ChatGPT

Plateforme TravauxLinks.fr – Cahier des Charges et Spécifications

Introduction

TravauxLinks.fr est une plateforme web visant à mettre en relation des particuliers ayant des projets de travaux avec des artisans qualifiés, tout en intégrant un espace dédié aux aides de rénovation énergétique (reprise du contenu du site *EnvironnementCEE* sous la section `/energie`). L'objectif est de fournir une application moderne, sécurisée et mobile-first qui inspire confiance aux utilisateurs et maximise la conversion. La plateforme se compose de trois espaces principaux : un portail client pour le dépôt de projets (sans création de compte), un espace artisans (avec authentification obligatoire) et un back-office admin (CRM interne). Le site devra également intégrer les contenus existants d'EnvironnementCEE (simulateur d'économies d'énergie, formulaires d'aide, etc.) afin de proposer une expérience unifiée.

1. Authentification des artisans et profils complets

Inscription et profil des artisans : Seuls les artisans peuvent créer un compte sur la plateforme (les clients n'ont pas besoin de compte). L'inscription artisan est obligatoire et doit permettre de constituer un **profil professionnel complet** comportant :
- Les **métiers/compétences** de l'artisan (ex. plombier, électricien, maçon…),
- La **zone d'intervention** (départements ou régions couverts),
- Les éventuelles **certifications RGE** (Reconnu Garant de l'Environnement) détenues,
- L'upload de **documents justificatifs** (assurance décennale, certification RGE, etc.),
- Des informations de contact (téléphone, adresse, SIRET…) qui ne seront toutefois visibles aux clients qu'après mise en relation effective (voir section Sécurité).

Certification RGE : Il est important de stocker et afficher si un artisan est certifié RGE, car cela garantit son engagement qualité et conditionne l'accès aux aides publiques pour les clients [1]. En effet, *« la mention RGE (…) est une condition pour que vos clients obtiennent les aides publiques allouées aux travaux de performance énergétique »* [2]. Le profil artisan devrait donc indiquer les domaines RGE obtenus le cas échéant, afin de rassurer les particuliers et de filtrer les artisans pour les projets de rénovation énergétique.

Pas de compte client requis : Les particuliers utilisateurs du site n'ont pas à s'inscrire. Leur parcours est simplifié : ils remplissent un formulaire de projet et recevront ensuite des contacts d'artisans par email/téléphone. Cette absence de friction côté client vise à maximiser le taux de conversion du dépôt de projet.

2. Formulaire de dépôt de projet (client) et matching des artisans

Formulaire en trois étapes : La page ***/deposer-projet*** proposera un formulaire guidé en 3 étapes pour collecter toutes les informations nécessaires sur le projet du particulier. Par exemple : *(1) Type de travaux et description, (2) Localisation du chantier, (3) Coordonnées du client.* Ce découpage rend le processus moins intimidant et améliore le taux d'achèvement du formulaire.

Matching V1 métier + zone : À la soumission d'un projet, l'application effectue un **matching** basique : identification des artisans dont le **métier** correspond au type de travaux demandé et dont la **zone géographique** d'intervention couvre l'adresse du projet. Ce matching V1 utilise les données du profil artisan (métiers, zones) pour sélectionner les professionnels pertinents.

Transmission des leads : Une fois les artisans correspondant identifiés, le **lead** (demande de travaux) est automatiquement transmis aux artisans concernés. Concrètement, cela peut générer : - Une **notification email via Resend** aux artisans éligibles, contenant les détails du projet (sauf les coordonnées du client, masquées tant que l'artisan n'a pas manifesté d'intérêt – voir Sécurité).
- L'apparition du nouveau lead dans le **dashboard artisan** (voir section suivante) avec un statut "Nouveau".

Chaque lead inclut un identifiant, les informations du projet (type, description, budget éventuel, localisation approximative), et permet à l'artisan de prendre une décision (intéressé ou non).

3. Dashboard Artisan – Gestion des leads et chantiers

Vue d'ensemble des leads : L'espace **/artisans/dashboard** offre à chaque artisan une liste de tous les leads qui lui ont été envoyés, avec leur statut. Pour chaque **fiche lead**, l'artisan peut voir : le type de travaux, la description fournie, la localisation (commune, code postal), et éventuellement un indice sur l'urgence ou l'échéance du projet. Les **coordonnées directes du client sont masquées** tant que le lead n'a pas été attribué (voir Sécurité).

Actions sur les leads : Pour chaque lead, l'artisan a la possibilité de réaliser plusieurs actions :
- **"Intéressé"** : marquer qu'il souhaite contacter ce client. À ce stade, le système pourra révéler les coordonnées (si l'artisan est le premier ou fait partie des sélectionnés) et éventuellement débiter un crédit/monétisation si en place. Le statut du lead passe à "En cours – artisan intéressé".
- **"Proposer un devis"** : l'artisan peut directement téléverser un devis PDF ou indiquer qu'un devis a été envoyé au client. Ces documents sont stockés dans la table **documents** (liés au lead ou au chantier). Le statut passe à "Devis proposé".
- **"Refuser"** : indiquer qu'il n'est pas intéressé par ce projet. Le lead passe alors en statut "Refusé par l'artisan". Cela permettra au système d'évaluer le lead (peut-être à reproposer à d'autres artisans si personne n'en veut).
- **Conversion en chantier :** si le contact aboutit et que l'artisan réalise effectivement les travaux, le lead est converti en **"chantier"**. L'artisan peut alors marquer le lead comme **gagné/chantier démarré**, et l'associer éventuellement à un enregistrement dans la table **chantiers** (pour suivi de l'historique des travaux effectués).

Suivi des chantiers et documents : L'artisan a également une section listant ses **chantiers en cours ou terminés**, avec possibilité d'uploader des documents (devis signés, factures, photos avant/après, etc.). Ces documents seront stockés dans la table *documents* et rattachés soit aux leads soit aux chantiers correspondants. Cela permet de conserver un historique complet et éventuellement de le partager avec l'administrateur ou le client si besoin.

