Rapport d’audit

Application de prise de rendez-vous générée avec Lovable

Exemple fictif. Ce rapport illustre le livrable de l’audit Sparkforge (forfait « Reprise de POC »). L’entreprise, l’application, les extraits de code, les chiffres et les clés sont inventés. Toute ressemblance avec une application existante serait fortuite.

Client	Société Exemple (fictive), entreprise de dépannage et d’entretien de chaudières, 8 salariés, 3 techniciens
Application	Prise de rendez-vous en ligne, avec acompte et résumé automatique de la demande
Stack	Lovable (React, Vite, TypeScript), Supabase (base Postgres, authentification, fonctions Edge), Stripe Checkout, API d’un fournisseur d’IA
Version auditée	branche `main`, commit `4e1c9a7`
Auditeur	Aurélien Fontaine, Sparkforge
Remise du rapport	6 novembre 2026, 4 jours ouvrés après l’ouverture des accès
Prix de l’audit	1500 €, restitution d’une heure en visio comprise
Statut	Confidentiel, propriété du client

Un audit comme celui-ci : 1500 €, rapport sous 5 jours ouvrés.

Demander un appel gratuit de 30 minutes

EXEMPLE FICTIF

1. Synthèse

La question posée : l'application peut-elle accueillir de nouveaux clients, et à quelles conditions ?

La réponse courte : pas encore. Deux failles permettent aujourd'hui à n'importe qui sur Internet, sans compte, de lire la liste de vos clients et d'utiliser à vos frais votre compte chez le fournisseur d'IA. Elles se corrigent en un cycle de deux semaines, sans réécrire l'application. Je vous les ai signalées par écrit le jour où je les ai trouvées, sans attendre ce rapport.

Ce qui marche, et qu'on garde. Le parcours d'inscription et l'interface sont propres, agréables sur téléphone, et vos clients les utilisent déjà. Le résumé automatique des demandes fait gagner du temps à vos techniciens. Votre POC a prouvé que l'idée tient : il n'y a aucune raison de repartir de zéro.

Ce qui doit changer, par ordre d'urgence

Gravité	Constat	En clair
Bloquant	Clé d'API du fournisseur d'IA visible dans le navigateur	N'importe qui peut la copier et faire payer ses propres usages par votre compte, sans plafond.
Bloquant	Table `clients` lisible sans connexion	Noms, téléphones, adresses et codes d'accès de 1 284 clients sont téléchargeables en une commande.
Majeur	Clé d'administration de la base dans l'historique du code	Deux anciens prestataires ont encore un accès complet à vos données.
Majeur	Données personnelles envoyées au fournisseur d'IA	À chaque réservation, le nom, l'adresse et les codes d'accès du client partent chez le fournisseur d'IA, hors de l'Union européenne.
Majeur	Paiement mal contrôlé	Un acompte peut être perdu, ou un faux paiement accepté, et personne n'est prévenu.
Majeur	Ni sauvegarde testée, ni préproduction, ni tests	Une erreur de manipulation peut effacer vos données, et chaque modification se fait directement sur l'application en ligne.
Modéré	Calcul des créneaux disponibles	Deux clients peuvent réserver le même créneau ; depuis le passage à l'heure d'hiver, les rendez-vous pris avant cette date s'affichent avec une heure de décalage.

Ce que je recommande : une mise en sécurité, puis deux étapes, chacune avec son contenu et son prix fixe.

Étape	Contenu	Durée	Prix
Lot 1 · Mise en sécurité	Les deux bloquants, clés renouvelées, sauvegardes avec restauration testée, préproduction, tests des parcours critiques	1 cycle	5 000 €
Lot 2 · Paiement fiable et tests	Paiement qui se rattrape tout seul, tests étendus, alertes	1 cycle	5 000 €
Lot 3 · Créneaux et passation	Créneaux reconstruits, documentation, passation à votre équipe	1 cycle	5 000 €
Total		**3 cycles**	**15 000 €**

Soit 13 500 € avec la déduction de l'audit (1 500 €), si vous commandez le lot 1 dans les 30 jours suivant la remise du rapport. Vous pouvez vous arrêter après chaque lot.

