

ChatGPT

Offre gratuite

Partager ...

Nouveau chat

Rechercher des chats

Images

Applications

Codex

GPT

Website Ai Designer

Website Generator

Apob AI Guide 🤖

Explorer les GPT

Projets

Nouveau projet

Asterisk

Mr Bot Officiel

Vos chats

Sous-domaine pour Dokploy

Prénoms Français et Anglais

Analyse système Webylead

Déférence AWD et AWD Lock

A. Garçonnösey Free

Profiter de l'offre

infos-devis.md

Fichier

Je viens de parcourir ton fichier [infos-devis](#). Ce que tu as là, ce n'est pas juste une série de formulaires, c'est un mini-écosystème de "lead bridge" multi-sites avec deux types de destinations :

1. Dashboard Webylead (API directe)
2. Applicatif Webylead (bridge HTTP interne)
 - dans un cas, un routage supplémentaire vers Culligan

Demander à ChatGPT ne s'est construit par couches successives. Ça fonctionne, mais il y a de l'hétérogénéité. Et c'est là que ça devient intéressant.

Ce que j'observe stratégiquement

- Chaque site a sa propre logique cURL.
- Les validations sont répétées.
- Les payloads sont construits "à la main".
- Les envois bridge sont parfois bloquants, parfois timeout 1 seconde.
- Il n'y a pas de gestion d'erreur centralisée, sauf sur le premier site (le plus avancé).

En clair : tu as déjà un embryon d'architecture propre sur fontaine-a-eau.com, et des versions plus anciennes sur les autres.

La vraie question n'est pas "est-ce que ça marche", mais : comment rendre ça scalable, maintenable et auditable quand tu auras 20 sites au lieu de 6 ?

Vision d'évolution logique

1. Centraliser l'envoi HTTP

Tu as déjà une méthode `post_http()` propre sur le premier site. Elle devrait devenir un helper commun réutilisé partout.
2. Standardiser la structure de log

Seul le premier site loggue proprement en JSONL. Pour un analyste cybersécurité, tu sais que l'absence de logs exploitables, c'est le mal absolu.

Uniformiser :

- timestamp ISO8601
- target
- payload (avec masquage email/téléphone)
- résultat HTTP
- durée

Demander à ChatGPT payloads via mapping

Au lieu de :

```
$postData = [
  "Nom" => ...
  "Prénom" => ...
];
```

Tu pourrais définir un mapping déclaratif :

```
$mapping = [
  'nom' => 'Nom',
  'prenom' => 'Prénom',
  ...
];
```

Puis générer dynamiquement.

Résultat : plus de cohérence, moins d'erreurs humaines.

4. Séparer "métier" et "transport"

Aujourd'hui, la méthode `devis_form` :

- valide
- nettoie
- construit le payload
- appelle cURL
- echo OK

Demain, idéalement :

Controller → Service Lead → Adapter HTTP → Logger

Ca te permettrait :

Demander à ChatGPT

- de mockler les envois
- d'ajouter retry
- d'ajouter circuit breaker
- d'ajouter métriques

5. Timeout 1 seconde sur les bridges

C'est une optimisation intelligente pour ne pas bloquer l'utilisateur. Mais sans gestion d'erreur ni queue, tu peux perdre des leads silencieusement.

Une évolution moderne serait :

- pousser dans une queue (Redis, RabbitMQ, même fichier spool)
- worker asynchrone
- retry avec backoff exponentiel

Là tu passes d'un simple "form handler" à une petite plateforme robuste.

6. Sécurité et hygiène

Points positifs :

- `xss_clean` partout
- validation CI
- pas d'exposition directe des endpoints

Points à renforcer :

Point à revoir car :

- Masquage systématique des données sensibles dans les logs
- Ajout d'un hash d'intégrité dans les logs
- Limiter les domaines d'envoi via allowlist
- Ajouter un identifiant de corrélation par lead

Imagine que demain un client conteste un lead.
Tu dois pouvoir reconstruire toute la chaîne.

»» Demander à ChatGPT : ture devient stratégique, pas seulement technique.

Conclusion stratégique

Tu as déjà les briques.
Le site fontaine-a-eau est la version "mature".
Les autres sont des versions "legacy".