4. Espace admin – CRM de suivi et pilotage

L'application comprend un back-office **/admin** réservé aux équipes internes, jouant le rôle de CRM pour le suivi global de l'activité :

- **Vue globale des leads :** liste de tous les leads soumis par les clients, avec leurs statuts (en attente, en cours, attribué, conclu, abandonné…). L'admin peut filtrer par date, par zone, par métier, par statut, etc.
- **Assignation manuelle d'artisans :** en plus du matching automatique, l'administrateur doit pouvoir **assigner ou suggérer manuellement un artisan** à un lead. Par exemple, si aucun artisan n'a répondu à un lead, l'admin peut décider de l'attribuer directement à un artisan particulier (avec son accord téléphonique préalable) puis de lui débloquer les coordonnées. L'admin peut aussi retirer ou ajouter des artisans assignés à un lead.
- **Modification des statuts :** si un lead est finalement conclu (le particulier a trouvé son pro via la plateforme), l'admin peut marquer le lead comme *chantier démarré* ou *chantier terminé*. Inversement, si un lead n'aboutit pas (ex : client non joignable, projet annulé), l'admin peut le clore comme *abandonné* ou *sans suite*.
- **Scoring et qualité :** le back-office affichera des **scores** (voir section Scoring) pour chaque lead, chaque client et chaque artisan, permettant à l'équipe de repérer d'éventuels problèmes (ex : clients peu sérieux, artisans peu réactifs). L'admin peut ainsi filtrer ou contacter les utilisateurs en fonction de ces indicateurs.
- **Historique et timeline :** pour chaque lead et chaque artisan, l'admin a accès à une **timeline des événements** (stockée dans `events_timeline`) retraçant les actions : création du lead, notifications envoyées, artisan X intéressé le [date], devis uploadé, etc. Ceci permet un support client efficace et une traçabilité (utile aussi pour la RGPD).

Cet espace admin servira de **tour de contrôle** pour optimiser le taux de conversion des leads en chantiers, équilibrer l'offre et la demande (nombre d'artisans par lead), et assurer la qualité du service.

5. Intégration du site EnvironnementCEE dans `/energie`

Le site existant **environnementcee.fr** sera intégré dans la nouvelle plateforme sous la route ***/energie***, afin de centraliser les contenus et outils liés aux aides énergétiques. L'objectif est de **migrer tout le contenu existant sans perte** en veillant à la continuité du référencement SEO.

Contenus à migrer :
- Pages d'information sur les dispositifs d'aide à la rénovation énergétique (par ex, explication des CEE – Certificats d'Économies d'Énergie, primes disponibles, etc.). Ces pages deviendront par exemple ***/energie/les-aides***, ***/energie/cee***, etc., en conservant le texte, images et liens.
- Un **simulateur d'économie d'énergie (simulateur LED)** qui permet aux utilisateurs d'estimer les économies réalisables en remplaçant des éclairages par des LED. Ce simulateur, s'il existe en version web ou iframe, devra être intégré dans une page ***/energie/simulateur-led***.
- Un **formulaire de demande d'aide** (éventuellement lié aux CEE ou à des offres partenaires) présent sur environnementcee.fr. Celui-ci sera placé sous ***/energie/demande-aide*** et devra conserver son fonctionnement (envoi d'un email ou sauvegarde en base des données saisies).
- Les pages légales et obligatoires : **Mentions légales**, **Politique de confidentialité**, éventuellement **CGU/CGV**. Celles-ci peuvent être communes à TravauxLinks et EnvironnementCEE, mais accessibles depuis la section /energie pour l'utilisateur venant chercher une information sur les aides.
- Tout autre contenu (actualités, FAQ, articles) du site EnvironnementCEE devra soit être rapatrié dans /energie, soit redirigé.

Canonical & SEO pour contenu dupliqué : Si le domaine *environnementcee.fr* reste actif pendant une période de transition, il est crucial d'éviter le **duplicate content**. On utilisera des balises `<link rel="canonical">` appropriées pour indiquer la version canonique du contenu. En pratique, les pages sous `/energie/...` porteront une balise canonique pointant vers elles-mêmes (ou éventuellement vers les URL environnementcee.fr si on souhaite conserver celles-ci comme références principales). **L'utilisation de la balise canonique permet d'éviter les problèmes de contenu dupliqué** en signalant aux moteurs quelle URL indexer lorsqu'un même contenu est accessible à plusieurs adresses [3]. *« En synthèse, la balise canonical indique aux moteurs de recherche quelle version d'une URL vous souhaitez voir apparaître dans les résultats, donc la page canonique »* [4]. Ainsi, le référencement acquis du site existant ne sera pas perdu et Google comprendra l'intégration des deux sites. Par ailleurs, on veillera à configurer des redirections 301 depuis les anciennes URL vers les nouvelles (ou inversement) selon la stratégie retenue, afin de ne pas impacter l'expérience utilisateur.

En somme, la section **Énergie** du nouveau site reprendra l'intégralité des fonctionnalités et informations d'EnvironnementCEE, offrant aux particuliers un guichet unique : trouver un artisan pour ses travaux et s'informer sur les aides financières disponibles.

6. LeadRisk Engine – Relance des leads inactifs

Pour optimiser le traitement des demandes, un module **LeadRisk Engine** sera implémenté. Son rôle : détecter les leads qui **demeurent inactifs pendant plus de 5 jours** et engager des actions pour éviter la déperdition.

Détection d'inactivité : Un lead est considéré comme inactif si, 5 jours après sa création, **aucun artisan ne l'a marqué comme "intéressé"** et qu'aucun devis n'a été initié. Ces leads seront automatiquement **flaggés** par le système (par ex. champ `lead_risk = true` dans la table `leads`).

Relance automatique : Une fois flaggé, le lead inactif déclenche :
- **Envoi d'un email de relance** au(x) artisan(s) notifiés initialement, pour leur rappeler la demande en attente et les inciter à y répondre s'ils sont intéressés.
- Éventuellement, **notification à l'administrateur** dans le CRM, afin qu'il puisse décider d'actions manuelles (contacter d'autres artisans, appeler le client pour affiner le besoin, etc.).

Baisse du prix du lead : Si un modèle économique de **vente de leads** aux artisans est en place, on peut intégrer une fonction de **décote progressive** : par exemple, après 5 jours sans preneur, le coût d'achat du lead pour les artisans est abaissé (par ex -20%). Cette mesure incitative vise à augmenter l'attrait du lead "âgé" et à lui trouver preneur. Cela s'inspire de pratiques de certains marchés où les leads anciens sont soldés pour être monétisés plutôt que perdus.

