



Guide Complet Stripe

Intégration Paiement pour MyParcs

Buy Me a Coffee • Abonnements & Plans



TABLE DES MATIÈRES

1. Présentation de Stripe & Prérequis
2. Création du Compte & Configuration
3. Installation & Setup Technique
4. Use Case 1 — Buy Me a Coffee
5. Use Case 2 — Abonnements & Plans
6. Webhooks — Écouter les Événements Stripe
7. Mode Test & Cartes de Test
8. Passage en Production
9. Sécurité & Bonnes Pratiques
10. Dépannage & Erreurs Courantes

1. PRÉSENTATION DE STRIPE & PRÉREQUIS

Pourquoi Stripe ?

Stripe est la solution de paiement leader mondial, utilisée par des millions de sites. Elle offre une API simple, ultra-documentée, et gère tout : cartes, virements, abonnements, factures.

☑ Points forts	📦 Ce que tu vas faire
API REST simple	Créer un bouton "Buy Me a Coffee"
Dashboard complet	Gérer les abonnements MyParcs
Mode test intégré	Envoyer des factures auto
Webhooks en temps réel	Réagir aux événements paiement
Support 135+ devises	Tester sans argent réel
Conformité PCI DSS	Passer en production

Prérequis

- Un compte Stripe (gratuit à créer sur stripe.com)
- PHP 7.4+ sur votre serveur (vous l'avez déjà)
- Composer installé (gestionnaire de paquets PHP)
- Accès HTTPS sur votre domaine (obligatoire en prod)
- Votre fichier db.php et auth.php en place (déjà ok)

💡 Tarification Stripe

- 0 € d'abonnement mensuel
- 1,5% + 0,25€ par transaction (cartes européennes)
- 2,9% + 0,30\$ pour les cartes non-EU
- Gratuit pour les remboursements

2. CRÉATION DU COMPTE & CONFIGURATION

Étape 1 : Créer votre compte

1. Aller sur <https://dashboard.stripe.com/register>
2. Saisir email, nom, pays (France) et créer le compte
3. Vérifier votre email
4. Remplir les informations de votre entreprise

Étape 2 : Récupérer vos clés API

Dans le Dashboard Stripe → Développeurs → Clés API, vous trouvez :

Vos deux clés essentielles

Clé publique (pk_test_...) → Côté client, visible dans le JS

Clé secrète (sk_test_...) → Côté serveur SEULEMENT, ne jamais exposer !

En mode test : clés commençant par pk_test_ et sk_test_

En production : clés commençant par pk_live_ et sk_live_

Étape 3 : Créer les Produits dans le Dashboard

Avant de coder, créez vos produits dans Stripe → Catalogue de produits :

- Produit "Buy Me a Coffee" → Paiement unique → Prix fixe (ex: 3€, 5€, 10€)
- Produit "Plan Fan" → Abonnement récurrent → mensuel/annuel
- Produit "Plan VIP" → Abonnement récurrent → mensuel/annuel

Notez bien les Price ID générés (ex: price_1ABC...xyz) → vous en aurez besoin dans le code.

3. INSTALLATION & SETUP TECHNIQUE

Installation du SDK Stripe via Composer

```
# Dans le terminal, à la racine de votre projet :  
composer require stripe/stripe-php  
  
# Vérifier l'installation :  
php -r "require 'vendor/autoload.php'; echo Stripe\Stripe::VERSION;"
```

Fichier de configuration stripe_config.php

Créez ce fichier à la racine du projet et NE le mettez JAMAIS en ligne sur GitHub :

```
<?php  
// stripe_config.php - FICHIER PRIVÉ (ajouter dans .gitignore)  
  
define('STRIPE_SK', 'sk_test_VOTRE_CLE_SECRETE_ICI');  
define('STRIPE_PK', 'pk_test_VOTRE_CLE_PUBLIQUE_ICI');  
define('STRIPE_WEBHOOK_SECRET', 'whsec_VOTRE_SECRET_WEBHOOK');  
  
// IDs de vos produits créés dans le Dashboard  
define('STRIPE_PRICE_FAN_MONTHLY', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_FAN_YEARLY', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_VIP_MONTHLY', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_VIP_YEARLY', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_COFFEE_3', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_COFFEE_5', 'price_XXXXXXXXXXXXXXXX');  
define('STRIPE_PRICE_COFFEE_10', 'price_XXXXXXXXXXXXXXXX');
```