En attendant, trois gestes à faire cette semaine, sans moi (détail en section 7) : fixer un plafond de dépense sur votre compte du fournisseur d'IA, faire une copie de votre base, retirer les accès au dépôt des personnes qui n'en ont plus besoin.

2. Périmètre et méthode

Ce que j'ai fait

- Lu le code de `main` et son historique Git (212 commits), fichiers supprimés compris ; inspecté le code envoyé au navigateur par l'application en ligne.
- Installé et exécuté l'application (inscription, réservation, paiement en mode test Stripe, back-office) ; mesuré les appels au fournisseur d'IA sur dix réservations de test.
- Interrogé l'API Supabase de production avec la seule clé publique, sans compte, comme le ferait n'importe quel visiteur, avec votre accord écrit. Je me suis limité à compter les lignes et à relever les noms des colonnes : aucune donnée réelle n'a été lue, copiée ni conservée.
- Lu la configuration Supabase (tables, règles d'accès, fonctions Edge, offre souscrite) et Stripe (webhooks, mode test et mode réel).

Hors périmètre : test d'intrusion et recherche exhaustive des failles ; avis juridique (les points liés au RGPD sont des points de vigilance techniques, sans valeur de conseil) ; design au-delà des écrans existants.

Comment lire les constats

Gravité	Signification
Bloquant	Exploitable aujourd'hui, depuis Internet, avec un effet direct sur vos clients ou votre trésorerie. À traiter avant toute ouverture.
Majeur	Risque réel, mais qui demande une condition particulière (un accès, une panne, une erreur). À traiter dans le premier ou le deuxième lot.
Modéré	Gêne, surcoût ou fragilité qui grandit avec le nombre de clients.
Mineur	Bonne pratique à adopter au passage.

Chaque constat indique sa **surface d'attaque** (qui peut s'en servir, et depuis où), sa **preuve**, son **impact** pour vous et pour vos clients, la **correction** proposée et le **lot** qui la traite.

3. Constats détaillés

C-01 · Clé d'API du fournisseur d'IA visible dans le navigateur

Gravité : Bloquant · **Surface** : n'importe qui sur Internet, sans compte · **Lot 1**

Signalé par écrit le jour de sa découverte (J+1 de l'audit), sans attendre ce rapport.

Preuve. Le résumé de la demande est calculé directement dans le navigateur. La clé est lue dans une variable d'environnement préfixée `VITE_`, que Vite recopie en clair dans le JavaScript publié.

```
// src/lib/summarize.ts
const res = await fetch("https://api.fournisseur-ia.example/v1/chat/completions", {
  method: "POST",
  headers: {
    Authorization: `Bearer ${import.meta.env.VITE_AI_API_KEY}`,
    "Content-Type": "application/json",
  },
  body: JSON.stringify({ model: MODEL, messages }),
});
```

```
$ curl -s https://rdv.example.com/assets/index-8f3a21c4.js | grep -o 'sk-[A-Za-z0-9_-]\{6\}'
sk-Xk4q9T
```

Le compte du fournisseur n'a aucun plafond de dépense mensuel.

Impact. Pour vous : n'importe qui peut copier la clé et l'utiliser pour ses propres usages, facturés sur votre carte. Des robots cherchent en permanence les clés d'IA exposées ; sans plafond, la facture n'a pas de limite. Pour vos clients : aucun effet direct, mais le service de résumé tombe dès que le fournisseur suspend le compte.

Correction. Déplacer l'appel dans une fonction Edge Supabase, qui seule connaît la clé ; n'accepter que les utilisateurs connectés ; limiter à un résumé par réservation et 30 par utilisateur et par jour ; fixer un plafond de dépense chez le fournisseur ; renouveler la clé, puisque l'ancienne doit être considérée comme compromise.