Importance de la réactivité : Ce moteur de relance se justifie par l'importance critique de la **vitesse de réponse sur la conversion**. Des études montrent qu'un délai de seulement 5 minutes dans le suivi d'un lead peut réduire de 80% les chances de le qualifier [5]. Autrement dit, plus un particulier reste longtemps sans réponse, plus il a de chances d'aller voir ailleurs. Réagir sous 24-48h est donc primordial. Le LeadRisk Engine agit comme un **filet de sécurité** pour éviter qu'aucune demande ne tombe "dans les mailles du filet" sans réponse. En relançant rapidement, on améliore l'expérience client et le taux de transformation des leads.

7. Scoring automatisé des clients et des artisans

La plateforme mettra en place un système de **scoring** pour évaluer la qualité et la fiabilité des deux côtés du marché :

- **Score client :** calculé de façon automatisée en fonction du comportement du particulier. Par exemple : un client qui dépose un projet puis ne répond jamais aux appels des artisans aura un score plus faible. À l'inverse, un client qui valide un devis rapidement ou qui dépose plusieurs projets aboutissant à des chantiers aura un score élevé. Des critères possibles : *taux de réponse aux artisans, délais de réponse moyens, taux de conversion de ses leads en chantiers*. Ce score aide à détecter les leads "à risque" (ex : faux numéro, démarcheur commercial, etc.) afin d'en informer les artisans ou l'administration.

- **Score artisan :** évalue la performance et le sérieux de l'artisan. Critères potentiels : *taux de réponse aux leads reçus, rapidité de prise de contact, nombre de devis fournis, taux de réussite (devis acceptés), avis/satisfaction des clients après chantier, respect des délais*. Un artisan très réactif, proposant des devis pertinents et obtenant de bons retours clients verra son score augmenter. Au contraire, des refus systématiques ou une absence de réponse aux leads feront baisser le score. Ce score pourra être visible par l'admin, voire communiquer partiellement aux clients sous forme d'indicateurs de confiance.

Utilisation des scores : Ces scores ont plusieurs utilités :
- Aider l'algorithme de matching futur (v2) : on pourrait prioriser l'envoi d'un lead à des artisans bien notés, pour maximiser la satisfaction du client.
- Afficher des badges de **"Artisan de confiance"** ou de **"Top Artisans"** pour ceux qui ont un score élevé, afin de rassurer les clients. De même, l'admin pourrait filtrer/suspendre temporairement des artisans dont le score tombe trop bas (signe de problèmes).
- Côté artisans, visualiser le score d'un lead client pourrait les aider à estimer la probabilité de concrétisation.

Inspiration des marketplaces : Ce principe s'inspire des systèmes de notation bi-directionnelle que l'on retrouve sur Uber, Airbnb et autres plateformes de mise en relation où **les deux parties s'évaluent**. Par exemple, Uber utilise les étoiles données par les passagers et conducteurs pour exclure proactivement les mauvais acteurs des deux côtés et maintenir un haut niveau de confiance [6] [7]. *« La plupart des places de marché populaires ont un système de notation quelque part dans le processus, pour les acheteurs et les vendeurs… Ces notes sont utilisées de façon quasi-invisible pour écarter les mauvais utilisateurs et améliorer le service »* [6]. De même, sur TravauxLinks, le scoring vise à **instaurer un climat de confiance** et à **améliorer en continu la qualité** du réseau : les particuliers sauront que les artisans actifs sont réactifs et bien évalués, et les artisans auront davantage confiance en les leads provenant de clients sérieux.

La mécanique exacte de calcul sera à affiner, mais l'important est qu'elle soit **automatisée** (cron ou fonctions Supabase) et mise à jour au fil des événements (chaque interaction pertinente dans `events_timeline` peut ajuster un score).

8. Envoi des emails via Resend (Transactional Email)

L'application enverra de nombreux emails transactionnels : accusés de réception aux clients, notifications de lead aux artisans, relances, réinitialisation de mot de passe, etc. La solution retenue est **Resend**, un service d'email pour développeurs, via son API.

Deux expéditeurs distincts : On utilisera deux adresses d'envoi différentes, correspondant aux deux volets de la plateforme :
- `contact@travauxlinks.fr` pour tous les emails liés aux travaux (demandes de devis, notifications artisans, communications générales plateforme travaux).
- `contact@environnementcee.fr` pour les emails spécifiques à la partie aides énergie (confirmation de demande d'aide, newsletter énergie, etc.).

Ces deux domaines seront **configurés sur Resend** en tant que domaines autorisés. Il faudra donc vérifier ces domaines en ajoutant les enregistrements DNS requis (SPF, DKIM) afin que Resend puisse envoyer les emails en notre nom [8]. *« Resend envoie les emails via un domaine que vous possédez… pour vérifier un domaine, il faut ajouter deux entrées DNS (SPF et DKIM) »* [8]. Une fois les domaines `travauxlinks.fr` et `environnementcee.fr` vérifiés sur Resend, on pourra émettre des emails avec ces expéditeurs.

Gabarits d'emails et envois : On développera des gabarits d'emails personnalisés (HTML responsive, reprenant le logo et les couleurs bleu/vert de la marque) pour chaque type de notification : - **Au client :** confirmation de dépôt de projet, informations sur la suite (ex: "Nous recherchons des artisans disponibles…") ; éventuellement des conseils sur comment bien choisir son artisan.
- **À l'artisan :** nouveau lead reçu (avec lien vers son dashboard), rappel de lead (relance LeadRisk), notification qu'un client a accepté son devis ou répondu via la plateforme.
- **À l'admin :** alerte nouveau lead créé, rapports d'activité hebdo, etc.
- **Relances automatiques :** par ex, email au client 7 jours après dépôt pour évaluer son satisfaction s'il n'y a pas eu de suite, ou incitation à répondre si un artisan a fait une offre.

Tous ces envois se feront via l'API Resend (appel depuis le backend – via une fonction serverless ou directement depuis la logique Lovable/Supabase). Resend offre des SDK pour Node.js et autres, facilitant l'envoi. Nous pourrons utiliser des **modèles d'emails** enregistrés dans Resend pour y injecter les variables (nom du client, détails du projet, etc.), ce qui permet de centraliser la mise en page des emails.

Suivi des emails : Resend fournit un tableau de bord pour voir les emails envoyés, les taux d'ouverture, les erreurs éventuelles (bounces). Cela sera utile pour la QA et le support (ex: détecter si un artisan ne reçoit pas ses emails).

