# Phase 3 - Tests du workflow Payout

## ✅ Tests à effectuer

### 1. Test de création de demande de payout

**Endpoint:** `POST /api/payouts/request`
**Utilisateur:** Créateur authentifié

**Étapes:**
1. Se connecter en tant que créateur
2. Vérifier qu'il y a un solde disponible dans Stripe Connect
3. Envoyer une demande de payout
4. Vérifier que le statut est `REQUESTED`
5. Vérifier que `requestedAt` est défini
6. Vérifier qu'un PayoutAuditLog avec action `TRIGGERED` est créé avec `payoutId`

**Résultat attendu:**

```
{
  "success": true,
  "payout": {
    "id": "payout_xxx",
    "status": "pending_approval",
    "amountEur": 50.00,
    "currency": "EUR"
  }
}
```

**Vérifications base de données:**

```
SELECT * FROM "Payout" WHERE status = 'REQUESTED';
SELECT * FROM "PayoutAuditLog" WHERE action = 'TRIGGERED' AND "payoutId" IS NOT NULL;
SELECT * FROM "Notification" WHERE type = 'PAYOUT_REQUEST';
```

---

### 2. Test de liste des payouts créateur

**Endpoint:** `GET /api/payouts/creator`
**Utilisateur:** Créateur authentifié

**Étapes:**
1. Se connecter en tant que créateur
2. Récupérer la liste des payouts
3. Vérifier que les payouts sont triés par `requestedAt` DESC
4. Vérifier que le summary contient `totalPaid`, `totalRequested`, etc.

**Résultat attendu:**

```
{
  "payouts": [
    {
      "id": "payout_xxx",
      "status": "REQUESTED",
      "amount": 50.00,
      "currency": "EUR",
      "requestedAt": "2025-12-27T...",
      "approvedBy": null
    }
  ],
  "summary": {
    "totalPaid": 0,
    "totalRequested": 50.00,
    "totalApproved": 0,
    "totalRejected": 0
  }
}
```

---

## 3. Test de liste des payouts admin

**Endpoint:** `GET /api/admin/payouts`
**Utilisateur:** Admin authentifié

**Étapes:**
1. Se connecter en tant qu'admin
2. Récupérer tous les payouts
3. Tester les filtres par statut: `?status=REQUESTED`
4. Vérifier que les relations `creator.user` sont incluses
5. Vérifier que `approvedBy` est inclus

**Résultat attendu:**

```
[
  {
    "id": "payout_xxx",
    "status": "REQUESTED",
    "amount": 50.00,
    "currency": "EUR",
    "requestedAt": "2025-12-27T...",
    "creator": {
      "id": "creator_xxx",
      "user": {
        "id": "user_xxx",
        "name": "Créateur Test",
        "email": "creator@test.com"
      }
    },
    "approvedBy": null
  }
]
```

---

## 4. Test d'approbation de payout

**Endpoint:** `POST /api/admin/payouts/[id]/approve`
**Utilisateur:** Admin authentifié

**Étapes:**
1. Se connecter en tant qu'admin
2. Sélectionner un payout avec status `REQUESTED`
3. Envoyer la demande d'approbation
4. Vérifier que le statut passe à `APPROVED` puis `PROCESSING`
5. Vérifier que `approvedAt` est défini
6. Vérifier que `approvedById` contient l'ID de l'admin
7. Vérifier qu'un `stripePayoutId` est créé
8. Vérifier qu'un PayoutAuditLog est créé avec status `PROCESSING` et `payoutId`
9. Vérifier qu'une notification est envoyée au créateur

**Résultat attendu:**

```
{
  "success": true,
  "message": "Paiement approuvé et transfert Stripe déclenché",
  "payout": {
    "id": "payout_xxx",
    "status": "processing",
    "stripePayoutId": "po_xxx",
    "amountEur": 50.00
  }
}
```

**Vérifications base de données:**

```
SELECT * FROM "Payout" WHERE status = 'PROCESSING' AND "approvedAt" IS NOT NULL;
SELECT * FROM "PayoutAuditLog" WHERE status = 'PROCESSING' AND "payoutId" IS NOT NULL;
SELECT * FROM "Notification" WHERE type = 'PAYOUT_APPROVED';
```

---

## 5. Test de rejet de payout

**Endpoint:** `POST /api/admin/payouts/[id]/reject`
**Utilisateur:** Admin authentifié

**Étapes:**
1. Se connecter en tant qu'admin
2. Sélectionner un payout avec status `REQUESTED`
3. Envoyer la demande de rejet avec une raison
4. Vérifier que le statut passe à `REJECTED`
5. Vérifier que `rejectedAt` est défini
6. Vérifier que `rejectionReason` contient la raison
7. Vérifier que `approvedById` contient l'ID de l'admin (qui a rejeté)
8. Vérifier qu'un PayoutAuditLog est créé avec status `REJECTED` et `payoutId`
9. Vérifier qu'une notification est envoyée au créateur

