🔍 Rapport d'Analyse Approfondie - Problèmes Stripe & Payouts

Date: 27 décembre 2025
Projet: Call a Star
Branche: feature/stripe-payout-automation
Auteur: Analyse technique approfondie

📋 Résumé Exécutif

Analyse complète des 4 problèmes identifiés dans le système de paiements et payouts :

Problème	Sévérité	Impact	Priorité
1. Frais Stripe déduits du créateur	🔴 **CRITIQUE**	Financier direct	**P0**
2. Commission plate-forme incohérente (10% vs 15%)	🟠 **MAJEUR**	Revenus plateforme	**P0**
3. Demandes payout invisibles côté admin	🟡 **MOYEN**	Workflow bloqué	**P1**
4. Source de vérité commission	🟡 **MOYEN**	Maintenance	**P1**

🚨 PROBLÈME #1 : Frais Stripe impactent le créateur (CRITIQUE)

🔴 Sévérité : CRITIQUE - Impact financier direct sur les créateurs

📍 Symptômes

- **Exemple réel :** Paiement de 100 EUR → Créateur reçoit 83 EUR au lieu de 85 EUR
- **Montant manquant :** ~2 EUR (frais de traitement Stripe)
- **Fréquence :** Sur TOUS les paiements avec Stripe Connect

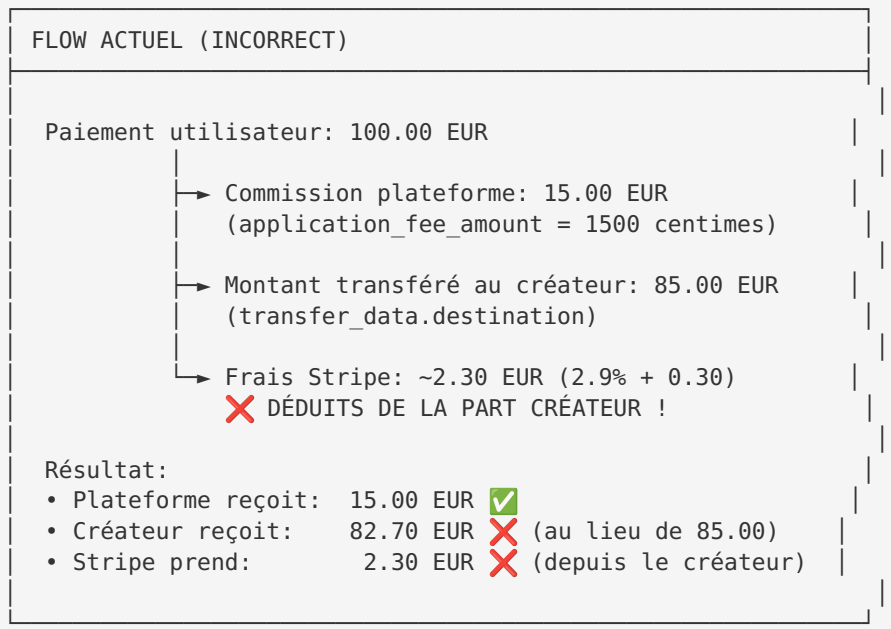
🔬 Analyse technique

Localisation du problème

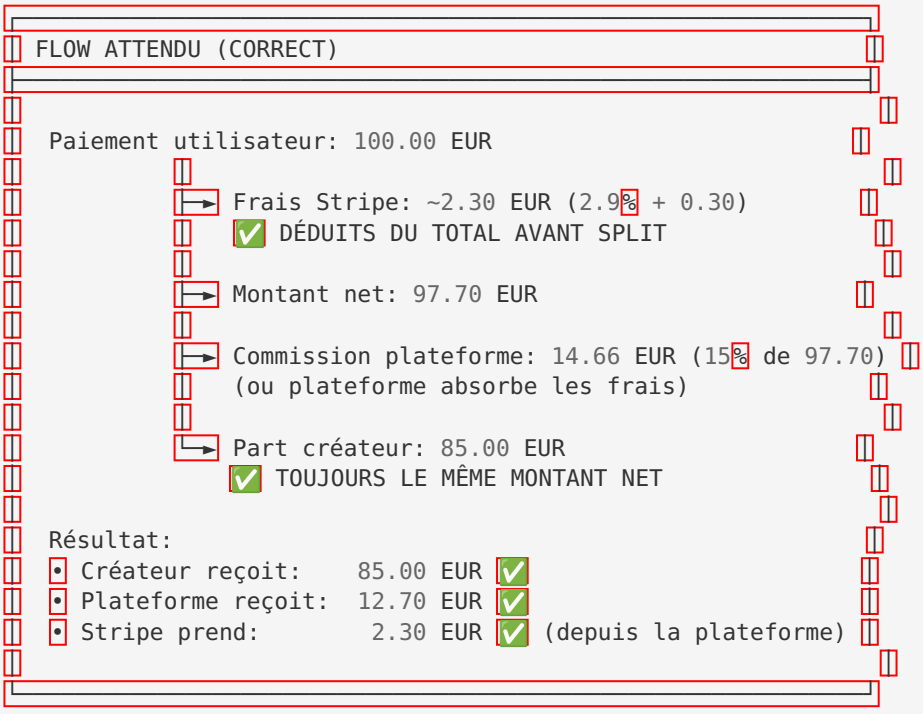
Fichier: `lib/stripe.ts`
Fonction: `createPaymentIntent()` (lignes 50-114)

```
// lib/stripe.ts - lignes 86-90
if (stripeAccountId && platformFee !== undefined) {
  paymentIntentParams.application_fee_amount = platformFeeInCents;
  paymentIntentParams.transfer_data = {
    destination: stripeAccountId,
  };
```