9. Stack technique et déploiement (Lovable + Supabase + Vercel)

La solution sera développée en utilisant une **stack moderne** alliant rapidité de prototypage et robustesse :
- **Lovable** : utilisation de la plateforme **Lovable AI** pour générer l'application web full-stack initiale. Lovable permet de créer des applications en langage naturel, avec du code exportable et modifiable [9]. *« Lovable AI est une plateforme de développement qui permet de générer des applications full-stack à partir de simples instructions en anglais, produisant du code réel pour le frontend, le backend et la base de données »* [9]. Concrètement, on exploitera Lovable pour obtenir un squelette d'application (probablement basé sur Next.js ou une stack Node) relié à Supabase. Cela accélérera la mise en place des pages, de l'authentification et des APIs de base. Le code généré sera ensuite affiné manuellement pour répondre aux spécifications précises (Lovable étant un point de départ rapide, mais nécessitant des ajustements pour un produit fini).
- **Supabase** : choisi comme backend **BaaS** (Backend-as-a-Service) couvrant la base de données PostgreSQL, l'authentification utilisateur et les **Fonctions Edge** au besoin. Supabase est en quelque sorte une alternative open-source à Firebase, offrant un Postgres avec API instantanée et abonnement

en temps réel [10]. Il fournit un **système d'authentification** out-of-the-box (emails, OAuth…) que l'on utilisera pour gérer les comptes artisans et admin (les clients n'ayant pas de compte, ils ne passent pas par Auth). Surtout, Supabase permet d'implémenter facilement des **politiques de sécurité row-level (RLS)** sur les tables pour restreindre l'accès aux données sensibles côté client. **Il est impératif d'activer RLS sur toutes les tables exposées**, comme le rappelle la documentation Supabase [11]. *« Supabase permet un accès pratique et sécurisé aux données depuis le navigateur, à condition d'activer RLS. RLS doit toujours être activé sur les tables dans le schéma public exposé… »* [11]. Nous définirons donc des policies pour que : chaque artisan ne puisse voir que ses leads, chaque admin ait un accès élargi, etc. Supabase offrira également des fonctionnalités utiles : stockage de fichiers (pour les documents PDF, photos…), fonctions serverless (ex: déclencher automatiquement la relance LeadRisk via un cron), et un **dashboard d'administration de la base** bien pratique.
- **Vercel** : la plateforme de déploiement **Vercel** sera utilisée pour héberger le frontend (et backend serverless s'il y a, typiquement Next.js API routes ou Edge Functions). Vercel est optimisé pour les applications React/Next.js et permettra un déploiement continu (CI/CD) de l'application Lovable exportée. L'avantage est la facilité de mise en échelle, le support des domaines custom (`travauxlinks.fr`, `environnementcee.fr`) et les performances via leur CDN. De plus, Vercel gère le SSL automatiquement et offre des previews pour tester les branches de développement.
- **Resend** : comme déjà abordé, pour l'envoi des emails transactionnels via l'API.

Autres choix techniques : Lovable générant un code *"full-stack app"*, il est probable que le framework soit basé sur Next.js ou Remix (vu l'orientation Vercel). On s'assurera que l'app est **SPA** responsive (React) avec un rendu côté serveur pour le SEO des pages publiques. Supabase fournira la couche de persistance (Postgres) et l'auth, évitant d'avoir à coder un backend Node custom pour la plupart des besoins – les appels de la partie frontend se feront directement aux APIs Supabase sécurisées (policies) ou via les hooks/SDK fournis.

Cette stack a été choisie pour sa **rapidité de développement** (Lovable permet un MVP en quelques heures/days [9]), sa **scalabilité** (Supabase+Vercel gèrent la montée en charge sans effort) et son **coût maîtrisé** (Supabase a un plan gratuit généreux, Vercel aussi pour projets modestes, Lovable évite d'embaucher une armée de développeurs dès le départ). L'objectif est de livrer un MVP fonctionnel très rapidement, tout en posant des bases solides pour évoluer par la suite.

10. Conception UI/UX : mobile-first, moderne et inspirant confiance

La plateforme sera conçue selon les principes **mobile-first** : cela signifie que le design des pages sera d'abord pensé pour les écrans mobiles (smartphones), puis enrichi pour les plus grands formats. Étant donné que beaucoup de particuliers naviguent sur mobile lorsqu'ils recherchent un artisan, c'est primordial. On veillera à une navigation simple (menus clairs, appels à l'action bien visibles) et à ce que toutes les fonctionnalités (formulaire multi-étapes, tableau de bord artisan) soient utilisables aisément sur un petit écran.

Design épuré et moderne : L'interface utilisera une **charte graphique** sobre mettant en avant la **clarté de l'information**. L'accent est mis sur la simplicité : fonds clairs, typographie lisible, pictogrammes explicites. L'aspect moderne rassurera l'utilisateur sur le sérieux et la qualité du service.

Couleurs et psychologie : Les couleurs dominantes seront :
- **Bleu professionnel** – symbole de confiance, de sécurité et de sérieux. Le bleu est couramment utilisé dans les services professionnels car il inspire un sentiment de fiabilité aux utilisateurs [12]. *« Le bleu symbolise la confiance… il est souvent associé à la stabilité et la fiabilité »* [12]. Cela conviendra parfaitement

pour instaurer la confiance envers la plateforme (notamment vis-à-vis des particuliers confiant leurs coordonnées et projets).
- **Vert énergie** – évoquant l'écologie, l'économie d'énergie et la rénovation verte. *« Le vert évoque la nature, la fraîcheur... Il est souvent utilisé pour promouvoir des produits écologiques »* [13]. Ce vert sera utilisé surtout dans la section /energie et pour mettre en avant les aspects d'aide éco-responsable (économies d'énergie, CEE). Il suggère aussi l'espoir et le positif, ce qui est adéquat pour parler de rénovation et d'amélioration de l'habitat.

Ces couleurs seront probablement utilisées par touches (boutons, en-têtes, icônes) sur un fond majoritairement blanc ou gris très clair, afin de conserver une apparence lumineuse et aérée. L'association bleu + vert doit être faite avec goût : le bleu pour le côté *service travaux pro*, le vert pour le côté *aides énergétiques*, les deux se mariant de façon harmonieuse pour l'identité globale.