**Résultat attendu:**

```
{
  "success": true,
  "message": "Paiement rejeté",
  "payout": {
    "id": "payout_xxx",
    "status": "rejected",
    "rejectionReason": "Raison du rejet"
  }
}
```

**Vérifications base de données:**

```
SELECT * FROM "Payout" WHERE status = 'REJECTED' AND "rejectedAt" IS NOT NULL;
SELECT * FROM "PayoutAuditLog" WHERE status = 'REJECTED' AND "payoutId" IS NOT NULL;
SELECT * FROM "Notification" WHERE type = 'SYSTEM' AND message LIKE '%rejeté%';
```

---

## 6. Test webhook payout.paid

**Endpoint:** `POST /api/payments/webhook`
**Type:** Webhook Stripe `payout.paid`

**Étapes:**
1. Déclencher un payout depuis Stripe (ou simuler un webhook)
2. Vérifier que le payout existe avec `stripePayoutId`
3. Vérifier que le statut passe à `PAID`
4. Vérifier que `paidAt` est défini
5. Vérifier qu'un PayoutAuditLog est créé avec action `COMPLETED` et `payoutId`
6. Vérifier qu'une notification `PAYOUT_COMPLETED` est envoyée

**Webhook payload (simulé):**

```
{
  "type": "payout.paid",
  "data": {
    "object": {
      "id": "po_xxx",
      "amount": 5000,
      "currency": "eur",
      "status": "paid",
      "arrival_date": 1735344000
    }
  }
}
```

**Vérifications base de données:**

```
SELECT * FROM "Payout" WHERE "stripePayoutId" = 'po_xxx' AND status = 'PAID' AND
"paidAt" IS NOT NULL;
SELECT * FROM "PayoutAuditLog" WHERE action = 'COMPLETED' AND "payoutId" IS NOT NULL;
SELECT * FROM "Notification" WHERE type = 'PAYOUT_COMPLETED';
```

---

## 7. Test webhook payout.failed

**Endpoint:** `POST /api/payments/webhook`
**Type:** Webhook Stripe `payout.failed`

**Étapes:**
1. Simuler un payout échoué depuis Stripe
2. Vérifier que le statut passe à `FAILED`
3. Vérifier que `failedAt` est défini
4. Vérifier que `failureReason` contient le message d'erreur
5. Vérifier qu'un PayoutAuditLog est créé avec action `FAILED` et `payoutId`
6. Vérifier qu'une notification d'erreur est envoyée

**Webhook payload (simulé):**

```
{
  "type": "payout.failed",
  "data": {
    "object": {
      "id": "po_xxx",
      "amount": 5000,
      "currency": "eur",
      "status": "failed",
      "failure_message": "Insufficient funds",
      "failure_code": "insufficient_funds"
    }
  }
}
```

**Vérifications base de données:**

```
SELECT * FROM "Payout" WHERE "stripePayoutId" = 'po_xxx' AND status = 'FAILED' AND
"failedAt" IS NOT NULL;
SELECT * FROM "PayoutAuditLog" WHERE action = 'FAILED' AND "payoutId" IS NOT NULL;
SELECT * FROM "Notification" WHERE type = 'SYSTEM' AND message LIKE '%échoué%';
```

---

## 8. Test UI Admin - Affichage des payouts

**Page:** `/dashboard/admin/payouts`

**Étapes:**
1. Se connecter en tant qu'admin
2. Vérifier que la liste affiche tous les payouts
3. Tester le filtre par statut `REQUESTED`
4. Vérifier que les boutons "Approuver" et "Rejeter" apparaissent uniquement pour status `REQUESTED`
5. Cliquer sur "Approuver" → vérifier le modal de confirmation
6. Confirmer l'approbation → vérifier le toast de succès
7. Vérifier que le payout disparaît de la liste `REQUESTED`
8. Tester le rejet avec raison → vérifier le modal avec textarea
9. Vérifier que le payout est marqué `REJECTED`

---

### 9. Test notifications admin

**Scénario:** Nouveau payout créé par créateur

**Étapes:**
1. Créateur crée une demande de payout
2. Vérifier que tous les admins reçoivent une notification in-app
3. Vérifier que tous les admins reçoivent un email
4. Vérifier le contenu de l'email (montant, créateur, lien vers dashboard)
5. Vérifier le compteur de notifications non lues dans le NotificationBell

---

### 10. Test intégration complète

**Workflow complet:** Demande → Approbation → Paiement

**Étapes:**
1. **Créateur:** Créer une demande de payout
- Vérifier status = `REQUESTED`
- Vérifier notification admin
2. **Admin:** Voir la demande dans le dashboard
- Vérifier que le payout apparaît dans la liste
- Vérifier que les boutons "Approuver" et "Rejeter" sont visibles
3. **Admin:** Approuver la demande
- Vérifier status = `APPROVED` puis `PROCESSING`
- Vérifier notification créateur
4. **Stripe:** Déclencher le payout
- Attendre le webhook `payout.paid`
5. **Webhook:** Vérifier status = `PAID`
- Vérifier `paidAt` défini
- Vérifier notification créateur "Paiement effectué"
6. **Créateur:** Voir le payout dans son dashboard
- Vérifier qu'il apparaît dans la liste avec status `PAID`
- Vérifier le summary avec `totalPaid` mis à jour