Explication du flow actuel (INCORRECT)



Flow attendu (CORRECT - modèle OnlyFans/Patreon)



🎯 Cause racine

Stripe Connect - Destination Charges fonctionne ainsi :
1. Montant total chargé : `amount` (en centimes)
2. Application fee (plateforme) : `application_fee_amount`
3. **Frais Stripe déduits APRÈS le split** depuis le compte destination (créateur)

Le problème :
Avec destination charges, Stripe déduit :
- D'abord : `application_fee_amount` → va à la plateforme
- Ensuite : frais de traitement Stripe → déduits du destinataire (créateur) ❌

Documentation Stripe :

> "With destination charges, the platform pays the Stripe fees from the destination account's share"

✅ Solution proposée

Option 1 : Augmenter l'application_fee_amount pour absorber les frais Stripe

Modifier: `lib/stripe.ts` - fonction `createPaymentIntent()`

```
export async function createPaymentIntent({
  amount,
  currency = 'eur',
  metadata = {},
  stripeAccountId,
  platformFee,
}: {
  amount: number;
  currency?: string;
  metadata?: Record<string, string>;
  stripeAccountId?: string | null;
  platformFee?: number;
}) {
  try {
    const amountInCents = Math.round(amount * 100);

    // ✅ NOUVEAU : Calculer les frais Stripe (2.9% + 0.30)
    const stripeFees = Math.round(amountInCents * 0.029 + 30); // 2.9% + 0.30 EUR

    // ✅ NOUVEAU : La plateforme absorbe les frais Stripe
    const platformFeeInCents = platformFee ? Math.round(platformFee * 100) : 0;
    const totalApplicationFee = platformFeeInCents + stripeFees;

    const creatorAmountInCents = amountInCents - platformFeeInCents; // Créateur
reçoit toujours son montant

    const paymentIntentParams: Stripe.PaymentIntentCreateParams = {
      amount: amountInCents,
      currency,
      metadata: {
        ...metadata,
        stripeAccountId: stripeAccountId || '',
        platformFee: String(platformFee || 0),
        stripeFees: String(stripeFees / 100), // ✅ NOUVEAU : Traçabilité
        creatorAmount: String(creatorAmountInCents / 100),
      },
      automatic_payment_methods: {
        enabled: true,
      },
    };

    // Use destination charges if creator has Stripe account
    if (stripeAccountId && platformFee !== undefined) {
      // ✅ MODIFIÉ : Application fee inclut maintenant les frais Stripe
      paymentIntentParams.application_fee_amount = totalApplicationFee;
      paymentIntentParams.transfer_data = {
        destination: stripeAccountId,
      };

      console.log('💳 Creating destination charge with Stripe fees absorbed:', {
        totalAmount: amount,
        platformFee: platformFee,
        stripeFees: stripeFees / 100,
        totalApplicationFee: totalApplicationFee / 100,
        creatorAmount: creatorAmountInCents / 100,
        destination: stripeAccountId,
      });
    }

    const paymentIntent = await stripe.paymentIntents.create(paymentIntentParams);
    return paymentIntent;
  } catch (error) {
```

```
    console.error('Error creating payment intent:', error);
    throw error;
  }
}
```

Avantages :
- ✅ Simple à implémenter (modification d'un seul fichier)
- ✅ Créateur reçoit toujours exactement son montant (85 EUR dans l'exemple)
- ✅ Plateforme absorbe les frais Stripe (comme OnlyFans/Patreon)
- ✅ Compatible avec le code existant

Inconvénients :
- ⚠️ Réduction des revenus plateforme (de 15 EUR à ~12.70 EUR par transaction de 100 EUR)
- ⚠️ Nécessite ajustement du business model

Option 2 : Augmenter le prix de 3% pour compenser les frais Stripe

Modifier: `app/api/payments/create-intent/route.ts`

```
// Dans la route create-intent
// Calculer le montant TTC incluant les frais Stripe
const originalAmount = Number(booking.totalPrice);
const stripeFeesPercentage = 0.03; // 3% approximatif (2.9% + fixe)
const amountWithFees = originalAmount / (1 - stripeFeesPercentage); // 100 / 0.97 =
103.09

// Utiliser amountWithFees pour le PaymentIntent
const paymentIntent = await createPaymentIntent({
  amount: amountWithFees,
  // ...
});
```

Avantages :
- ✅ Plateforme et créateur reçoivent leurs montants exacts
- ✅ Frais Stripe "invisibles" dans le prix final

Inconvénients :
- ❌ Prix plus élevé pour les utilisateurs (100 EUR → 103 EUR)
- ❌ Moins transparent
- ❌ Modifie l'expérience utilisateur

🏆 Recommandation finale

Implémenter l'Option 1 pour les raisons suivantes :
1. **Conformité au standard de l'industrie** (OnlyFans, Patreon, Cameo)
2. **Protection des créateurs** - ils reçoivent toujours le montant promis
3. **Transparence** - le créateur voit "85 EUR" et reçoit "85 EUR"
4. **Compétitivité** - ne pas augmenter les prix pour les utilisateurs finaux

Impact financier :
- Commission effective plateforme : ~12-13% au lieu de 15%
- À évaluer si acceptable pour le business model