Ajouter stripe_config.php dans votre .gitignore

```
# .gitignore  
stripe_config.php  
vendor/  
.env
```

4. USE CASE 1 — BUY ME A COFFEE

Principe de fonctionnement

Le bouton "Buy Me a Coffee" permet à vos utilisateurs de faire un don unique en quelques clics. Stripe gère tout le formulaire de paiement via son Checkout sécurisé.

Flux simplifié

1. L'utilisateur clique sur "Buy Me a Coffee"
2. Redirection vers la page Stripe Checkout (sécurisée)
3. L'utilisateur saisit ses infos de carte
4. Stripe débite et vous notifie via webhook
5. Retour sur votre site avec page de succès

create_coffee_session.php — Créer la session de paiement

```
<?php
// create_coffee_session.php
require 'vendor/autoload.php';
require 'stripe_config.php';
require 'auth.php';
requireLogin();

\Stripe\Stripe::setApiKey(STRIPE_SK);

$amount = $_POST["amount"] ?? "5"; // 3, 5 ou 10
$priceId = constant("STRIPE_PRICE_COFFEE_" . $amount);
$userId = $_SESSION["user_id"];

$session = \Stripe\Checkout\Session::create([
    "mode" => "payment",
    "payment_method_types" => ["card"],
    "line_items" => [[
        "price" => $priceId,
        "quantity" => 1,
    ]],
    "success_url" =>
    "https://parc.cyberethique.com/coffee_success.php?session_id={CHECKOUT_SESSION_ID}",
    "cancel_url" => "https://parc.cyberethique.com/dashboard.php",
    "metadata" => ["user_id" => $userId, "type" => "coffee"],
    "customer_email" => $_SESSION["user_email"] ?? null,
]);

header("Location: " . $session->url);
```

```
exit;
```

Le bouton HTML dans votre page

```
<div class="coffee-widget">
  <h3>☕ Soutenir le projet MyParcs</h3>
  <p>Si vous aimez l'application, offrez-moi un café !</p>

  <form method="POST" action="create_coffee_session.php">
    <div class="coffee-options">
      <label>
        <input type="radio" name="amount" value="3"> ☕ 3€
      </label>
      <label>
        <input type="radio" name="amount" value="5" checked> ☕☕ 5€
      </label>
      <label>
        <input type="radio" name="amount" value="10"> ☕☕☕ 10€
      </label>
    </div>
    <button type="submit" class="btn-coffee">
      ☕ Offrir un café !
    </button>
  </form>
</div>
```

coffee_success.php — Page de confirmation

```
<?php
// coffee_success.php
require 'vendor/autoload.php';
require 'stripe_config.php';
require 'db.php';
require 'auth.php';
requireLogin();

\Stripe\Stripe::setApiKey(STRIPE_SK);

$sessionId = $_GET["session_id"] ?? "";

// Récupérer la session Stripe pour vérification
$session = \Stripe\Checkout\Session::retrieve($sessionId);

if ($session->payment_status === "paid") {
    $userId    = $session->metadata->user_id;
    $amount    = $session->amount_total / 100; // Stripe stocke en centimes
```

```
// Enregistrer en BDD
$stmt = $pdo->prepare("INSERT INTO coffees (user_id, amount,
stripe_session_id, paid_at)
VALUES (?, ?, ?, NOW())");
$stmt->execute([$userId, $amount, $sessionId]);

echo "Merci ! ☺ Votre soutien est bien reçu !";
}
```

5. USE CASE 2 — ABONNEMENTS & PLANS

Principe des abonnements Stripe

Pour les abonnements, Stripe gère le prélèvement récurrent automatiquement chaque mois/année. Vous n'avez pas à gérer les dates ni les relances.

