📊 Rapport d'Analyse Approfondie - Bugs Payouts Call a Star

Date: 27 décembre 2024
Branche: `feature/stripe-payout-automation`
Analyste: DeepAgent AI

🎯 Résumé Exécutif

4 problèmes critiques identifiés dans le système de paiement :

1. ❌ **BUG D'AFFICHAGE DES MONTANTS (PRIORITÉ 1)** - Incohérence centimes/unités
2. ⚠️ **WORKFLOW PAYOUT INCOMPLET** - Pas de validation admin avant déclenchement Stripe
3. 🔔 **WEBHOOKS PAYOUTS MANQUANTS** - Événements `transfer.*` non gérés
4. 💱 **AGRÉGATION MULTI-DEVISES INCORRECTE** - Totaux mélangent EUR et CHF

---

1. 🐛 BUG D'AFFICHAGE DES MONTANTS (PRIORITÉ 1)

📝 Description du problème

Symptôme: Les montants s'affichent comme `0.17 CHF` au lieu de `17.00 CHF`.

Cause racine: Incohérence dans la gestion centimes vs unités entre Stripe API et base de données.

🔍 Analyse détaillée avec extraits de code

A. Stockage dans la base de données (EN UNITÉS)

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

```
// Ligne 82-88 : Calcul des montants EN UNITÉS (EUR/CHF)
const amount = Number(booking.totalPrice);  // Ex: 17.00
const platformFeePercentage = Number(settings.platformFeePercentage);
const platformFeeFixed = settings.platformFeeFixed ?
Number(settings.platformFeeFixed) : 0;
const platformFee = (amount * platformFeePercentage / 100) +
platformFeeFixed;  // Ex: 1.70
const creatorAmount = amount - platformFee;  // Ex: 15.30

// Ligne 123-134 : Stockage EN UNITÉS dans la DB
const payment = await db.payment.create({
  data: {
    bookingId: booking.id,
    amount,              // ✅ EN UNITÉS: 17.00
    currency: creatorCurrency,
    stripePaymentIntentId: paymentIntent.id,
    status: 'PENDING',
    platformFee,         // ✅ EN UNITÉS: 1.70
    creatorAmount,       // ✅ EN UNITÉS: 15.30
    payoutStatus: 'PENDING',
  },
});
```

B. Conversion pour Stripe API (EN CENTIMES)

Fichier: `lib/stripe.ts`

```
// Ligne 64-75 : Conversion en CENTIMES pour Stripe
export async function createPaymentIntent({
  amount,  // Reçoit 17.00
  currency = 'eur',
  metadata = {},
  stripeAccountId,
  platformFee,  // Reçoit 1.70
}) {
  const amountInCents = Math.round(amount * 100);  // ✅ 1700 centimes
  const platformFeeInCents = platformFee ? Math.round(platformFee * 100) : 0;  // ✅
170 centimes
  const creatorAmountInCents = amountInCents - platformFeeInCents;  // ✅ 1530
centimes

  const paymentIntentParams: Stripe.PaymentIntentCreateParams = {
    amount: amountInCents,  // ✅ Envoyé à Stripe en centimes
    currency,
    metadata: {
      ...metadata,
      platformFee: String(platformFee || 0),  // ⚠️ Metadata EN UNITÉS: "1.70"
      creatorAmount: String((amountInCents - platformFeeInCents) / 100),  // ⚠️
Metadata EN UNITÉS: "15.30"
    },
    // ...
  };
```

✅ **Bonne pratique:** La conversion en centimes est correcte pour l'API Stripe.