Si tu consolides tout dans une couche LeadBridge commune :

- Moins de duplication
- Plus de contrôle
- Moins de risques
- Plus de scalabilité

Ce n'est plus juste de l'intégration Webylead.
Ça devient une mini-infrastructure d'orchestration de leads.

Et dans un contexte futur où les APIs changent, où les partenaires imposent des webhooks signés, ou du OAuth, cette structuration fera toute la différence.

Si tu veux, on peut structurer ensemble un modèle d'architecture cible propre, prêt à supporter 50 sites sans dette technique explosive.

Sources



Je reprend tous les sites qui avait auparavant été codé en dur,
maintenant je fait une refonte sous wordpress. Quel formulaire utilisé et
comment envoyé à nouveau les infos saisi dans les formulaire comme
le faisait ce code ??

»» Demander à ChatGPT

Tu es en train de faire exactement ce qu'on devrait toujours faire : reprendre un système "codé en dur" et le faire entrer dans une architecture plus maintenable. Très bon mouvement stratégique.

Maintenant, question centrale :

Quel formulaire WordPress utiliser... et comment reproduire proprement tes envois vers Webylead / Bridge / Culligan ?

On va penser ça proprement, pas juste "plugin + copier-coller".

1 Quel plugin de formulaire choisir ?

Tu as deux approches sérieuses.

Option robuste et scalable

Gravity Forms

Pourquoi ?

- Hooks PHP natifs après soumission
- Gestion fine des champs
- Bonne stabilité
- Idéal si tu veux reproduire ta logique cURL personnalisée

C'est la solution "architecte".

Option gratuite et très flexible

Fluent Forms

»» Demander à ChatGPT :

- Webhook natif
- Log intégré
- Mapping facile des champs
- Plus moderne que Contact Form 7

Option simple mais limitée

Contact Form 7

Possible, mais nécessite du code pour les hooks.
Moins propre à long terme si tu multiplies les sites.

Si tu veux quelque chose de professionnel et évolutif :
Gravity Forms ou Fluent Forms.

2 Comment envoyer les données comme ton ancien code ?

Ton ancien système faisait :

1. Validation
2. Nettoyage
3. Construction d'un payload spécifique
4. Envoi cURL vers API

5. Log

Sous WordPress, on va faire pareil... mais proprement.

» Demander à ChatGPT propre (recommandée)**Étape 1 — Créer le formulaire normalement dans le plugin**

Exemple champs :

- nom
- prenom
- email
- telephone
- cp
- etc.

Étape 2 — Hook PHP après soumission

Avec Gravity Forms par exemple :

```
php Copier le code

add_action('gform_after_submission_1', 'send_to_webylead', 10, 2);

function send_to_webylead($entry, $form) {

    $payload = [
        'Nom' => rgar($entry, '1'),
        'Prénom' => rgar($entry, '2'),
        'Email' => rgar($entry, '3'),
        'Téléphone' => rgar($entry, '4'),
        'CP' => rgar($entry, '5'),
    ];

    $response = wp_remote_post(
        'https://dashboard.webylead.com/api/XXXX',
        [
            'body' => $payload,
            'timeout' => 15,
        ]
    );
}

WordPress utilise wp_remote_post() au lieu de cURL brut.
```

C'est plus propre, plus portable.

🔥 Encore mieux : utiliser les Webhooks natifs

Fluent Forms ou Gravity Forms permettent d'envoyer directement vers une URL API.

Tu mets :

- URL Webylead
- Format : x-www-form-urlencoded
- Mapping des champs

Zéro code.

Mais...

Dès que tu as une logique spéciale (ex: resolve city via IGN, construction d'un champ question, ajout de UTM fixes), le webhook simple ne suffit plus.

Et là tu reviens à la méthode avec hook PHP.

3 Reproduire ta logique avancée (exemple Culligan)**» Demander à ChatGPT**

- resolve city via API IGN
- construire question dynamique
- injecter UTM fixes
- envoyer avec referer

Ça sous WordPress devient :

1. Hook après soumission
2. Fonction resolve_city()
3. wp_remote_post()
4. Log custom dans wp-content/uploads/logs/

Exactement le même modèle... mais structuré en plugin personnalisé.