Flux abonnement

1. L'utilisateur choisit son plan (Fan ou VIP) et sa durée (mensuel/annuel)
2. Redirection vers Stripe Checkout en mode "subscription"
3. Stripe crée un Customer + Subscription automatiquement
4. À chaque renouvellement, Stripe envoie un webhook
5. Vous activez/désactivez l'accès selon les événements

create_subscription_session.php

```
<?php
// create_subscription_session.php
require 'vendor/autoload.php';
require 'stripe_config.php';
require 'db.php';
require 'auth.php';
requireLogin();

\Stripe\Stripe::setApiKey(STRIPE_SK);

$planId    = $_POST["plan_id"] ?? 2; // 2=Fan, 3=VIP
$interval  = $_POST["interval"] ?? "monthly"; // monthly ou yearly
$userId    = $_SESSION["user_id"];

// Déterminer le bon Price ID
$key       = ($planId == 3 ? "VIP" : "FAN") . "_" . strtoupper($interval);
$priceId   = constant("STRIPE_PRICE_" . $key);

// Vérifier si Customer Stripe existe déjà
$stmt = $pdo->prepare("SELECT stripe_customer_id FROM users WHERE id = ?");
$stmt->execute([$userId]);
$user = $stmt->fetch();

$customerData = ["metadata" => ["user_id" => $userId]];
if (!empty($user["stripe_customer_id"])) {
    $customerData["customer"] = $user["stripe_customer_id"];
} else {
    $customerData["customer_email"] = $_SESSION["user_email"];
```



```
}

$session = \Stripe\Checkout\Session::create(array_merge($customerData, [
    "mode" => "subscription",
    "payment_method_types" => ["card"],
    "line_items" => [[
        "price" => $priceId,
        "quantity" => 1,
    ]],
    "success_url" =>
    "https://parc.cyberethique.com/sub_success.php?session_id={CHECKOUT_SESSION_ID}",
    "cancel_url" => "https://parc.cyberethique.com/checkout.php",
    "metadata" => ["user_id" => $userId, "plan_id" => $planId],
    "subscription_data" => ["metadata" => ["user_id" => $userId]],
]));

header("Location: " . $session->url);
exit;
```

sub_success.php — Activation du plan

```
<?php
// sub_success.php
require 'vendor/autoload.php';
require 'stripe_config.php';
require 'db.php';
require 'auth.php';
requireLogin();

\Stripe\Stripe::setApiKey(STRIPE_SK);

$sessionId = $_GET["session_id"] ?? "";
$session = \Stripe\Checkout\Session::retrieve($sessionId);

if ($session->payment_status === "paid") {
    $userId = $session->metadata->user_id;
    $planId = $session->metadata->plan_id;
    $customerId = $session->customer;
    $subId = $session->subscription;

    // Récupérer la subscription pour la date de fin
    $sub = \Stripe\Subscription::retrieve($subId);
    $expiresAt = date("Y-m-d H:i:s", $sub->current_period_end);

    // Sauvegarder en BDD
    $pdo->prepare("UPDATE users SET
        plan_id = ?,
        plan_status = 'active',
        plan_expires_at = ?,
        stripe_customer_id = ?,
```

```
        stripe_sub_id      = ?
    WHERE id = ?")->execute([$planId, $expiresAt, $customerId, $subId, $userId]);

    // Notification interne
    $pdo->prepare("INSERT INTO user_notifications (user_id, title, message, type,
is_read, created_at)
        VALUES (?, ?, ?, 'success', 0, NOW())")
        ->execute([$userId, "Abonnement activé ! 🎉",
            "Votre abonnement est maintenant actif jusqu'au " . date("d/m/Y",
                strtotime($expiresAt))] );

    header("Location: dashboard.php?success=1");
    exit;
}
```

6. WEBHOOKS — ÉCOUTER LES ÉVÉNEMENTS STRIPE

Pourquoi les webhooks sont indispensables ?