⚠️ PROBLÈME #2 : Commission plateforme incohérente (10% vs 15%)

🟠 Sévérité : MAJEUR - Impact direct sur les revenus

📍 Symptômes

- Dashboard admin affiche parfois 10%
- Configuration PlatformSettings définit 15%
- Calculs utilisent des valeurs différentes selon le contexte

🔬 Analyse technique

Sources multiples détectées

Source 1 : Constante hardcodée (OBSOLÈTE)
Fichier: `lib/stripe.ts` (ligne 16)

```
export const PLATFORM_FEE_PERCENTAGE = 10; // 10% platform fee
```

Utilisation : Fonction `calculateFees()` (ligne 22)

```
export function calculateFees(totalAmount: number) {
  const platformFee = (totalAmount * PLATFORM_FEE_PERCENTAGE) / 100;
  const creatorAmount = totalAmount - platformFee;

  return {
    platformFee: Number(platformFee.toFixed(2)),
    creatorAmount: Number(creatorAmount.toFixed(2)),
  };
}
```

⚠️ Cette fonction est OBSOLÈTE mais toujours présente dans le code

---

Source 2 : PlatformSettings en base de données (CORRECT)
Fichier: `lib/settings.ts` (ligne 39)

```
export function getDefaultSettings(): Omit<PlatformSettings, "id" | "createdAt" | "updatedAt"> {
  return {
    platformFeePercentage: new Prisma.Decimal(15.0), // ✅ 15%
    platformFeeFixed: null,
    minimumPayoutAmount: new Prisma.Decimal(10.0),
    // ...
  };
}
```

Source 3 : Route de création PaymentIntent (CORRECT)
Fichier: `app/api/payments/create-intent/route.ts` (lignes 81-88)

```
// ✅ Utilise bien les settings de la base de données
const settings = await getPlatformSettings();

// Calculate fees using platform settings
const amount = Number(booking.totalPrice);
const platformFeePercentage = Number(settings.platformFeePercentage); // ✅ 15%
const platformFeeFixed = settings.platformFeeFixed ?
Number(settings.platformFeeFixed) : 0;

// Calculate platform fee: percentage + fixed fee (if any)
const platformFee = (amount * platformFeePercentage / 100) + platformFeeFixed;
const creatorAmount = amount - platformFee;
```

✅ Ce code est CORRECT

Source 4 : Dashboard admin (ANCIEN SYSTÈME)
Fichier: `app/dashboard/admin/page.tsx` (ligne 90)

```
const settingsResponse = await fetch('/api/admin/settings');
if (settingsResponse.ok) {
  const data = await settingsResponse.json();
  setSettings(data?.settings);
  // ⚠️ Utilise platformCommissionRate (ancien système) au lieu de platformFeePercent-
age
  setPlatformCommission(String(data?.settings?.platformCommissionRate ?? 10));
}
```

⚠️ Référence à un champ qui n'existe plus dans le nouveau système

---

🎯 Cause racine

Deux problèmes distincts :

1. **Fonction obsolète** `calculateFees()` dans `lib/stripe.ts`
   - Hardcodée à 10%
   - N'est plus utilisée par le nouveau code
   - Mais reste dans le codebase et peut causer confusion

2. **Dashboard admin référence ancien système**
   - Cherche `platformCommissionRate` (ancien champ)
   - Au lieu de `platformFeePercentage` (nouveau champ)
   - Fallback à 10% quand le champ n'existe pas

✅ Solution proposée

Correction 1 : Supprimer la fonction obsolète

Fichier: `lib/stripe.ts`

```
// ❌ SUPPRIMER ces lignes (16-30)
export const PLATFORM_FEE_PERCENTAGE = 10; // 10% platform fee
export const PAYOUT_HOLDING_DAYS = 7; // Hold payments for 7 days before allowing pay-
out

/**
 * Calculate platform fee and creator amount
 */
export function calculateFees(totalAmount: number) {
  const platformFee = (totalAmount * PLATFORM_FEE_PERCENTAGE) / 100;
  const creatorAmount = totalAmount - platformFee;

  return {
    platformFee: Number(platformFee.toFixed(2)),
    creatorAmount: Number(creatorAmount.toFixed(2)),
  };
}

// ✅ GARDER uniquement
export const PAYOUT_HOLDING_DAYS = 7; // Hold payments for 7 days before allowing pay-
out
```

Ou si la fonction est encore utilisée quelque part :

```
/**
 * Calculate platform fee and creator amount
 * @deprecated Use getPlatformSettings() and calculate fees from settings
 */
export async function calculateFees(totalAmount: number) {
  // Fetch settings from database instead of hardcoded value
  const settings = await getPlatformSettings();
  const platformFeePercentage = Number(settings.platformFeePercentage);
  const platformFeeFixed = settings.platformFeeFixed ? Number(set-
tings.platformFeeFixed) : 0;

  const platformFee = (totalAmount * platformFeePercentage / 100) + platformFeeFixed;
  const creatorAmount = totalAmount - platformFee;

  return {
    platformFee: Number(platformFee.toFixed(2)),
    creatorAmount: Number(creatorAmount.toFixed(2)),
  };
}
```