💡 Ce que je te conseille vraiment

Vu ton profil technique :

👉 Crée un mini plugin "LeadBridge" commun
Que tu installeras sur tous tes sites WordPress.

Ce plugin contient :

- Une classe LeadSender
- Méthode send_to_dashboard()
- Méthode send_to_bridge()
- Méthode resolve_city()
- Méthode log_event()

Chaque site aura juste :

- son endpoint API

son endpoint API
mapping champs → payload
Demandez à ChatGPT

ressources :

- 1 code maintenu
- 6 sites branchés dessus
- maintenance divisée par 6

4 Petit piège WordPress à anticiper

Par défaut :

- wp_remote_post a un timeout plus long
- erreurs silencieuses si pas gérées
- pas de retry

Tu devrais :

- vérifier wp_remote_retrieve_response_code()
- logger erreurs
- éventuellement ajouter retry simple

Sinon tu risques de perdre des leads sans le voir.

5 Résumé stratégique

Ancien système :

Chaque site = code isolé

Nouvelle vision WordPress :

Tous les sites → même moteur d'envoi centralisé

Tu ne fais pas juste une refonte graphique.

Tu es en train de transformer un ensemble de scripts en infrastructure cohérente.

Demandez à ChatGPT

tre un emplACEMENT de formulaires...
et un réseau de collecte de leads robuste.

Si tu veux, on peut concevoir ensemble la structure idéale du plugin LeadBridge pour que tu partes sur une base propre dès le premier site.

...

Oui on peut concevoir ensemble la structure idéale du plugin LeadBridge pour que tu partes sur une base propre dès le premier site.

NB: pour le formulaire wordpress, nous utiliserons fluent Form gratuit

Excellent. Là on ne parle plus d'un simple formulaire, on parle d'un **moteur d'orchestration de leads** sous WordPress.

Fluent Forms (version gratuite) va capturer les données.

Ton plugin **LeadBridge** va gérer la logique métier, la sécurité, les envois multiples et les logs.

On va faire ça proprement, modulaire, réutilisable sur 50 sites sans dette technique.

6 Philosophie d'architecture

Séparer clairement :

- Capture (Fluent Forms)
 - Transformation (mapping champs → payload partenaire)
 - Transport (wp_remote_post)
- Demandez à ChatGPT
- gs propres
 - Configuration (endpoints par site)

Ton plugin devient une couche d'abstraction entre WordPress et tes partenaires.

7 Structure idéale du plugin

Nom :

leadbridge

Structure :

arduino

leadbridge/
|
|— leadbridge.php
|— includes/
| |— class-leadbridge-core.php
| |— class-leadbridge-sender.php
| |— class-leadbridge-logger.php
| |— class-leadbridge-utils.php
|
|— config/
| |— endpoints.php

Copier le code

8 Fichier principal : leadbridge.php

Il charge les classes et initialise le système.

PHP

Copier le code

Demandez à ChatGPT

/**
 * Plugin Name: LeadBridge

```

 * Plugin Name: LeadBridge
 * Description: Orcheteur de leads multi-endpoints.
 * Version: 1.0
 */

if (!defined('ABSPATH')) exit;

require_once plugin_dir_path(__FILE__) . 'includes/class-leadbridge-core.php';

new LeadBridge_Core();

```

2 Core : Hook avec Fluent Forms

Fluent Forms déclenche :

```

php Copier le code

add_action('fluentform_submission_inserted', function($entryId, $formData, $form) {
    LeadBridge_Core::handle_submission($form->id, $formData);
}, 10, 3);

```

On ne dépend pas du nom des champs visuels.

On dépend des **slugs** Fluent Forms.

3 Configuration centralisée par site

Dans config/endpoints.php :

```

" Demander à ChatGPT
php Copier le code

Demandez à ChatGPT

return [
    1 -> [ // ID du formulaire Fluent
        'dashboard_url' -> 'https://dashboard.webylead.com/api/XXXX',
        'bridge_url' -> 'http://app.webylead.com/prospect/devis-XXX',

        'mapping' -> [
            'nom' -> 'Nom',
            'prenom' -> 'Prénom',
            'email' -> 'Email',
            'telephone' -> 'Téléphone',
            'cp' -> 'CP',
        ],
        'fixed_fields' -> [
            'utm_source' -> 'Affiliation'
        ]
    ]
];

```

Chaque site aura son propre fichier config.