Les webhooks permettent à Stripe de notifier votre serveur automatiquement à chaque événement (paiement reçu, abonnement renouvelé, échec de paiement...). Sans eux, votre BDD ne sera jamais mise à jour automatiquement.

Configuration dans le Dashboard Stripe

5. Stripe Dashboard → Développeurs → Webhooks → Ajouter un endpoint
6. URL : https://parc.cyberethique.com/stripe_webhook.php
7. Sélectionner les événements : checkout.session.completed, customer.subscription.updated, customer.subscription.deleted, invoice.payment_failed
8. Copier le "Signing Secret" (whsec_...) dans votre stripe_config.php

stripe_webhook.php — Votre écouteur d'événements

```
<?php
// stripe_webhook.php - NE PAS PROTÉGER PAR requireLogin() !
require 'vendor/autoload.php';
require 'stripe_config.php';
require 'db.php';

\Stripe\Stripe::setApiKey(STRIPE_SK);

$payload    = @file_get_contents("php://input");
$sigHeader  = $_SERVER["HTTP_STRIPE_SIGNATURE"] ?? "";

try {
    $event = \Stripe\Webhook::constructEvent(
        $payload, $sigHeader, STRIPE_WEBHOOK_SECRET
    );
} catch (\Exception $e) {
    http_response_code(400);
    exit("Webhook error: " . $e->getMessage());
}

// Router les événements
switch ($event->type) {

    // ☒ ABONNEMENT RENOUVELÉ
    case "invoice.payment_succeeded":
```

```
$sub = $event->data->object;
if ($sub->subscription) {
    $stripeSubId = $sub->subscription;
    $subscription = \Stripe\Subscription::retrieve($stripeSubId);
    $expiresAt = date("Y-m-d H:i:s", $subscription->current_period_end);
    $pdo->prepare("UPDATE users SET plan_expires_at=?,
plan_status='active'
                WHERE stripe_sub_id=?")
    ->execute([$expiresAt, $stripeSubId]);
}
break;

// ✕ ABONNEMENT ANNULÉ / EXPIRÉ
case "customer.subscription.deleted":
    $sub = $event->data->object;
    $pdo->prepare("UPDATE users SET plan_id=1, plan_status='expired'
                WHERE stripe_sub_id=?")
    ->execute([$sub->id]);
    break;

// ⚠ ÉCHEC DE PAIEMENT
case "invoice.payment_failed":
    $inv = $event->data->object;
    $customerId = $inv->customer;
    $user = $pdo->prepare("SELECT id FROM users WHERE stripe_customer_id=?");
    $user->execute([$customerId]);
    $row = $user->fetch();
    if ($row) {
        $pdo->prepare("INSERT INTO user_notifications
(user_id,title,message,type,is_read,created_at)
                VALUES (?,?,'alert',0,NOW())")
        ->execute([$row["id"],
                "⚠ Échec de paiement",
                "Votre renouvellement a échoué. Mettez vos infos à jour."]);
    }
    break;
}

http_response_code(200);
echo "OK";
```



7. MODE TEST & CARTES DE TEST

Cartes de test Stripe

En mode test, utilisez ces numéros de carte pour simuler différents scénarios :

Numéro de carte	Scénario	CVC / Date	Résultat
4242 4242 4242 4242	Paieement réussi	123 / 12/34	☑ Succès
4000 0000 0000 3220	Auth 3D Secure	123 / 12/34	☑ Succès avec 3DS
4000 0000 0000 0002	Carte refusée	123 / 12/34	✗ Déclinée
4000 0000 0000 9995	Fonds insuffisants	123 / 12/34	✗ Fonds insuff.