C. Récupération depuis webhook (EN UNITÉS depuis metadata)

Fichier: `app/api/payments/webhook/route.ts`

```
// Ligne 515-519 : Récupération depuis metadata (EN UNITÉS)
async function handlePaymentIntentSucceeded(event: Stripe.Event): Promise<void> {
  const paymentIntent = event.data.object as Stripe.PaymentIntent;

  const amount = Number(booking.totalPrice);  // ✅ EN UNITÉS depuis DB
  const platformFee = Number(paymentIntent.metadata?.platformFee || 0);  // ✅ EN
UNITÉS depuis metadata
  const creatorAmount = Number(paymentIntent.metadata?.creatorAmount || 0);  // ✅ EN
UNITÉS depuis metadata

  // Ligne 523-540 : Stockage EN UNITÉS
  const payment = await prisma.payment.upsert({
    where: { bookingId: booking.id },
    update: {
      status: 'SUCCEEDED',
      payoutReleaseDate,
      payoutStatus: 'HELD',
    },
    create: {
      bookingId: booking.id,
      amount,           // ✅ EN UNITÉS: 17.00
      stripePaymentIntentId: paymentIntent.id,
      status: 'SUCCEEDED',
      platformFee,      // ✅ EN UNITÉS: 1.70
      creatorAmount,    // ✅ EN UNITÉS: 15.30
      payoutStatus: 'HELD',
      payoutReleaseDate,
    },
  });
}
```

D. ❌ PROBLÈME : Affichage avec division incorrecte

Fichier: `app/api/payments/webhook/route.ts`

```
// Ligne 179, 205, 273, 298 : ❌ Division par 100 INCORRECTE
// Ces lignes SUPPOSENT que payout.amount est en centimes, mais il est EN UNITÉS !

// Dans handlePayoutPaid :
message: `Un paiement de ${(payout.amount / 100).toFixed(2)} ${currency} a été trans-
féré...`
// Si payout.amount = 17.00 (EN UNITÉS), cela affiche : 0.17 CHF ❌

// CORRECT serait :
message: `Un paiement de ${Number(payout.amount).toFixed(2)} ${currency} a été trans-
féré...`
// Afficherait : 17.00 CHF ✅
```

MAIS dans d'autres endroits :

```
// Ligne 984, 1044 : ✅ Affichage CORRECT sans division
message: `Votre paiement de ${(stripePayout.amount / 100).toFixed(2)} EUR a été trans-
féré...`
// ICI c'est correct car stripePayout.amount vient DIRECTEMENT de Stripe (en centimes)

message: `Votre paiement de ${Number(payout.amount).toFixed(2)} EUR a été trans-
féré...`
// ICI aussi correct car payout.amount vient de notre DB (en unités)
```

E. Composants d’affichage

Fichier: `components/admin/CurrencyDisplay.tsx`

```
// Ligne 21-22, 47-52 : Affichage direct sans conversion
const numAmount = typeof amount === 'string' ? parseFloat(amount) : amount;

const formatAmount = (value: number) => {
  return value.toFixed(2);  // ✅ Correct SI amount est déjà en unités
};

// Ligne 59-66 : Rendu
return (
  <span className={cn('font-medium', className)}>
    {showSymbol && hasSymbol && symbol}
    {formattedAmount}  {/* Affiche directement la valeur */}
    {' '}
    {showSymbol && !hasSymbol && symbol}
    {!showSymbol && currencyCode}
  </span>
);
```

Fichier: `components/ui/currency-display.tsx`

```
// Ligne 27 : Affichage direct
return (
  <span className={className}>
    {amount.toFixed(2)} {displayCurrency}
  </span>
);
```

✅ **Ces composants sont corrects** - ils affichent ce qu’on leur donne.

F. Schéma Prisma (Stockage EN UNITÉS avec Decimal)

Fichier: `prisma/schema.prisma`

```
model Payment {
  id                    String        @id @default(cuid())
  bookingId             String        @unique
  amount                Decimal       @db.Decimal(10, 2)  // ✅ EN UNITÉS (ex: 17.00)
  currency              String        @default("EUR")
  stripePaymentIntentId String
  status                PaymentStatus @default(PENDING)
  platformFee           Decimal       @db.Decimal(10, 2)  // ✅ EN UNITÉS (ex: 1.70)
  creatorAmount         Decimal       @db.Decimal(10, 2)  // ✅ EN UNITÉS (ex: 15.30)
  // ...
}

model Payout {
  id              String       @id @default(cuid())
  creatorId       String
  amount          Decimal      @db.Decimal(10, 2)  // ✅ EN UNITÉS (EUR)
  stripePayoutId  String?
  status          PayoutStatus @default(PENDING)
  // ✅ NEW: Currency conversion tracking
  amountPaid      Decimal?     @db.Decimal(10, 2)  // ✅ EN UNITÉS (CHF/autre)
  currency        String?      @default("EUR")
  conversionRate  Decimal?     @db.Decimal(10, 6)
  // ...
}
```