Confiance et conversion : Chaque élément de design cherchera à **mettre l'utilisateur en confiance** et à **optimiser la conversion**. Cela passe par :
- Des **messages de réassurance** sur la page d'accueil (*"Réseau d'artisans vérifiés", "Vos données protégées", "+1000 chantiers réalisés"* etc. – chiffres et labels à ajuster en fonction de la réalité).
- Des **CTA (call-to-action) bien visibles** pour déposer un projet, avec une accroche claire (« Obtenez jusqu'à 5 devis gratuits » par ex).
- Un ton et un contenu **orientés client**, pédagogiques dans la section énergie, incitant à passer à l'action (par ex sur les aides : « Profitez des primes énergie avant qu'elles ne baissent »).
- Une **navigation fluide** : peu de pop-ups intrusifs, un menu compact, une hiérarchie visuelle évidente (titres, sous-titres, boutons).

Enfin, on respectera les bonnes pratiques d'**accessibilité** (contrastes suffisants, police >= 16px, labels sur les formulaires, etc.) afin d'offrir une expérience agréable au plus grand nombre.

11. Sécurité des données et conformité RGPD

La sécurité et la protection des données personnelles seront traitées avec le plus grand sérieux, d'autant plus que la plateforme manipule des coordonnées privées et potentiellement des documents confidentiels (devis, etc.).

Row-Level Security (RLS) sur la base : Comme évoqué, toutes les données sensibles stockées dans Supabase seront protégées par des **politiques RLS**. Par exemple : un artisan authentifié ne pourra sélectionner que les leads qui lui sont destinés (`lead_artisan_matches` liant son user_id), un artisan ne verra pas les données d'un autre artisan, etc. Par défaut, tant qu'un utilisateur n'est pas authentifié, aucune donnée n'est retournée (policies basées sur `auth.uid()` fournies par Supabase). **Aucune donnée client ou lead ne sera accessible via l'API publique sans vérification d'autorisation**, assurant une **défense en profondeur** côté base de données [11].

Masquage des coordonnées clients : Pour préserver la confidentialité du particulier et éviter tout démarchage intempestif, la plateforme **masque les coordonnées** (téléphone, email, adresse précise) du client dans les leads, tant qu'une relation de confiance n'est pas établie :
- Dans le dashboard artisan, le lead sera affiché sans les coordonnées tant que l'artisan n'a pas cliqué "Intéressé" ET que le lead ne lui est pas **attribué** (on peut imaginer que si plusieurs artisans cliquent "Intéressé", l'admin ou le système attribue le lead à 2 artisans max, ceux-ci recevront alors les contacts complets).
- Une fois un lead attribué à un artisan, cet artisan peut voir les infos du client pour le contacter. Les autres artisans non choisis ne les verront jamais.

- Ce fonctionnement protège également le client d'une sur-sollicitation : il ne sera contacté que par un nombre restreint d'artisans choisis, et non par tous ceux initialement notifiés.

RGPD : Le site sera **100% conforme RGPD** :
- Affichage d'une **bannière cookies** si des trackers ou cookies non essentiels sont utilisés, avec recueil du consentement.
- Mise à disposition d'une **Politique de confidentialité** détaillant les données collectées (formulaires de projet, compte artisan), la finalité (mise en relation travaux, obtention d'aides), la base légale (consentement du client pour être contacté par des artisans, intérêt légitime pour amélioration du service, etc.), la durée de conservation, et les droits des personnes (accès, rectification, suppression…).
- **Formulaire de contact RGPD** ou adresse email dédiée pour que les utilisateurs puissent exercer leurs droits (ex: *contact@travauxlinks.fr* pour demandes RGPD). Supabase facilitera l'extraction/suppression des données sur demande.
- Hébergement des données dans l'UE (Supabase propose des hébergeurs européens normalement, à vérifier pour conformité).
- Journalisation des accès et actions (la table `events_timeline` joue aussi un rôle de **log d'activité** qui peut être utile en cas de violation ou de demande d'audit).

Sécurité applicative : Outre la base de données, on mettra en place :
- **HTTPS obligatoire** sur toutes les pages (Vercel fournit SSL par défaut).
- **Protection XSS/CSRF** assurée en grande partie par le framework (Next.js a des protections, Supabase API aussi). Les formulaires sensibles auront des tokens anti-CSRF si nécessaire.
- **Mots de passe** : Supabase Auth s'occupe du hashing sécurisé des passwords (Bcrypt) et peut imposer une politique (longueur minimale). Possibilité d'activer la 2FA via Supabase si on le souhaite à terme pour les artisans.
- **Limiter les données stockées** : on n'enregistrera pas de données superflues. Les documents uploadés seront contrôlés (limiter types MIME autorisés pour éviter qu'un artisan n'upload un .exe malveillant par exemple).

En résumé, la plateforme traitera les données personnelles avec rigueur, en ne donnant accès aux informations qu'à ceux qui en ont besoin, et en étant transparente vis-à-vis des utilisateurs sur l'usage de leurs données.

12. SEO : optimisation du référencement naturel

Dès le lancement, il faut que le site soit **bien indexé** par les moteurs de recherche pour attirer du trafic organique (particulièrement sur la partie contenu /energie et éventuellement sur des pages comme "Trouver un artisan [métier] [ville]"). Les bonnes pratiques SEO suivantes seront appliquées :

- **Sitemap XML dynamique** : Une sitemap.xml listant toutes les pages importantes sera générée et mise à jour automatiquement à chaque déploiement. Cela inclura les pages statiques (accueil, énergie, mentions…) et éventuellement des URL dynamiques si on propose un annuaire d'artisans ou de métiers. *« Un sitemap XML est un document listant toutes les pages d'un site pour fournir aux moteurs un aperçu du contenu disponible »* [14]. On veillera à n'y inclure **que les pages indexables** et à le tenir à jour [15] (par exemple en l'actualisant via un script ou un plugin à chaque modification de contenu). Ce fichier sera déclaré dans le fichier robots.txt et via la Google Search Console pour accélérer l'indexation.

- **Balises canoniques** : Comme vu, mise en place de `<link rel="canonical">` sur les pages de la section énergie pour éviter le contenu dupliqué entre l'ancien et le nouveau site [3].

Chaque page aura soit une self-canonical (pointant sur elle-même si contenu unique), soit pointera vers l'URL de référence si le même contenu existe sur deux domaines.