---

## ✅ Checklist de validation

- [ ] Migration appliquée sans erreur
- [ ] Client Prisma généré
- [ ] Enum PayoutStatus simplifié (7 statuts)
- [ ] Modèle Payout avec tous les champs (requestedAt, approvedAt, paidAt, failedAt, rejectedAt, metadata)
- [ ] PayoutAuditLog lié au Payout (payoutId)
- [ ] API /api/payouts/request crée un Payout avec status REQUESTED
- [ ] API /api/payouts/creator retourne les Payouts du créateur
- [ ] API /api/admin/payouts retourne tous les Payouts avec filtres
- [ ] API /api/admin/payouts/[id]/approve vérifie REQUESTED et crée stripePayoutId
- [ ] API /api/admin/payouts/[id]/reject enregistre rejectedAt et rejectionReason

- [ ] Webhooks payout.paid met à jour avec paidAt
- [ ] Webhooks payout.failed met à jour avec failedAt
- [ ] UI admin affiche les payouts avec nouveaux statuts
- [ ] UI admin affiche les boutons Approuver/Rejeter uniquement pour REQUESTED
- [ ] Notifications admin lors de nouvelle demande
- [ ] Notifications créateur lors d'approbation/rejet/paiement
- [ ] Tous les PayoutAuditLog incluent payoutId

---

## 🚀 Comment tester

### 1. Test manuel via l'application

```
# 1. Démarrer l'application
npm run dev

# 2. Créer un compte créateur et configurer Stripe Connect
# 3. Faire une réservation pour avoir un solde disponible
# 4. Demander un payout en tant que créateur
# 5. Se connecter en tant qu'admin
# 6. Approuver ou rejeter le payout
# 7. Vérifier les notifications et statuts
```

### 2. Test via Prisma Studio

```
npx prisma studio
```

Ouvrir dans le navigateur et vérifier:
- Modèle `Payout` avec les nouveaux champs
- Modèle `PayoutAuditLog` avec `payoutId`
- Statuts correctement enregistrés

### 3. Test via curl/Postman

```
# Créer un payout
curl -X POST http://localhost:3000/api/payouts/request \
  -H "Cookie: auth-token=YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"amount": 50}'

# Lister les payouts (créateur)
curl http://localhost:3000/api/payouts/creator \
  -H "Cookie: auth-token=YOUR_CREATOR_TOKEN"

# Lister les payouts (admin)
curl http://localhost:3000/api/admin/payouts \
  -H "Cookie: auth-token=YOUR_ADMIN_TOKEN"

# Approuver un payout
curl -X POST http://localhost:3000/api/admin/payouts/PAYOUT_ID/approve \
  -H "Cookie: auth-token=YOUR_ADMIN_TOKEN"

# Rejeter un payout
curl -X POST http://localhost:3000/api/admin/payouts/PAYOUT_ID/reject \
  -H "Cookie: auth-token=YOUR_ADMIN_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"reason": "Raison du rejet"}'
```

---

## 📊 Résultats attendus

### Base de données

- Tous les Payouts ont un statut valide (REQUESTED, APPROVED, PROCESSING, PAID, FAILED, REJECTED, CANCELED)
- Tous les PayoutAuditLog ont un `payoutId` non null
- Les dates (requestedAt, approvedAt, paidAt, failedAt, rejectedAt) sont correctement renseignées

### Notifications

- Admins reçoivent une notification lors de nouvelle demande
- Créateurs reçoivent une notification lors d’approbation
- Créateurs reçoivent une notification lors de rejet
- Créateurs reçoivent une notification lors de paiement réussi
- Créateurs reçoivent une notification lors d’échec

### UI

- Dashboard admin affiche tous les payouts
- Filtres fonctionnent correctement
- Boutons Approuver/Rejeter visibles uniquement pour REQUESTED
- Modals de confirmation fonctionnent
- Toasts de succès/erreur s’affichent

---

## 🐛 Problèmes potentiels

1. **Migration échouée**
   - Solution: Vérifier qu'il n'y a pas de données avec anciens statuts
   - Rollback: `npx prisma migrate resolve --rolled-back 20251227112756_phase3_payout_entity_refactor`

2. **Statuts non reconnus**
   - Solution: Régénérer le client Prisma: `npx prisma generate`

3. **PayoutAuditLog sans payoutId**
   - Solution: Vérifier que tous les appels à `prisma.payoutAuditLog.create` incluent `payoutId`

4. **Webhooks ne fonctionnent pas**
   - Solution: Vérifier que Stripe CLI est en cours d'exécution
   - Solution: Vérifier la signature du webhook

5. **Notifications non envoyées**
   - Solution: Vérifier les logs de la console
   - Solution: Vérifier que les admins existent dans la base de données