🎯 Localisation précise des bugs

Fichier	Lignes	Bug	Impact
`app/api/payments/webhook/route.ts`	179, 205, 273, 298	Division `/100` sur `payout.amount` qui est déjà en unités	Affiche 0.17 au lieu de 17.00 dans notifications/emails
`app/api/payouts/request/route.ts`	127-134	Division `/100` sur balance Stripe puis stockage direct	Potentiellement stocke 0.17 au lieu de 17.00
`lib/payout-eligibility.ts`	171-174	Division `/100` sur balance Stripe correct	✅ Correct (balance vient de Stripe)

📍 Tous les endroits où le bug se manifeste

1. **Notifications utilisateurs** (webhooks payout.paid/failed)
2. **Emails de confirmation de payout** (webhooks)
3. **Dashboard créateur** - section revenus (si utilise données incorrectes)
4. **Dashboard admin** - historique payouts (selon source des données)

💡 Solution

Règle simple à suivre :

- ✅ Données depuis **Stripe API directe** (balance, payout objects) → **EN CENTIMES** → diviser par 100
- ✅ Données depuis **notre DB** (Payment, Payout models) → **EN UNITÉS** → utiliser directement
- ✅ Données depuis **metadata Stripe** (créées par nous) → **EN UNITÉS** → utiliser directement

Corrections à effectuer :

```
// ❌ AVANT (webhook ligne 179) :
message: `Un paiement de ${(payout.amount / 100).toFixed(2)} ${currency}...`

// ✅ APRÈS :
message: `Un paiement de ${Number(payout.amount).toFixed(2)} ${currency}...`

// OU déterminer la source :
const amountToDisplay = payout.amount;  // depuis notre DB = déjà en unités
// vs
const amountToDisplay = stripePayout.amount / 100;  // depuis Stripe API = en centimes
```

---

2. ⚠️ WORKFLOW PAYOUT INCOMPLET

📝 Description du workflow actuel

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

Flux actuel (PROBLÈME)

```
// Ligne 204-215 : Création immédiate du Payout record
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.PROCESSING,  // ⚠️ Directement PROCESSING
  },
});

// Ligne 235-252 : Déclenchement IMMÉDIAT de Stripe payout
const stripePayout = await createConnectPayout({
  amount: payoutAmountInStripeCurrency,
  currency: stripeCurrency.toLowerCase(),
  stripeAccountId: creator.stripeAccountId,
  metadata: {
    creatorId: creator.id,
    payoutId: payout.id,
    manual: 'true',
    requestedBy: jwtUser.userId,
    adminOverride: String(adminOverride && isAdmin),
    // ...
  },
});

// ⚠️ PAS DE VALIDATION ADMIN - Le payout est IMMÉDIATEMENT envoyé à Stripe !
```

Flux automatique (CRON)

Fichier: `app/api/cron/process-payouts/route.ts`

```
// Ligne 184-189 : Création du Payout avec status PROCESSING
const payout = await prisma.payout.create({
  data: {
    creatorId: creator.id,
    amount: totalAmount,
    status: PayoutStatus.PROCESSING,  // ⚠️ Directement PROCESSING
  },
});

// Ligne 217-230 : Déclenchement IMMÉDIAT de Stripe transfer
const transfer = await createPayout({
  amount: totalAmount,
  stripeAccountId: creator.stripeAccountId!,
  metadata: {
    creatorId: creator.id,
    payoutId: payout.id,
    paymentIds: readyPayments.map((p) => p.id).join(','),
    paymentCount: String(readyPayments.length),
    automatic: 'true',
    frequency: schedule.frequency,
  },
});

// ⚠️ PAS DE VALIDATION ADMIN - Le payout est IMMÉDIATEMENT envoyé à Stripe !
```