- **Métadonnées (title, description) soignées** : Chaque page aura une **balise `<title>` unique et descriptive**, ainsi qu'une **meta description** optimisée. Par exemple la page d'accueil TravauxLinks aura un title du type *"TravauxLinks – Trouvez des artisans qualifiés pour vos travaux"* et une description incluant des mots-clés (travaux, devis, artisans de confiance, rénovation énergétique…). Idem pour les pages /energie qui décriront clairement le contenu (ex: *"Aides à la rénovation énergétique – primes CEE et conseils | TravauxLinks Énergie"*). Il est impératif que **toutes les pages indexables aient un titre et une meta description uniques** [16], car cela aide à distinguer les pages entre elles et à améliorer le taux de clic en résultats de recherche. *« Toutes les pages indexables de votre site devraient avoir une meta title et description. Cela soutient le SEO organique… »* [16]. On respectera les longueurs recommandées (~60 caractères pour le title, ~155 pour la description) pour qu'ils ne soient pas tronqués [17].

- **Contenu indexable et optimisé :** Le site étant principalement applicatif, il faudra s'assurer que le contenu texte important est bien présent dans le HTML rendu côté serveur (Next.js SSR pour l'accueil, les pages info etc., afin que Googlebot puisse le lire). Les pages d'information /energie seront rédigées avec une optique SEO (mots-clés pertinents sur les aides, structure h1/h2/h3 logique, liens internes entre pages d'aide et pages de projet travaux). On pourrait envisager un blog ou des pages "actualités" pour attirer du trafic supplémentaire (non requis initialement, mais la structure pourrait le permettre).

- **URL propres et structurées** : Utilisation d'URL courtes et explicites, en minuscules, séparées par des tirets. Par exemple `/deposer-projet`, `/artisans/inscription`, `/energie/simulateur-led`, `/mentions-legales`. Pas de paramètres ou d'identifiants non nécessaires dans les URL publiques.

- **Performance et mobile friendly** : Les Core Web Vitals sont importants pour le référencement. Le fait d'être sur Vercel/Next.js aidera à avoir un site rapide (préchargement, CDN). On fera attention au poids des images, au chargement paresseux si besoin, et on testera les pages avec Google Lighthouse pour corriger d'éventuels points bloquants. Le design mobile-first garantit une bonne **compatibilité mobile**, critère indispensable (Google indexant en mobile-first).

- **Balisage sémantique** : Ajouter du **balisage schema.org** où pertinent, par exemple un *LocalBusiness* pour TravauxLinks, des *FAQ* si on intègre une section questions fréquentes, etc., afin d'obtenir d'éventuels rich snippets.

En mettant en œuvre ces optimisations dès la phase de développement, on s'assure que le site neuf démarre avec de bonnes bases SEO et que l'intégration du contenu d'EnvironnementCEE conserve toute sa visibilité dans les moteurs.

13. Fonctionnalités/Pages attendues

En synthèse, voici la liste des pages et fonctionnalités clés du MVP à développer :

- **Page d'accueil "/" :** présentation du service (comment ça marche en 3 étapes), argumentaires pour clients et pour artisans, CTA dépôt de projet. Témoignages ou logos éventuellement, et un encart sur les aides énergie pour rediriger vers /energie.

- **Page "/deposer-projet" :** formulaire dynamique en 3 étapes pour saisir un projet. Après validation, redirection vers…

- **Page "/merci" :** page de confirmation de dépôt, expliquant la suite (ex: "Merci, votre projet a été soumis. Nos artisans partenaires proches de chez vous vont vous contacter bientôt…"). Cette page peut aussi encourager à visiter la section énergie pour voir les aides disponibles.

- **Section artisans "/artisans/…" :** il y aura plusieurs pages dans l'espace artisans :

- **Inscription/Login** (`/artisans/inscription` et `/artisans/login` éventuellement) si on ne passe pas par des modales.
- **Dashboard** (`/artisans/dashboard`) avec la liste des leads, filtres par statut.
- **Page lead détail** (`/artisans/lead/{id}`) montrant le détail et les boutons Action (Intéressé, etc.). Celle-ci pourrait aussi être une modal sur le dashboard.
- **Chantiers/Projets réalisés** (`/artisans/chantiers`) listant les chantiers confirmés.
- **Profil** (`/artisans/profil`) pour voir/modifier ses infos, uploader documents, etc.

- **Section admin "/admin" :** accès restreint admin :

- **Dashboard CRM** listant les nouveaux leads et indicateurs (leads ce jour, leads en attente, taux de conversion…).
- **Vue détail lead** (`/admin/leads/{id}`) avec possibilité d'éditer assignations, voir timeline.
- **Liste artisans** (`/admin/artisans`) pour valider les inscriptions (ex: vérifier documents RGE envoyés) et éventuellement *activer/désactiver* un artisan.
- **Paramètres** (`/admin/settings`) pour configurer certains seuils (ex: délai LeadRisk).

- **Section énergie "/energie" :**

- **Page d'accueil énergie** (`/energie/`): introduction des aides, présentation de l'engagement environnemental, renvoi vers simulateur et formulaires.
- **Simulateur LED** (`/energie/simulateur-led`) : page intégrant l'outil de simulation d'économies (soit codé, soit en iframe si fourni).
- **Formulaire aides** (`/energie/demande-aide` ou `/energie/{type}-prime`) selon ce qui existait, pour collecter les demandes liées aux CEE.
- **Pages de contenu** : ex `/energie/CEE` (explication dispositif CEE), `/energie/isolation` (si existait), etc., en fonction de l'architecture du site EnvironnementCEE.
- **Mentions légales** (`/mentions-legales`) – accessible aussi via `/energie/mentions-legales` éventuellement.
- **Politique de confidentialité** (`/politique-confidentialite`) – à rédiger pour RGPD.

(Nota: La structure finale des URLs énergie sera à affiner une fois tout le contenu audité – l'essentiel est de tout conserver et de le rendre accessible sous /energie.)

- **Pages d'erreur** (`/404` , `/500`) customisées pour rester dans la charte, avec redirection vers l'accueil ou message clair.

Toutes ces pages seront reliées par une navigation cohérente : header avec menu (par ex "Trouver un artisan", "Aides énergie", "Espace Artisan"), footer avec les liens légaux, etc.

14. Modèle de données (Supabase)

Pour soutenir ces fonctionnalités, les principales tables Supabase (Postgres) envisagées sont :

- `users` : comptes utilisateurs de base. Ceux-ci représenteront les artisans et les admins (distinguer via un champ rôle, par ex `role = 'artisan'` ou `'admin'`). Les champs d'auth (email, password hash, etc.) sont gérés via Supabase Auth et stockés dans `auth.users`, mais on peut avoir une table publique `users` liée contenant profil de base (nom, rôle, date d'inscription). (Les clients n'ont pas de compte donc pas de user enregistré).