Aucun code métier dupliqué.

4 Classe Sender (transport HTTP propre)

On utilise wp_remote_post.

```

php Copier le code

" Demander à ChatGPT
Sender {

    public static function send($url, $payload) {
        $response = wp_remote_post($url, [
            'body' => $payload,
            'timeout' => 15,
            'headers' => [
                'Content-Type' => 'application/x-www-form-urlencoded'
            ]
        ]);

        if (is_wp_error($response)) {
            return ['ok' -> false, 'error' -> $response->get_error_message()];
        }

        return [
            'ok' -> wp_remote_retrieve_response_code($response) >= 200
                && wp_remote_retrieve_response_code($response) < 300,
            'code' -> wp_remote_retrieve_response_code($response),
            'body' -> wp_remote_retrieve_body($response)
        ];
    }
}

```

5 Logger JSONL (comme ton meilleur site actuel)

Fichier :

wp-content/uploads/leadbridge.log

```

php Copier le code

" Demander à ChatGPT
Logger {

    public static function log($target, $payload, $result) {
        $record = [
            'ts' -> current_time('c'),
            'target' -> $target,
            'payload' -> $payload,
            'result' -> $result
        ];
    }
}

```

```

    'payload' => self::mask($payload),
    'result' => $result
);

$file = wp_upload_dir()['basedir'] . '/leadbridge.log';

file_put_contents(
    $file,
    json_encode($record, JSON_UNESCAPED_UNICODE) . PHP_EOL,
    FILE_APPEND | LOCK_EX
);
}

private static function mask($payload) {
    if (isset($payload['Email'])) {
        $payload['Email'] = substr($payload['Email'], 0, 3) . '***';
    }
    return $payload;
}
}

```

Traçabilité = puissance.

6 Gestion spéciale (ex : resolve_city IGN)

[dige-utils.php](#)

» Demander à ChatGPT

Même logique que ton ancien code, mais isolée :

```

php
class LeadBridge_Utils {

    public static function resolve_city($cp) {

        if (!preg_match('/^\d{5}$/', $cp)) {
            return 'Non précisée';
        }

        $response = wp_remote_get(
            'https://apicarto.ign.fr/api/codes-postaux/communes/' . $cp,
            ['timeout' => 8]
        );

        if (is_wp_error($response)) return 'Non précisée';

        $data = json_decode(wp_remote_retrieve_body($response), true);

        return $data[0]['nomCommune'] ?? 'Non précisée';
    }
}

```

[Copier le code](#)

7 Sécurité minimale obligatoire

Ajoute :

- Honeypot Fluent Forms
- reCAPTCHA

» Demander à ChatGPT visions par IP

- Validation stricte email/téléphone
- Vérification domaine referrer si nécessaire

Un lead system attire les bots comme le miel attire les fourmis.

8 Vision long terme

Tu peux ajouter plus tard :

- Retry automatique si HTTP != 200
- File d'attente WP-Cron
- Interface admin pour voir les erreurs
- Envoi Slack en cas d'échec
- Signature HMAC des payloads
- Identifiant unique par lead

À ce moment-là, tu n'as plus un plugin.

Tu as une mini-plateforme de routing.

9 Résumé stratégique

Fluent Forms = interface utilisateur

» Demander à ChatGPT ↴

endpoints = configuration

Séparation claire.

Maintenance simple.

Scalabilité naturelle.

Logs exploitables.

Tu passes d'un patchwork PHP à une architecture cohérente.

Prochaine étape logique :

On écrit ensemble la version complète prête à copier-coller pour ton premier site.

[Copier](#) [Partager](#) [Imprimer](#) [Envoyer](#) [...](#)

Is this conversation helpful so far? [Upvote](#) [Downvote](#) [Close](#)

+ Poser une question



ChatGPT peut commettre des erreurs. Il est recommandé de vérifier les informations importantes. Voir les [préférences en matière de cookies](#).