❌ Problèmes identifiés

1. **Pas de validation admin** avant déclenchement Stripe
2. **Statut PROCESSING immédiat** au lieu de PENDING → APPROVED → PROCESSING
3. **Pas d'interface admin** pour approuver/rejeter les demandes
4. **Pas d'audit trail** pour validation admin

✅ Workflow attendu (style OnlyFans/Patreon)

```
1. Créateur demande payout
   ↓
2. Status: PENDING (en attente validation admin)
   ↓
3. Admin review dans dashboard
   → Approuve: Status → APPROVED → Déclenchement Stripe → PROCESSING
   → Rejette: Status → REJECTED (avec raison)
   ↓
4. Webhook payout.paid → Status: PAID
   ou
   Webhook payout.failed → Status: FAILED
```

📊 État actuel vs attendu

Aspect	Actuel	Attendu
Création demande	✅ Route `/api/payouts/request`	✅ OK
Validation admin	❌ Aucune	✅ Route `/api/admin/payouts/approve`
Interface admin	⚠️ Lecture seule	✅ Actions approve/reject
Status flow	PENDING → PROCESSING	PENDING → APPROVED → PROCESSING
Audit log	⚠️ Incomplet	✅ Toutes actions admin logged
Déclenchement Stripe	❌ Immédiat	✅ Après approbation

📁 Fichiers concernés

Existants à modifier :

- `app/api/payouts/request/route.ts` - Créer avec status PENDING
- `app/api/cron/process-payouts/route.ts` - Créer avec status PENDING_APPROVAL
- `app/dashboard/admin/payouts/page.tsx` - Ajouter boutons approve/reject

À créer :

- `app/api/admin/payouts/approve/route.ts` - Approuver et déclencher Stripe
- `app/api/admin/payouts/reject/route.ts` - Rejeter avec raison
- `app/api/admin/payouts/pending/route.ts` - Liste des payouts en attente (existe déjà !)

🔍 Interface admin actuelle

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

```
// Ligne 184-259 : Tableau lecture seule
{payouts.length > 0 ? (
  <div className="overflow-x-auto">
    <table className="w-full text-sm">
      <thead className="border-b">
        <tr className="text-left">
          <th className="py-3 px-2">ID</th>
          <th className="py-3 px-2">Créateur</th>
          <th className="py-3 px-2">Montant</th>
          <th className="py-3 px-2">Statut</th>
          <th className="py-3 px-2">Raison échec</th>
          <th className="py-3 px-2">Tentatives</th>
          <th className="py-3 px-2">Date création</th>
          <th className="py-3 px-2">Date paiement</th>
          <th className="py-3 px-2">Actions</th>  {/* ⚠️ Seulement "View" */}
        </tr>
      </thead>
      <tbody>
        {payouts.map((payout) => (
          <tr key={payout.id} className="border-b hover:bg-gray-50">
            {/* ... */}
            <td className="py-3 px-2">
              <Button
                size="sm"
                variant="ghost"
                onClick={() => handleViewDetails(payout)}
              >
                <Eye className="w-4 h-4" />
              </Button>
              {/* ❌ PAS de boutons Approve/Reject */}
            </td>
          </tr>
        ))}
      </tbody>
    </table>
  </div>
```

✅ **Une route** `/api/admin/payouts/pending` **existe déjà !** (ligne 10 du listing grep)