- `artisans` : profil détaillé des artisans. Clé primaire liée à `users.id`. Champs : nom entreprise, adresse, téléphone, secteurs d'intervention (peut-être stockés dans une table séparée `artisan_zones` pour les N:N département), métiers (pareil possiblement table `artisan_metiers` N:N ou un champ JSON), certification_RGE (bool ou champ texte avec n° de certif), URL logo ou photo, description libre, etc. Inclure un champ statut (actif, en attente validation docs, suspendu). Documents justificatifs liés via table `documents`.

- `leads` : demandes de travaux soumises. Champs : id, date, type_travaux (catégorie), description, code_postal/ville (lieu du chantier), éventuellement budget, prénom/nom client, téléphone, email, etc. **Attention** : les coordonnées clients *pourront être stockées chiffrées ou dans une table séparée sécurisée* si on veut être très RGPD ; mais plus simple est de les stocker ici et de s'assurer via RLS qu'elles ne sont pas lisibles par un artisan tant qu'il n'est pas assigné. Un champ status (nouveau, en_cours, conclu, etc.). Possibilité de stocker un score lead ou notes internes.

- `lead_artisan_matches` : table de liaison entre leads et artisans. C'est ici qu'on enregistre quels artisans ont reçu chaque lead. Champs : lead_id, artisan_id, date_notif env, status_cote_artisan (valeurs : "envoyé" par défaut, puis "interessé", "refusé", "assigné", "gagné/perdu" éventuellement). Cette table permet de lister pour un artisan ses leads, et de savoir pour un lead quels artisans ont répondu présent. L'administrateur peut y gérer les assignations (par ex passer un artisan en "assigné = true" pour lui signifier qu'il a accès au client). Elle sert aussi au RLS : un artisan authentifié pourra `SELECT` les leads en jointure seulement s'il a un enregistrement ici.

- `chantiers` : projets aboutis. Lorsqu'un lead se concrétise, on crée un chantier lié. Champs : id, lead_id (nullable si le chantier vient pas d'un lead ? mais ici oui issu lead), artisan_id, date_debut, date_fin (ou statut en cours/terminé), note_satisfaction (si client a évalué), commentaire éventuel. Cela permet de garder un historique pour l'artisan et potentiellement d'afficher des retours d'expérience.

- `documents` : stockage des documents uploadés sur la plateforme. Champs : id, type (ex "devis signé", "attestation RGE", "facture"), filename or URL (pointant vers le storage Supabase ou externe), artisan_id (propriétaire du doc si pertinent), lead_id ou chantier_id selon contexte. Par exemple, un devis PDF lié à un lead, une assurance décennale liée à un artisan. On pourra aussi stocker ici des éventuelles images (photos chantier).

- `events_timeline` : journal des événements et interactions. Champs : id, timestamp, type_evenement (ex "lead_created", "artisan_interested", "email_sent"), utilisateurs concernés (ex user_id de l'artisan ou client), entité concernée (lead_id, etc.), description/metadata JSON si besoin. Cette table permet à la fois d'afficher des timelines (vue admin notamment) et d'avoir du logging pour diagnostics.

- *(Tables annexes potentiellement :* `metiers` (liste des métiers pour normaliser), `zones` (liste départements), tables pivot `artisan_metier`, `artisan_zone` si on normalise ces relations N:N plutôt que champs text/array.*)*

Bien entendu, on activera RLS sur ces tables pour que :
- un artisan ne puisse voir/mettre à jour que son propre enregistrement dans `artisans` (et jamais ceux des autres),
- un artisan puisse voir les leads uniquement via `lead_artisan_matches` qui lui correspond, et seulement les champs autorisés (masquer contacts clients tant que condition non remplie, on peut réaliser cela via des policies SQL ou des views).
- l'admin (rôle spécifique) ait accès en lecture/écriture à tout, possiblement en contournant RLS grâce à un service key côté backend ou en ayant un rôle distinct dans les policies.

15. Assurance Qualité (QA) et Tests

Étant une application métier manipulant des leads commerciaux, une batterie de tests et de vérifications est prévue afin d'assurer la fiabilité du MVP avant sa livraison :

- **Tests E2E – Parcours client :** on simulera le parcours d'un particulier depuis l'arrivée sur la home jusqu'à la soumission du formulaire. Vérifier que chaque étape du formulaire fonctionne, que les validations (champs obligatoires, format email…) marchent, et qu'à la fin on a bien un lead créé en base, un email de confirmation envoyé et l'affichage de la page de remerciement.

- **Tests E2E – Parcours artisan :** créer un compte artisan de test, vérifier la réception du mail de vérification (si double opt-in), la complétion du profil. Ensuite, simuler l'envoi d'un lead qui correspond à ses critères et s'assurer qu'il le voit dans son dashboard. Tester les actions : cliquer "Intéressé" -> vérifier que le statut change, que les contacts client se dévoilent, qu'un email est envoyé au client (si prévu). Tester "Refuser" -> lead n'apparaît plus comme actif pour lui. Tester l'upload d'un document (devis) -> vérifier dans Supabase Storage ou table documents.

- **Matching et assignation :** créer plusieurs artisans avec zones/métiers variés, soumettre des leads et vérifier que seuls les artisans ad hoc reçoivent. Tester le cas où *aucun* artisan ne correspond (lead orphelin) : l'admin doit voir ce lead et pouvoir soit le réassigner manuellement soit marquer "Pas d'artisan disponible" dans le CRM.

- **LeadRisk Engine :** via des tests unitaires ou en manipulant l'horloge, vérifier que lorsqu'un lead reste sans intérêt 5 jours, il passe en flag inactif et qu'un email de relance est envoyé. Idéalement, simuler ce scénario en base (en réglant temporairement le délai à quelques minutes pour test). Vérifier la baisse de prix (si applicable) : par exemple un champ "prix" dans lead_artisan_matches mis à jour.

- **Scoring :** soumettre des suites d'actions et vérifier que le score évolue selon les règles. Ex: un artisan qui répond à 5 leads sur 5 a un score plus haut qu'un autre qui n'en répond qu'à 1 sur 5. S'assurer que le calcul est correct et qu'il n'y a pas de division par zéro ou autre.