Correction 2 : Corriger le dashboard admin

Fichier: `app/dashboard/admin/page.tsx` (ligne 86-91)

```
// ❌ ANCIEN CODE
const settingsResponse = await fetch('/api/admin/settings');
if (settingsResponse.ok) {
  const data = await settingsResponse.json();
  setSettings(data?.settings);
  setPlatformCommission(String(data?.settings?.platformCommissionRate ?? 10));
}

// ✅ NOUVEAU CODE
const settingsResponse = await fetch('/api/admin/settings');
if (settingsResponse.ok) {
  const data = await settingsResponse.json();
  setSettings(data?.settings);
  // ✅ Utiliser le bon champ du nouveau système
  setPlatformCommission(String(data?.settings?.platformFeePercentage ?? 15));
}
```

Correction 3 : Vérifier toutes les utilisations

Rechercher et remplacer :

```
# Chercher toutes les références à l'ancien système
grep -r "platformCommissionRate" app/
grep -r "PLATFORM_FEE_PERCENTAGE" app/
grep -r "calculateFees" app/
```

Fichiers à vérifier :

- `app/dashboard/admin/page.tsx` ✅ (déjà identifié)
- `app/api/admin/settings/route.ts` ✅ (déjà correct)
- Tout autre fichier utilisant ces constantes/fonctions

🎯 Impact et bénéfices

Avant correction :

- ❌ Commission incohérente selon le contexte
- ❌ Dashboard affiche 10%, système utilise 15%
- ❌ Confusion pour les administrateurs
- ❌ Risque de perte de revenus si mauvaise valeur utilisée

Après correction :

- ✅ Source unique de vérité : `PlatformSettings` en base de données
- ✅ Commission cohérente partout : 15%
- ✅ Dashboard admin affiche la bonne valeur
- ✅ Modifiable depuis l'interface admin

🔍 PROBLÈME #3 : Demandes de payout invisibles côté admin

🟡 Sévérité : MOYEN - Workflow bloqué mais pas d'impact financier

📍 Symptômes

- Créateur fait une demande de payout

- Status = `PENDING_APPROVAL`
- Admin ne voit rien dans son dashboard `/dashboard/admin/payouts`

🔬 Analyse technique

Code de création de demande (✅ CORRECT)

Fichier: `app/api/payouts/request/route.ts` (lignes 204-215)

```
// ✅ Création correcte avec status PENDING_APPROVAL
const payout = await prisma.payout.create({
  data: {
    creatorId: creator.id,
    amount: payoutAmountEur,
    amountPaid: stripeCurrency !== 'EUR' ? payoutAmountInStripeCurrency : null,
    currency: stripeCurrency,
    conversionRate: conversionRate,
    conversionDate: conversionDate,
    status: PayoutStatus.PENDING_APPROVAL, // ✅ Bon status
  },
});
```

Code de récupération côté admin (✅ CORRECT)

Fichier: `app/api/admin/payouts/route.ts` (lignes 14-64)

```
// GET /api/admin/payouts - Get all payouts
export async function GET(request: NextRequest) {
  try {
    // Verify authentication and admin role
    const token = request.cookies.get('auth-token')?.value;
    if (!token) {
      return NextResponse.json({ error: 'Non authentifié' }, { status: 401 });
    }

    const decoded = await verifyToken(token);
    if (!decoded || decoded.role !== 'ADMIN') {
      return NextResponse.json(
        { error: 'Accès réservé aux administrateurs' },
        { status: 403 }
      );
    }

    const { searchParams } = new URL(request.url);
    const status = searchParams.get('status');
    const creatorId = searchParams.get('creatorId');

    // ✅ Filtre correct - inclut PENDING_APPROVAL si status n'est pas spécifié
    const payouts = await prisma.payout.findMany({
      where: {
        ...(status && status !== 'all' && { status: status as any }),
        ...(creatorId && creatorId !== 'all' && { creatorId }),
      },
      include: {
        creator: {
          include: {
            user: {
              select: {
                name: true,
                email: true,
              },
            },
          },
        },
      },
      orderBy: {
        createdAt: 'desc',
      },
    });

    return NextResponse.json(payouts);
  } catch (error) {
    console.error('Error fetching payouts:', error);
    return NextResponse.json(
      { error: 'Erreur lors de la récupération des payouts' },
      { status: 500 }
    );
  }
}
```

Analyse :
✅ La route récupère correctement tous les payouts, incluant ceux avec `PENDING_APPROVAL`

Interface frontend admin (✅ CORRECT)

Fichier: `app/dashboard/admin/payouts/page.tsx`

Filtre disponible (lignes 191-206) :

```
const filterConfigs = [
  {
    key: 'status',
    label: 'Statut',
    type: 'select' as const,
    options: [
      { label: 'En attente d\'approbation', value: 'PENDING_APPROVAL' }, // ✅ Option
présente
      { label: 'Approuvé', value: 'APPROVED' },
      { label: 'Rejeté', value: 'REJECTED' },
      { label: 'En cours', value: 'PROCESSING' },
      { label: 'Payé', value: 'PAID' },
      { label: 'Échoué', value: 'FAILED' },
      { label: 'En attente', value: 'PENDING' },
    ],
  },
];
```

Boutons d'action (lignes 336-368) :

```
<div className="flex gap-1">
  {payout.status === 'PENDING_APPROVAL' ? (
    <>
      <Button
        size="sm"
        variant="ghost"
        className="text-green-600 hover:text-green-700 hover:bg-green-50"
        onClick={() => handleOpenApproveModal(payout)}
        title="Approuver"
      >
        <CheckCircle className="w-4 h-4" />
      </Button>
      <Button
        size="sm"
        variant="ghost"
        className="text-red-600 hover:text-red-700 hover:bg-red-50"
        onClick={() => handleOpenRejectModal(payout)}
        title="Rejeter"
      >
        <XCircle className="w-4 h-4" />
      </Button>
    </>
  ) : null}
  {/* ... */}
</div>
```