---

3. 🔔 WEBHOOKS PAYOUTS MANQUANTS

📝 Webhooks actuellement gérés

Fichier: `app/api/payments/webhook/route.ts`

```
// Ligne 391-450 : Liste des événements gérés
async function processWebhookEvent(event: Stripe.Event): Promise<void> {
  switch (event.type) {
    case 'payment_intent.succeeded':       // ✅ Géré
    case 'payment_intent.payment_failed':  // ✅ Géré
    case 'charge.refunded':                // ✅ Géré
    case 'charge.dispute.created':         // ✅ Géré
    case 'charge.dispute.closed':          // ✅ Géré
    case 'charge.dispute.funds_withdrawn': // ✅ Géré
    case 'charge.dispute.funds_reinstated':// ✅ Géré
    case 'payout.paid':                    // ✅ Géré (ligne 420-422, 927-1051)
    case 'payout.failed':                  // ✅ Géré (ligne 424-425, 1056-1143)
    case 'transfer.reversed':              // ✅ Géré (ligne 428-429, 1148-1191)
    case 'account.updated':                // ✅ Géré (ligne 432-433, 1198-1276)
    case 'capability.updated':             // ✅ Géré (ligne 436-437, 1282-1346)
    case 'account.application.authorized': // ✅ Géré (ligne 440-441, 1352-1368)
    case 'account.application.deauthorized':// ✅ Géré (ligne 444-445, 1374-1420)

    // ❌ MANQUANTS :
    // - payout.created
    // - payout.canceled
    // - payout.updated
    // - transfer.created
    // - transfer.failed
    // - transfer.updated

    default:
      console.log(`[Webhook] Unhandled event type: ${event.type}`);
  }
}
```

✅ Webhooks déjà implémentés dans le handler

Ligne 107-136 : `payout.created` - Traitement basique (update audit log)
Ligne 138-226 : `payout.paid` - ✅ Complet (notifications, emails)
Ligne 228-322 : `payout.failed` - ✅ Complet (notifications, emails)

❌ Webhooks manquants à ajouter

Événement	Priorité	Utilité	Status actuel
`payout.created`	⚠️ Moyenne	Confirmer création dans Stripe	⚠️ Partiellement géré (hors switch)
`payout.canceled`	🔴 Haute	Gérer annulation payout	❌ Non géré
`payout.updated`	🟡 Basse	Tracking changements status	❌ Non géré
`transfer.created`	⚠️ Moyenne	Confirmer création transfer	❌ Non géré
`transfer.failed`	🔴 Haute	Gérer échec transfer (différent de payout.failed)	❌ Non géré
`transfer.updated`	🟡 Basse	Tracking changements status	❌ Non géré

🔍 Distinction transfer vs payout

Important : Dans Stripe Connect, on utilise actuellement **deux méthodes différentes** :

1. **Destination charges** (automatic routing) :
   `typescript`
   `// lib/stripe.ts ligne 86-90`
   `paymentIntentParams.application_fee_amount = platformFeeInCents;`
   `paymentIntentParams.transfer_data = {`
   `destination: stripeAccountId,`
   `};`
   → Génère événements `transfer.*`

2. **Payouts manuels** (Connect account balance → bank) :
   `typescript`
   `// app/api/payouts/request/route.ts ligne 238-252`
   `const stripePayout = await createConnectPayout({...});`
   → Génère événements `payout.*`

📊 Événements à gérer selon méthode

Méthode	Événements à gérer	Actuellement géré
Destination charges	`transfer.created`, `transfer.failed`, `transfer.reversed`	⚠️ Seulement `transfer.reversed`
Connect payouts	`payout.created`, `payout.paid`, `payout.failed`, `payout.canceled`	✅ Tous sauf `payout.canceled`

💡 Actions à effectuer

1. ✅ Ajouter cases dans le switch :

```typescript
  case 'payout.canceled':
    await handlePayoutCanceled(event);
    break;
  case 'transfer.created':
    await handleTransferCreated(event);
    break;
  case 'transfer.failed':
    await handleTransferFailed(event);
    break;
```

2. ✅ Implémenter les handlers correspondants
3. ✅ Logger tous les événements dans TransactionLog

---

4. 💱 AGRÉGATION MULTI-DEVISES INCORRECTE

📝 Problème d'agrégation

Fichier: `app/api/admin/payouts/dashboard/route.ts`

```
// Ligne 148-163 : Agrégation sans tenir compte de la devise
const payoutVolumeResult = await prisma.payout.aggregate({
  where: {
    status: PayoutStatus.PAID,
    createdAt: {
      gte: thirtyDaysAgo,
    },
  },
  _sum: {
    amount: true,  // ❌ Additionne tous les montants sans distinction de devise !
  },
  _count: true,
});

const totalPayoutVolume = Number(payoutVolumeResult._sum.amount || 0);
// ❌ Si on a 100 EUR + 50 CHF, on obtient 150... mais 150 quoi ? EUR ? CHF ?

// Ligne 244 : Affiché comme EUR par défaut
totalPayoutVolume: totalPayoutVolume.toFixed(2),
currency: 'EUR',  // ⚠️ Assumé EUR alors que ça peut être un mix !
```