Tester les webhooks en local (Stripe CLI)

```
# 1. Installer Stripe CLI :  
brew install stripe/stripe-cli/stripe # macOS  
# OU télécharger sur https://stripe.com/docs/stripe-cli  
  
# 2. Se connecter :  
stripe login  
  
# 3. Écouter les webhooks en local :  
stripe listen --forward-to localhost/stripe_webhook.php  
  
# 4. Simuler un paiement réussi :  
stripe trigger checkout.session.completed  
  
# 5. Simuler un échec de paiement :  
stripe trigger invoice.payment_failed
```

8. PASSAGE EN PRODUCTION

Checklist avant de passer en production

☒ Checklist complète

- ☐ Compte Stripe activé avec informations bancaires complètes
- ☐ HTTPS actif sur votre domaine (obligatoire)
- ☐ Remplacer sk_test_ par sk_live_ dans stripe_config.php
- ☐ Remplacer pk_test_ par pk_live_ dans stripe_config.php
- ☐ Recréer les produits en mode live (les prix test ≠ prix live)
- ☐ Mettre à jour tous les Price ID en live
- ☐ Reconfigurer le webhook avec l'URL de prod et les bonnes clés
- ☐ Tester un vrai paiement de 1€ pour valider
- ☐ Activer la gestion des remboursements
- ☐ Configurer les emails de facture automatiques dans Stripe

Activer/Désactiver le mode test facilement

```
<?php
// stripe_config.php - Switcher facilement test/live

$mode = 'test'; // Changer en 'live' pour la production

if ($mode === 'live') {
    define('STRIPE_SK', 'sk_live_XXXXX');
    define('STRIPE_PK', 'pk_live_XXXXX');
    define('STRIPE_WEBHOOK_SECRET', 'whsec_LIVE_XXXXX');
} else {
    define('STRIPE_SK', 'sk_test_XXXXX');
    define('STRIPE_PK', 'pk_test_XXXXX');
    define('STRIPE_WEBHOOK_SECRET', 'whsec_TEST_XXXXX');
}
```

9. SÉCURITÉ & BONNES PRATIQUES

● Règles de sécurité absolues

1. JAMAIS mettre la clé secrète sk_... dans le JavaScript ou le HTML
2. TOUJOURS vérifier la signature des webhooks (Webhook::constructEvent)
3. TOUJOURS utiliser HTTPS en production
4. TOUJOURS traiter l'idempotence (ne pas insérer 2 fois le même paiement)
5. Stripe_config.php dans le .gitignore
6. Valider côté serveur le montant (ne jamais faire confiance au client)

Idempotence — Éviter les doublons en BDD

```
// Avant d'insérer un paiement, vérifier qu'il n'existe pas déjà
$check = $pdo->prepare("SELECT id FROM payments WHERE stripe_session_id = ?");
$check->execute([$sessionId]);

if (!$check->fetch()) {
    // Seulement là on insère
    $pdo->prepare("INSERT INTO payments (...) VALUES (...)")->execute([...]);
}
```



10. DÉPANNAGE & ERREURS COURANTES

Erreur	Cause probable	Solution
No such price	Price ID incorrect ou test/live mismatch	Vérifier Price ID dans Dashboard
Invalid API Key	Clé secrète incorrecte ou mauvais mode	Copier la clé depuis Developers > API Keys
Webhook signature invalid	Signing secret incorrect	Regénérer le secret dans Webhooks
This customer has no attached payment source	Pas de carte enregistrée	Utiliser Checkout (gère tout auto)
HTTP 400 sur webhook	Erreur dans votre code webhook	Regarder les logs PHP + Stripe Dashboard > Webhooks
Paieement réussi mais BDD non mise à jour	Webhook non configuré ou URL incorrecte	Vérifier URL et events dans Dashboard

Ressources utiles

- Documentation officielle : <https://stripe.com/docs>
- Référence API PHP : <https://stripe.com/docs/api?lang=php>
- Stripe CLI : <https://stripe.com/docs/stripe-cli>
- Tableau de bord test : <https://dashboard.stripe.com/test>
- Logs des webhooks : Dashboard → Développeurs → Webhooks → Logs



Vous êtes prêt !

Buy Me a Coffee + Abonnements sont maintenant configurés.

Bon courage pour la mise en production ! 