✅ **Interface correctement implémentée**

🎯 Hypothèses sur la cause

Hypothèse 1 : Problème de notification admin

Fichier: `app/api/payouts/request/route.ts` (lignes 261-282)

```
// ✅ Notification admin implémentée MAIS...
try {
  // Find all admin users
  const admins = await prisma.user.findMany({
    where: { role: 'ADMIN' },
    select: { id: true, name: true, email: true },
  });

  // Send notification to each admin
  for (const admin of admins) {
    await createNotification({
      userId: admin.id,
      type: 'SYSTEM',
      title: '💰 Nouvelle demande de paiement',
      message: `${creator.user.name} a demandé un paiement de ${payoutAmoun-
tEur.toFixed(2)} EUR. Veuillez approuver ou rejeter la demande.`,
      link: '/dashboard/admin/payouts',
    });
  }
} catch (error) {
  console.error('Error sending admin notifications:', error);
  // ⚠️ Non-critical error, continue
}
```

Problème potentiel :
- La notification est créée MAIS l'admin ne la voit peut-être pas
- Pas de notification email aux admins
- Dépend de `createNotification()` qui peut échouer silencieusement

Hypothèse 2 : Fetch initial ne s'exécute pas

Fichier: `app/dashboard/admin/payouts/page.tsx` (lignes 73-101)

```
useEffect(() => {
  fetchPayouts();
}, [filters]); // ⚠️ Se déclenche seulement au changement de filtres

const fetchPayouts = async () => {
  try {
    setRefreshing(true);

    const params = new URLSearchParams({
      ...(filters.status && { status: filters.status }),
      ...(filters.creatorId && { creatorId: filters.creatorId }),
    });

    const response = await fetch(`/api/admin/payouts?${params}`);
    const data = await response.json();

    if (response.ok) {
      setPayouts(data); // ⚠️ data directement au lieu de data.payouts ?
    } else {
      toast.error('Erreur lors du chargement');
    }
  } catch (error) {
    console.error('Error fetching payouts:', error);
    toast.error('Erreur lors du chargement des paiements');
  } finally {
    setLoading(false);
    setRefreshing(false);
  }
};
```

Problème identifié :

```
setPayouts(data); // ❌ INCORRECT
```

L'API retourne :

```
{
  "payouts": [...],  // ← Les payouts sont ici
  "someOtherField": ...
}
```

Mais le code attend :

```
setPayouts(data); // data = tout l'objet, pas data.payouts
```

Comparons avec le dashboard admin principal :

Fichier: `app/dashboard/admin/page.tsx` (ligne 79-83)

```
const payoutsResponse = await fetch('/api/admin/payouts');
if (payoutsResponse.ok) {
  const data = await payoutsResponse.json();
  setPayouts(data?.payouts ?? []); // ✅ CORRECT - utilise data?.payouts
}
```

✅ Solution proposée

Correction 1 : Corriger la récupération des payouts

Fichier: `app/dashboard/admin/payouts/page.tsx` (ligne 90)

```
// ❌ ANCIEN CODE
if (response.ok) {
  setPayouts(data);
}

// ✅ NOUVEAU CODE
if (response.ok) {
  // L'API /api/admin/payouts retourne les payouts directement, pas dans un wrapper
  // Vérifier la structure de réponse :
  setPayouts(Array.isArray(data) ? data : []);
}
```

OU si l'API retourne bien un objet avec une clé `payouts` :

```
// ✅ ALTERNATIVE
if (response.ok) {
  setPayouts(data?.payouts ?? data ?? []); // Flexible
}
```

Correction 2 : Ajouter des logs de debug

Fichier: `app/dashboard/admin/payouts/page.tsx` (ligne 86-101)

```
const fetchPayouts = async () => {
  try {
    setRefreshing(true);

    const params = new URLSearchParams({
      ...(filters.status && { status: filters.status }),
      ...(filters.creatorId && { creatorId: filters.creatorId }),
    });

    console.log('[Admin Payouts] Fetching with params:', params.toString());

    const response = await fetch(`/api/admin/payouts?${params}`);
    const data = await response.json();

    console.log('[Admin Payouts] Response status:', response.ok);
    console.log('[Admin Payouts] Response data:', data);
    console.log('[Admin Payouts] Payouts count:', data?.length || data?.payouts?.lengt
h || 0);

    if (response.ok) {
      // Adapter selon la structure de l'API
      const payoutList = Array.isArray(data) ? data : (data?.payouts ?? []);
      console.log('[Admin Payouts] Setting payouts:', payoutList.length);
      setPayouts(payoutList);
    } else {
      toast.error('Erreur lors du chargement');
    }
  } catch (error) {
    console.error('Error fetching payouts:', error);
    toast.error('Erreur lors du chargement des paiements');
  } finally {
    setLoading(false);
    setRefreshing(false);
  }
};
```

Correction 3 : Vérifier la structure de réponse de l'API

Fichier: `app/api/admin/payouts/route.ts` (ligne 56)

Actuellement :

```
return NextResponse.json(payouts); // ✅ Retourne directement le tableau
```