❌ Problème identifié

Scénario :

- Payout 1 : 100.00 EUR (creator avec compte EUR)
- Payout 2 : 50.00 CHF (creator avec compte CHF)
- Payout 3 : 30.00 EUR (creator avec compte EUR)

Agrégation actuelle :

```
SELECT SUM(amount) FROM Payout WHERE status = 'PAID'
-- Résultat : 180.00
-- ❌ INCORRECT : On additionne 100 EUR + 50 CHF + 30 EUR = 180 ??
```

Affichage :

```
Total payout volume: 180.00 EUR
```

❌ **FAUX** : Ce n’est pas 180 EUR, c’est un mélange !

✅ Solution correcte

Option 1 : Agrégation par devise

```
// Ligne 148-163 : GROUP BY currency
const payoutVolumeByurrency = await prisma.payout.groupBy({
  by: ['currency'],
  where: {
    status: PayoutStatus.PAID,
    createdAt: { gte: thirtyDaysAgo },
  },
  _sum: {
    amount: true,
  },
  _count: true,
});

// Résultat :
// [
//   { currency: 'EUR', _sum: { amount: 130.00 }, _count: 2 },
//   { currency: 'CHF', _sum: { amount: 50.00 }, _count: 1 }
// ]

// Ligne 235-252 : Retourner par devise
return NextResponse.json({
  summary: {
    totalPayoutVolumeByCurrency: payoutVolumeByurrency.map(c => ({
      currency: c.currency,
      amount: Number(c._sum.amount || 0).toFixed(2),
      count: c._count,
    })),
    // ...
  },
});
```

Option 2 : Conversion en devise de référence (EUR)

```
// Si on a des taux de conversion stockés dans Payout.conversionRate
const payouts = await prisma.payout.findMany({
  where: {
    status: PayoutStatus.PAID,
    createdAt: { gte: thirtyDaysAgo },
  },
  select: {
    amount: true,
    currency: true,
    conversionRate: true,
  },
});

// Convertir tout en EUR
let totalInEur = 0;
for (const payout of payouts) {
  if (payout.currency === 'EUR') {
    totalInEur += Number(payout.amount);
  } else if (payout.conversionRate) {
    // Convertir en EUR en utilisant le taux inverse
    totalInEur += Number(payout.amount) / Number(payout.conversionRate);
  }
}

return NextResponse.json({
  summary: {
    totalPayoutVolume: totalInEur.toFixed(2),
    currency: 'EUR',  // ✅ Maintenant c'est correct !
    currencyNote: 'Tous les montants convertis en EUR au taux du jour du payout',
  },
});
```

📊 Autres endroits avec le même problème

Fichier	Ligne	Problème
`app/api/admin/payouts/dashboard/route.ts`	184-194	Agrégation `creatorAmount` sans distinction devise
`app/api/payouts/creator/route.ts`	62-74	Totaux calculés sans distinction devise
`app/dashboard/creator/page.tsx`	431	`totalRevenue` calculé sans distinction devise

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

```
// Ligne 62-74 : ❌ Agrégation incorrecte
const totalEarnings = payments
  .filter(p => p.payoutStatus === 'PAID')
  .reduce((sum, p) => sum + Number(p.creatorAmount), 0);
  // ❌ Si payments contient du EUR et du CHF, on mélange tout !

const pendingEarnings = payments
  .filter(p => p.payoutStatus === 'HELD' || p.payoutStatus === 'READY')
  .reduce((sum, p) => sum + Number(p.creatorAmount), 0);
  // ❌ Pareil

const readyForPayout = payments
  .filter(p => p.payoutStatus === 'READY')
  .reduce((sum, p) => sum + Number(p.creatorAmount), 0);
  // ❌ Pareil
```

✅ **Solution :** Un créateur ne peut avoir qu'une seule devise (champ `creator.currency`), donc pour les routes créateurs c'est OK. **MAIS** il faut vérifier que tous les payments d'un créateur ont la même devise !

