécurité et notifications

Notifications créateur :

1. **Dette enregistrée** : Email + notification in-app
2. **Payouts bloqués** : Email détaillé avec raison
3. **Déduction payout** : Notification du montant déduit
4. **Payouts débloqués** : Notification de déblocage

Notifications admin :

1. **Nouvelle dette** : Alerte sur nouvelle dette créateur
2. **Payouts bloqués** : Alerte blocage créateur
3. **Dispute créée** : Alerte critique (🚨)
4. **Transfer Reversal échoué** : Alerte pour intervention

📝 Notes importantes

1. **Transfer Reversal** :
   - Possible uniquement dans les **180 jours**
   - Nécessite solde suffisant sur compte connecté
   - Peut échouer pour raisons techniques Stripe

2. **Seuil de blocage** :
   - Par défaut : **100 EUR**
   - Configurable via paramètre

3. **Déductions partielles** :
   - Si payout < dette → Déduction partielle
   - Dette restante mise à jour pour prochaine fois

4. **Compatibilité** :
   - Fonctionne avec multi-currency (EUR, CHF, USD, GBP, etc.)
   - Calculs toujours en EUR (devise de la DB)

🧪 Tests à effectuer

Tests manuels :

1. ✅ Simuler un refund < 180 jours (Transfer Reversal réussit)
2. ✅ Simuler un refund > 180 jours (Transfer Reversal échoue, dette enregistrée)
3. ✅ Simuler une dispute créée puis perdue
4. ✅ Simuler une dispute créée puis gagnée
5. ✅ Simuler un payout avec dette > payout (déduction partielle)
6. ✅ Simuler un payout avec dette < payout (déduction totale)
7. ✅ Vérifier blocage/déblocage automatique des payouts
8. ✅ Vérifier notifications créateur et admin

Tests Stripe CLI :

```
# Simuler refund
stripe trigger charge.refunded

# Simuler dispute created
stripe trigger charge.dispute.created

# Simuler dispute closed (won)
stripe trigger charge.dispute.closed --add charge_dispute:status=won

# Simuler dispute closed (lost)
stripe trigger charge.dispute.closed --add charge_dispute:status=lost
```

📦 Fichiers modifiés/créés

Modifiés :

- `prisma/schema.prisma` - Ajout champs creatorDebt et reconciled
- `app/api/payments/webhook/route.ts` - Webhooks refund et dispute
- `app/api/payouts/request/route.ts` - Déduction automatique
- `app/api/admin/payouts/dashboard/route.ts` - Fix TypeScript
- `app/dashboard/admin/page.tsx` - Fix TypeScript

Créés :

- `lib/creator-debt.ts` - Helper functions gestion dettes
- `app/api/admin/refunds-disputes/route.ts` - API admin
- `app/dashboard/admin/refunds-disputes/page.tsx` - UI admin

Migrations :

- `20251227105047_add_creator_debt_tracking/migration.sql`

✅ Statut : TERMINÉ

Toutes les fonctionnalités de la Phase 1.2 sont implémentées et testées avec succès.

Date de complétion : 27 décembre 2025
Version : 1.2.0
Auteur : DeepAgent