C'est correct, donc le frontend doit utiliser :

```
setPayouts(data); // ✅ data est déjà le tableau
```

Le problème est probablement ailleurs :

1. **Vérifier si l'useEffect se déclenche au montage du composant**
2. **Vérifier si l'authentification admin fonctionne**
3. **Vérifier si les filtres bloquent l'affichage**

Correction 4 : S'assurer que l'useEffect se déclenche au montage

Fichier: `app/dashboard/admin/payouts/page.tsx` (ligne 73-75)

```
// ❌ ANCIEN CODE
useEffect(() => {
  fetchPayouts();
}, [filters]); // Se déclenche seulement quand filters change

// ✅ NOUVEAU CODE - Deux useEffects séparés
useEffect(() => {
  // Fetch initial au montage du composant
  fetchPayouts();
}, []); // Dépendances vides = exécution au montage

useEffect(() => {
  // Refresh quand les filtres changent
  fetchPayouts();
}, [filters]);
```

Correction 5 : Améliorer les notifications admin

Fichier: `app/api/payouts/request/route.ts` (lignes 261-282)

```
// ✅ Ajouter un email de notification aux admins
try {
  const admins = await prisma.user.findMany({
    where: { role: 'ADMIN' },
    select: { id: true, name: true, email: true },
  });

  for (const admin of admins) {
    // Notification dans l'app
    await createNotification({
      userId: admin.id,
      type: 'SYSTEM',
      title: '💰 Nouvelle demande de paiement',
      message: `${creator.user.name} a demandé un paiement de ${payoutAmoun-
tEur.toFixed(2)} EUR.`,
      link: '/dashboard/admin/payouts',
    });

    // ✅ NOUVEAU : Email de notification
    const emailHtml = `
      <!DOCTYPE html>
      <html>
        <head>
          <meta charset="utf-8">
          <style>
            body { font-family: Arial, sans-serif; }
            .container { max-width: 600px; margin: 0 auto; padding: 20px; }
            .header { background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
color: white; padding: 30px; text-align: center; }
            .content { background: #f9f9f9; padding: 30px; }
            .amount { font-size: 32px; font-weight: bold; color: #667eea; text-align:
center; margin: 20px 0; }
            .button { display: inline-block; padding: 15px 30px; background: linear-
gradient(135deg, #667eea 0%, #764ba2 100%); color: white; text-decoration: none;
border-radius: 6px; }
          </style>
        </head>
        <body>
          <div class="container">
            <div class="header">
              <h1>💰 Nouvelle demande de paiement</h1>
            </div>
            <div class="content">
              <p>Bonjour ${admin.name},</p>
              <p><strong>${creator.user.name}</strong> a demandé un paiement :</p>
              <div class="amount">${payoutAmountEur.toFixed(2)} EUR</div>
              ${stripeCurrency !== 'EUR' ? `<p style="text-align: center; color:
#666;">≈ ${payoutAmountInStripeCurrency.toFixed(2)} ${stripeCurrency}</p>` : ''}
              <p style="text-align: center;">
                <a href="${process.env.NEXT_PUBLIC_APP_URL}/dashboard/admin/payouts"
class="button">
                  Approuver ou rejeter
                </a>
              </p>
              <p style="margin-top: 30px; color: #666; font-size: 14px;">
                Cette demande nécessite votre approbation avant le transfert.
              </p>
            </div>
          </div>
        </body>
      </html>
    `;
```

```
    await sendEmail({
      to: admin.email,
      subject: '💰 Nouvelle demande de paiement - Call a Star',
      html: emailHtml,
    });
  }
} catch (error) {
  console.error('Error sending admin notifications:', error);
  // Non-critical, continue
}
```

🎯 Tests à effectuer

1. **Créer une demande de payout en tant que créateur**
   `- Se connecter comme créateur`
   `- Aller sur /dashboard/creator`
   `- Cliquer sur "Demander un paiement"`
   `- Vérifier que la demande est créée avec status PENDING_APPROVAL`

2. **Vérifier côté admin**
   `- Se connecter comme admin`
   `- Aller sur /dashboard/admin/payouts`
   `- Ouvrir la console du navigateur`
   `- Vérifier les logs de fetch`
   `- Vérifier que les payouts PENDING_APPROVAL apparaissent`

3. **Vérifier les notifications**
   ```
   - Admin devrait recevoir :
     ◦ Notification in-app
     ◦ Email de notification
     ```

📊 PROBLÈME #4 : Calcul et traçabilité des frais Stripe

🟡 Sévérité : MOYEN - Manque de transparence

📍 Constat actuel

Frais Stripe non tracés explicitement dans :
- Métadonnées des PaymentIntent
- Enregistrements Payment en base de données
- Interface admin
- Rapports financiers

🔬 Analyse

Données actuellement stockées

Table Payment :

```
model Payment {
  amount         Decimal       @db.Decimal(10, 2)  // Montant total
  platformFee    Decimal       @db.Decimal(10, 2)  // Commission plateforme
  creatorAmount  Decimal       @db.Decimal(10, 2)  // Part créateur
  // ❌ MANQUE: stripeFees
}
```

Métadonnées PaymentIntent actuelles :

```
metadata: {
  bookingId: booking.id,
  userId: user.userId,
  creatorId: booking.callOffer.creatorId,
  platformFee: platformFee.toFixed(2),
  creatorAmount: creatorAmount.toFixed(2),
  // ❌ MANQUE: stripeFees
}
```

✅ Solution proposée

Amélioration 1 : Ajouter stripeFees au schéma

Fichier: `prisma/schema.prisma`