Fichier: `app/dashboard/creator/page.tsx`

```
// Ligne 325, 431 : ✅ OK pour un créateur
const totalRevenue = payments.reduce((sum, p) => sum + Number(p?.creatorAmount ?? 0),
0);
// Si le créateur a une devise unique (CHF), tous ses payments sont en CHF
// Ligne 431 : Affichage avec la devise du créateur
<div className="text-3xl font-bold">{totalRevenue.toFixed(2)} {creatorCurrency}</div>
// ✅ Correct si creatorCurrency est bien défini
```

💡 Recommandation

1. **Pour dashboard admin** : Implémenter agrégation par devise (Option 1)
2. **Pour dashboard créateur** : ✅ Déjà correct (une seule devise par créateur)
3. **Vérification** : Ajouter une contrainte pour s'assurer qu'un créateur ne reçoit que des payments dans sa devise

---

📋 Résumé des actions correctives

🔴 Priorité 1 - Bug affichage montants

- [ ] Corriger `app/api/payments/webhook/route.ts` lignes 179, 205, 273, 298
- [ ] Vérifier tous les usages de `payout.amount` et `payment.creatorAmount`
- [ ] Ajouter commentaires explicites : `// EN UNITÉS depuis DB` vs `// EN CENTIMES depuis Stripe`
- [ ] Tests : Vérifier affichage notifications et emails

🟡 Priorité 2 - Workflow payout validation admin

- [ ] Modifier `app/api/payouts/request/route.ts` : créer avec status `PENDING_APPROVAL`
- [ ] Créer `app/api/admin/payouts/approve/route.ts`
- [ ] Créer `app/api/admin/payouts/reject/route.ts`
- [ ] Modifier `app/dashboard/admin/payouts/page.tsx` : ajouter boutons approve/reject
- [ ] Ajouter filtre "Pending approval" dans l'interface admin

- [ ] Tests : Workflow complet demande → approbation → déclenchement

🟡 Priorité 2 - Webhooks manquants

- [ ] Ajouter handler `handlePayoutCanceled()`
- [ ] Ajouter handler `handleTransferCreated()`
- [ ] Ajouter handler `handleTransferFailed()`
- [ ] Ajouter cases dans le switch principal
- [ ] Tests : Simuler événements avec Stripe CLI

🟢 Priorité 3 - Agrégation multi-devises

- [ ] Modifier `app/api/admin/payouts/dashboard/route.ts` : agrégation par devise
- [ ] Modifier interface admin pour afficher “100 EUR, 50 CHF, 30 USD”
- [ ] Option : Ajouter conversion en devise de référence avec disclaimer
- [ ] Tests : Créer payouts dans différentes devises et vérifier affichage

---

🧪 Scénarios de test recommandés

Test 1 : Bug affichage montants

1. Créer un payout de 17.00 CHF
2. Déclencher webhook `payout.paid`
3. Vérifier notification affiche “17.00 CHF” et non “0.17 CHF”
4. Vérifier email affiche montant correct

Test 2 : Workflow validation admin

1. Créateur demande payout de 50 EUR
2. Vérifier status = `PENDING_APPROVAL`
3. Admin voit demande dans dashboard
4. Admin approuve
5. Vérifier status → `PROCESSING` et déclenchement Stripe
6. Webhook `payout.paid` → status `PAID`