- **Emails :** utiliser un environnement de test (sandbox de Resend ou intercept d'emails) pour valider que chaque type d'email part bien avec le bon contenu. Vérifier les liens dans les emails (ex: lien de dashboard artisan fonctionne, lien de désinscription newsletter s'il y a, etc.).

- **Sécurité RLS :** essayer en tant qu'artisan de requêter (via supabase JS ou l'API) des données qui ne devraient pas lui appartenir – s'assurer que l'accès est interdit. Par exemple, tenter de changer l'ID dans une URL (ex: `/artisans/lead/999` d'un lead qui n'est pas à lui) doit renvoyer une erreur ou une page 404. Idem, vérifier qu'un artisan ne peut pas via l'API REST lire la table leads complète : il ne doit obtenir que ses leads. Ces tests peuvent être automatisés avec des appels HTTP aux endpoints.

- **SEO :** auditer le site déployé avec des outils comme Screaming Frog ou Google Search Console : vérifier que le sitemap.xml est bien accessible et complet, que chaque page a sa balise title/description, qu'aucune page importante n'est en noindex, que les canonical sont en place et corrects. Tester le chargement d'une page /energie en mode "Fetch as Google" pour s'assurer que le contenu est bien rendu côté serveur (pas besoin de JS).

- **Performances et compatibilité :** tester le site sur mobile, tablette, desktop (plusieurs navigateurs) pour repérer tout bug responsive. Vérifier les temps de chargement (Lighthouse), corriger si des scripts bloquants ralentissent trop.

Chaque bug relevé sera corrigé avant mise en production. On pourra utiliser des outils de suivi (GitHub Issues ou Trello) pour lister les tâches QA. De plus, un **contrôle final** sera fait sur la procédure de déploiement (Vercel + variables d'environnement pour communiquer avec Supabase et Resend) afin de s'assurer que l'ensemble (front + backend + emails) fonctionne de façon intégrée sur l'URL finale.

16. Livrable et déploiement

Au terme du développement et de la phase de tests, le livrable attendu est : **la plateforme TravauxLinks.fr déployée en production sur Vercel**, pleinement connectée à Supabase (base de données, auth…) et à Resend (envois d'emails), avec l'ensemble des fonctionnalités du MVP opérationnelles.

Déploiement sur le domaine : Le domaine `travauxlinks.fr` (et `www.travauxlinks.fr`) pointera vers Vercel. Idem, on pourra faire pointer `environnementcee.fr` vers la même application (ou mettre des redirections vers travauxlinks.fr/energie). Les certificats SSL seront gérés via Vercel automatiquement.

Migration des données existantes : Si le site EnvironnementCEE disposait de contenus dynamiques (demandes d'aides déjà reçues, etc.), il faudra les migrer dans la nouvelle base Supabase. Cependant, s'il ne s'agissait que de pages statiques et d'emails reçus, la migration consistera surtout à rapatrier le texte et remettre en ligne les outils. On veillera à ce qu'**aucune information ou fonctionnalité ne soit perdue** lors de cette transition.

Qualité du code et documentation : Le code source, bien que partiellement généré par Lovable, sera **proprement organisé et documenté**. Un README technique sera fourni, incluant : - Instructions pour lancer le projet en local (utile pour la maintenance), - Schéma simplifié de la base de données, - Informations pour configurer les clés d'API (Supabase, Resend) et variables d'environnement, - Procédure pour ajouter un nouvel admin, etc.

Objectif MVP : La version livrée sera un MVP complet, c'est-à-dire **Minimal mais Viable et Produit** : toutes les grandes lignes fonctionnelles décrites ci-dessus seront implémentées de manière opérationnelle, même si des optimisations ou fonctionnalités avancées pourront être ajoutées après (par ex, on pourrait imaginer à l'avenir un chat instantané client-artisan, ou un système de paiement,

mais ce n’est pas dans le périmètre MVP). L’important est que chaque utilisateur-type (particulier, artisan, admin) puisse accomplir sa mission principale sur le site sans friction.

Une fois en production, un suivi serré sera de mise lors des premiers jours (monitoring des erreurs, feedback des premiers utilisateurs) afin de corriger rapidement tout problème imprévu. La base posée par ce projet devrait permettre à TravauxLinks de démarrer son activité de mise en relation avec un outil efficace, et d’évoluer facilement par la suite en ajoutant des itérations en fonction des retours du marché.

En conclusion, TravauxLinks.fr combinera la **simplicité** pour le client (déposer un projet en quelques clics), la **pertinence** pour l’artisan (recevoir des leads qualifiés dans sa zone) et la **confiance** via la section rénovation énergétique et les contrôles de qualité mis en place. Cette plateforme moderne, sécurisée et optimisée SEO sera prête à être lancée comme un service innovant dans le secteur des travaux et de la transition énergétique.

---

1 2 Artisan reconnu garant de l'environnement (RGE) | France Rénov'
https://france-renov.gouv.fr/recrutement/qualifications-rge

3 4 Qu'est-ce qu'une page canonique ?
https://www.imagescreations.fr/qu-est-ce-qu-une-page-canonique/

5 Respond to Leads Immediately, or Conversions Drop Sharply
https://imagebuildingmedia.com/marketing-edu/lead-generation/lead-response/respond-to-leads-immediately-or-conversions-drop-sharply

6 7 How Modern Marketplaces Like Uber and Airbnb Build Trust to Achieve Liquidity
https://review.firstround.com/how-modern-marketplaces-like-uber-airbnb-build-trust-to-hit-liquidity/

8 Managing Domains - Resend
https://resend.com/docs/dashboard/domains/introduction

9 What is Lovable AI? A Deep Dive into the Builder | UI Bakery Blog
https://uibakery.io/blog/what-is-lovable-ai

10 Introduction to Supabase - an Open Source Firebase Alternative | Microsoft Learn
https://learn.microsoft.com/en-us/shows/open-at-microsoft/introduction-to-supabase-an-open-source-firebase-alternative

11 Row Level Security | Supabase Docs
https://supabase.com/docs/guides/database/postgres/row-level-security

12 13 La signification des couleurs en communication et marketing - NOIISE
https://www.noiise.com/ressources/content-marketing/signification-des-couleurs-communication-marketing/

14 15 XML Sitemap: the ultimate reference guide
https://www.conductor.com/academy/xml-sitemap/

16 17 Meta Tags SEO | Meta Titles Descriptions | ZAG Interactive
https://www.zaginteractive.com/insights/articles/november-2023/what-are-meta-tags-why-are-they-important