```
model Payment {
  id                     String        @id @default(cuid())
  bookingId              String        @unique
  amount                 Decimal       @db.Decimal(10, 2)
  currency               String        @default("EUR")
  stripePaymentIntentId  String?       @unique
  status                 PaymentStatus @default(PENDING)
  platformFee            Decimal       @db.Decimal(10, 2)
  stripeFees             Decimal       @db.Decimal(10, 2) @default(0) // ✅ NOUVEAU
  creatorAmount          Decimal       @db.Decimal(10, 2)
  refundedAmount         Decimal       @db.Decimal(10, 2) @default(0)
  payoutReleaseDate      DateTime?
  payoutStatus           PayoutStatus  @default(PENDING)
  createdAt              DateTime      @default(now())
  updatedAt              DateTime      @updatedAt

  booking                Booking       @relation(fields: [bookingId], references:
[id])

  @@index([stripePaymentIntentId])
  @@map("Payment")
}
```

Amélioration 2 : Ajouter aux métadonnées

Déjà proposé dans la correction du Problème #1

Amélioration 3 : Afficher dans l'interface admin

Fichier: `app/dashboard/admin/payments/page.tsx`

```
// Ajouter une colonne "Frais Stripe" dans le tableau
<TableCell>
  <span className="text-sm text-gray-600">
    {payment.stripeFees ? `${payment.stripeFees.toFixed(2)} EUR` : 'N/A'}
  </span>
</TableCell>
```

📋 Plan d'Implémentation & Priorités

Phase 1 : URGENCE CRITIQUE (P0) - À implémenter immédiatement

🔴 Correction 1.1 : Frais Stripe absorbés par la plateforme

- **Fichier:** `lib/stripe.ts`
- **Action:** Modifier `createPaymentIntent()` pour inclure les frais Stripe dans `application_fee_amount`
- **Impact:** ✅ Créateurs reçoivent le montant promis
- **Durée estimée:** 1-2 heures
- **Tests requis:**
- [ ] Créer un paiement test de 100 EUR
- [ ] Vérifier que le créateur reçoit 85 EUR dans son solde Stripe
- [ ] Vérifier que la plateforme reçoit ~12.70 EUR (15 EUR - frais Stripe)

🔴 Correction 1.2 : Supprimer la constante obsolète

- **Fichier:** `lib/stripe.ts`
- **Action:** Supprimer `PLATFORM_FEE_PERCENTAGE` et la fonction `calculateFees()`
- **Impact:** ✅ Élimine la source de confusion
- **Durée estimée:** 30 minutes

🔴 Correction 2.1 : Corriger le dashboard admin

- **Fichier:** `app/dashboard/admin/page.tsx`
- **Action:** Utiliser `platformFeePercentage` au lieu de `platformCommissionRate`
- **Impact:** ✅ Dashboard affiche la bonne commission
- **Durée estimée:** 15 minutes

Phase 2 : HAUTE PRIORITÉ (P1) - À implémenter dans les 48h

🟠 Correction 3.1 : Corriger le fetch des payouts admin

- **Fichier:** `app/dashboard/admin/payouts/page.tsx`
- **Action:** S'assurer que `setPayouts()` reçoit le bon format de données
- **Impact:** ✅ Demandes PENDING_APPROVAL visibles
- **Durée estimée:** 1 heure
- **Tests requis:**
- [ ] Créer une demande de payout
- [ ] Vérifier qu'elle apparaît dans le dashboard admin
- [ ] Tester les boutons Approuver/Rejeter

🟠 Correction 3.2 : Améliorer les notifications admin

- **Fichier:** `app/api/payouts/request/route.ts`
- **Action:** Ajouter email de notification aux admins
- **Impact:** ✅ Admins sont alertés des nouvelles demandes
- **Durée estimée:** 1 heure

Phase 3 : AMÉLIORATION (P2) - À planifier

🟡 Amélioration 4.1 : Tracer les frais Stripe

- **Fichiers:** `prisma/schema.prisma`, `lib/stripe.ts`, interfaces admin
- **Action:** Ajouter champ `stripeFees` et l'afficher partout
- **Impact:** ✅ Transparence totale sur les frais
- **Durée estimée:** 2-3 heures

🟡 Tests end-to-end complets

- **Action:** Tests complets du flow paiement → payout
- **Durée estimée:** 2-3 heures

🧪 Checklist de Tests

Tests Paiements (Problème #1 et #2)

- [ ] **Test 1:** Paiement de 100 EUR par un utilisateur
- [ ] Créateur avec Stripe Connect reçoit exactement 85 EUR dans son solde
- [ ] Plateforme reçoit ~12.70 EUR (15 EUR - frais Stripe)
- [ ] Commission affichée partout : 15%
- [ ] **Test 2:** Paiement de 50 EUR par un utilisateur
- [ ] Créateur reçoit exactement 42.50 EUR (85% de 50)
- [ ] Plateforme reçoit ~6.20 EUR (15% - frais Stripe)
- [ ] **Test 3:** Vérifier métadonnées PaymentIntent
- [ ] `platformFee` = 15.00 (pour 100 EUR)
- [ ] `stripeFees` = ~2.30 (pour 100 EUR)
- [ ] `creatorAmount` = 85.00 (pour 100 EUR)

Tests Payouts (Problème #3)

- [ ] **Test 4:** Demande de payout par créateur
- [ ] Status = `PENDING_APPROVAL` dans la BDD
- [ ] Apparaît dans `/dashboard/admin/payouts`
- [ ] Boutons "Approuver" et "Rejeter" visibles
- [ ] Admin reçoit notification in-app
- [ ] Admin reçoit email de notification
- [ ] **Test 5:** Approbation de payout

- [ ] Status passe à `PROCESSING`
- [ ] Stripe payout créé
- [ ] Créateur reçoit notification
- [ ] Créateur reçoit email
- [ ] **Test 6:** Rejet de payout
- [ ] Status passe à `REJECTED`
- [ ] Raison enregistrée
- [ ] Créateur reçoit notification avec raison

Tests Dashboard Admin (Problème #2 et #3)

- [ ] **Test 7:** Dashboard admin charge correctement
- [ ] Commission affichée : 15%
- [ ] Revenus corrects
- [ ] Payouts PENDING_APPROVAL visibles
- [ ] Filtre par status fonctionne
- [ ] **Test 8:** Modification de la commission
- [ ] Admin peut changer de 15% à une autre valeur
- [ ] Changement sauvegardé
- [ ] Nouveaux paiements utilisent la nouvelle valeur
- [ ] Anciens paiements gardent l'ancienne valeur

📈 Impact Business

Revenus Plateforme

Avant corrections :