Test 3 : Webhooks

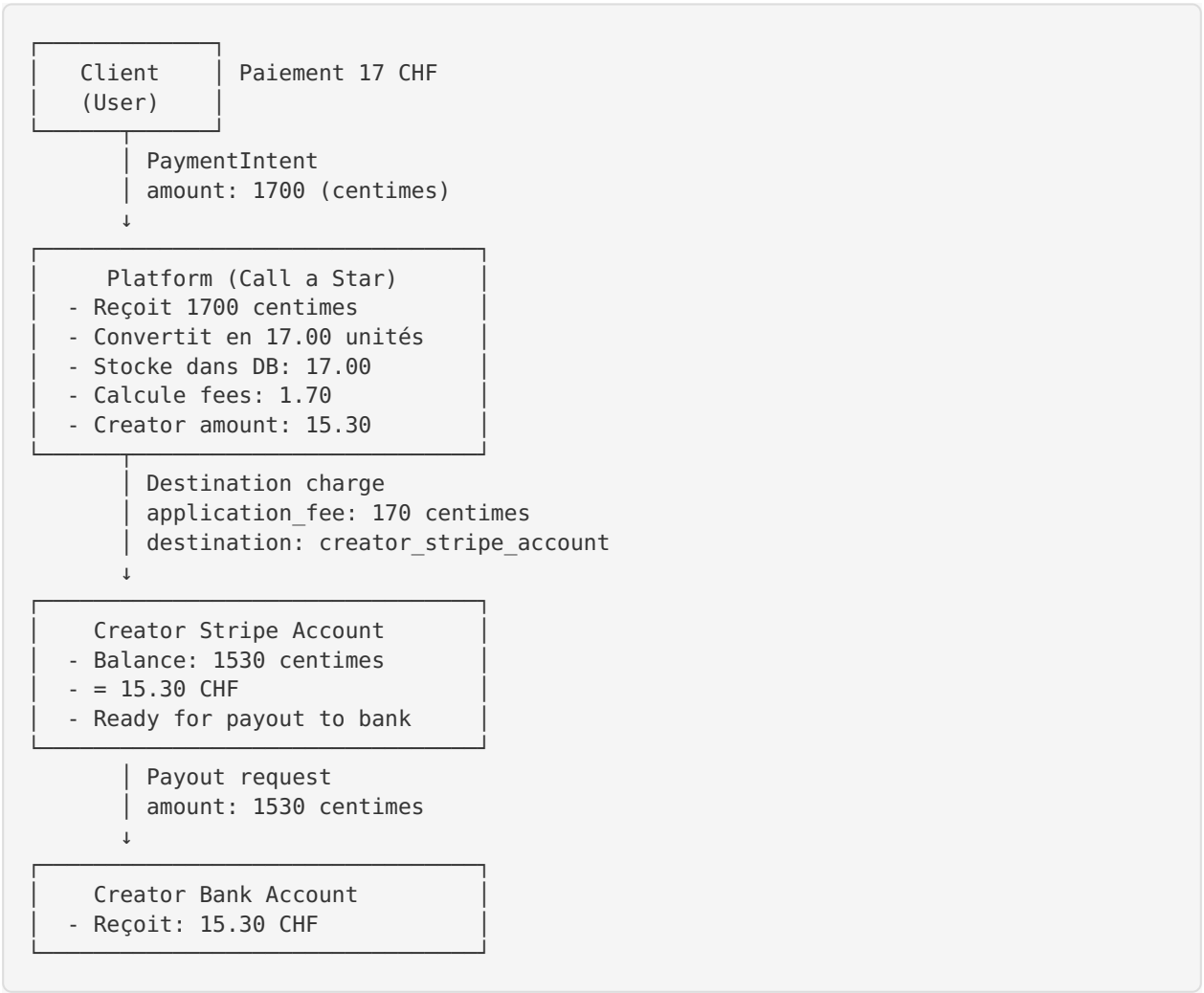
1. Simuler `transfer.failed` avec Stripe CLI
2. Vérifier logging dans TransactionLog
3. Vérifier update status dans DB
4. Vérifier notification admin

Test 4 : Multi-devises

1. Créer 2 créateurs : un EUR, un CHF
2. Créer payouts : 100 EUR, 50 CHF, 30 EUR
3. Dashboard admin affiche :
   - EUR: 130.00 (2 payouts)
   - CHF: 50.00 (1 payout)

📚 Documentation technique

Architecture Stripe Connect



Flux de données montants

```
1. API Stripe      → Montants EN CENTIMES
2. PaymentIntent   → metadata EN UNITÉS (notre choix)
3. DB (Prisma)     → Montants EN UNITÉS (Decimal)
4. Composants UI   → Affichage direct EN UNITÉS
5. Webhooks        →
   - stripePayout.amount : EN CENTIMES (direct Stripe)
   - payout.amount (DB) : EN UNITÉS (notre DB)
```

Fin du rapport

Généré par DeepAgent AI - 27 décembre 2024