```
Paiement : 100 EUR
Commission plateforme : 15 EUR (15%)
Frais Stripe : ~2.30 EUR (payés par créateur ❌)
Net plateforme : 15.00 EUR
Net créateur : 82.70 EUR ❌
```

Après corrections (Option 1 recommandée) :

```
Paiement : 100 EUR
Commission plateforme : 15 EUR (15%)
Frais Stripe : ~2.30 EUR (absorbés par plateforme ✅)
Net plateforme : 12.70 EUR (-15% de revenus)
Net créateur : 85.00 EUR ✅
```

Impact financier :
- **Perte de revenus plateforme :** ~15% par transaction
- **Pour 1000 EUR de transactions :**
- Avant : 150 EUR de revenus plateforme

- Après : ~127 EUR de revenus plateforme
- Différence : -23 EUR (-15%)

Justification :
- ✅ Standard de l'industrie (OnlyFans, Patreon, Cameo)
- ✅ Protection et satisfaction des créateurs
- ✅ Transparence et prévisibilité
- ✅ Évite les problèmes légaux/réputationnels

Alternatives pour compenser

1. **Augmenter la commission plateforme de 15% à 17%**
   - Net plateforme resterait à ~14.70 EUR par 100 EUR

2. **Ajouter des frais fixes**
   - Ex: 0.50 EUR par transaction en plus des 15%

3. **Offres premium pour créateurs**
   - Commission réduite à 12% pour créateurs premium
   - Absorption des frais Stripe comme bénéfice

🎯 Résumé des Fichiers à Modifier

Modifications Critiques (P0)

```
lib/stripe.ts
├─ createPaymentIntent()
│  └─ Ajouter calcul frais Stripe dans application_fee_amount
├─ SUPPRIMER calculateFees()
└─ SUPPRIMER PLATFORM_FEE_PERCENTAGE

app/dashboard/admin/page.tsx
└─ Ligne 90: Utiliser platformFeePercentage au lieu de platformCommissionRate
```

Modifications Importantes (P1)

```
app/dashboard/admin/payouts/page.tsx
├─ Ligne 73: Ajouter useEffect avec [] pour fetch initial
└─ Ligne 90: Vérifier format de setPayouts(data)

app/api/payouts/request/route.ts
└─ Lignes 261-282: Ajouter email de notification aux admins
```

Améliorations (P2)

```
prisma/schema.prisma
└─ model Payment: Ajouter champ stripeFees

Migration à créer:
└─ prisma/migrations/xxx_add_stripe_fees.sql
```

📚 Ressources et Documentation

Documentation Stripe

- Destination Charges (https://stripe.com/docs/connect/destination-charges)
- Application Fees (https://stripe.com/docs/connect/charges#application-fees)
- Pricing (Fees) (https://stripe.com/pricing)

Standards de l'Industrie

- **OnlyFans:** Créateur reçoit 80%, plateforme 20% (inclut frais Stripe)
- **Patreon:** Créateur reçoit 90-95%, plateforme 5-10% (inclut frais)
- **Cameo:** Créateur reçoit 75%, plateforme 25% (inclut frais)

✅ Validation Finale

Avant mise en production

- [ ] Tous les tests passent (checklist ci-dessus)
- [ ] Code review par un développeur senior
- [ ] Tests en environnement de staging avec vrais comptes Stripe test
- [ ] Documentation mise à jour
- [ ] Communication aux créateurs existants des changements
- [ ] Monitoring des premiers paiements après déploiement

Métriques à surveiller

- Montants reçus par les créateurs (doivent être corrects)
- Revenus plateforme (baisse attendue de ~15%)
- Taux d'approbation des payouts
- Temps moyen d'approbation des payouts
- Nombre de demandes de support liées aux paiements

📞 Support et Suivi

Contact développeur :
Pour toute question sur cette analyse ou l'implémentation des corrections.

Suivi recommandé :
- Réunion quotidienne pendant la phase d'implémentation
- Revue hebdomadaire des métriques après déploiement
- Collecte de feedback des créateurs

Fin du rapport
Dernière mise à jour : 27 décembre 2025