C-02 · Table `clients` lisible sans connexion

Gravité : Bloquant · **Surface** : n'importe qui sur Internet, sans compte · **Lot 1**

Signalé par écrit le jour de sa découverte (J+1 de l'audit), sans attendre ce rapport.

Preuve. Avec Supabase, la clé publique (`anon`) est faite pour être dans le navigateur : ce sont les règles d'accès de la base (*Row Level Security*, RLS) qui protègent les données. Ici, elles sont désactivées sur 4 tables sur 6.

```
select tablename, rowsecurity from pg_tables where schemaname = 'public';
--  tablename      | rowsecurity
--  clients        | false
--  bookings       | false
--  technicians    | false
--  payments       | false
--  profiles       | true
--  services       | true
```

```
$ curl -sI "https://abcd1234.supabase.co/rest/v1/clients?select=id" \
    -H "apikey: $ANON_KEY" -H "Prefer: count=exact" -H "Range: 0-0"
content-range: 0-0/1284
```

Les colonnes exposées : `full_name`, `phone`, `email`, `address` et `access_notes`, que le formulaire d'inscription présente comme le champ des codes d'immeuble et des consignes d'accès. Les règles de la base permettent aussi de **modifier** la table `bookings` sans connexion : un visiteur peut annuler ou déplacer un rendez-vous.

Le back-office est protégé seulement dans l'interface, par un test sur l'adresse e-mail (`if (user.email === ADMIN_EMAIL)`) ; la base, elle, ne fait aucune différence entre un administrateur et un visiteur.

Impact. Pour vos clients : leurs coordonnées, leur adresse et parfois la façon d'entrer chez eux sont lisibles par tous. Point de vigilance, sans valeur de conseil juridique : à faire examiner sans attendre par votre conseil ou votre DPO (obligations RGPD possibles). Pour vous : risque de réputation, perte de confiance, rendez-vous annulés par un tiers sans que vous le sachiez. C'est une faille connue des applications générées par IA qui s'appuient sur Supabase : une faille de cette famille a fait l'objet d'un avis public en 2025 (CVE-2025-48757, règles RLS insuffisantes dans des sites générés par Lovable).

Correction. Activer RLS sur toutes les tables et écrire une règle par usage : un client ne voit que ses propres données, un technicien ses rendez-vous, un administrateur tout, selon un rôle stocké en base (une table `staff`, à créer) et non plus une adresse e-mail dans le code. Exemple :

```
alter table public.clients enable row level security;

create policy "client lit sa fiche" on public.clients
  for select to authenticated
  using (user_id = (select auth.uid()));

create policy "admin lit tout" on public.clients
  for select to authenticated
  using (exists (select 1 from public.staff
                 where user_id = (select auth.uid()) and role = 'admin'));
```

Chaque règle est couverte par un test automatique (pgTAP, `supabase test db`) qui vérifie à la fois ce qui doit passer et ce qui doit être refusé. Activer RLS sans écrire les règles bloquerait toute l'application : ce n'est pas une correction de cinq minutes, c'est pourquoi elle se fait en préproduction d'abord.

C-03 · Webhook Stripe sans vérification de signature

Gravité : Majeur · **Surface** : n'importe qui connaissant l'adresse du webhook et le format d'un identifiant de réservation · **Lot 1**

Preuve. La fonction `stripe-webhook` lit le corps de la requête et confirme la réservation sans jamais vérifier l'en-tête `stripe-signature`.

Impact. Un message forgé suffit à confirmer un rendez-vous sans acompte. L'adresse de la fonction est visible dans le dépôt et prévisible. Le préjudice reste limité (un déplacement de technicien non payé), mais il est invisible tant que personne ne rapproche les paiements des rendez-vous.

Correction. Vérifier la signature avec `stripe.webhooks.constructEventAsync(body, signature, secret, undefined, Stripe.createSubtleCryptoProvider())` sur le corps brut de la requête, et refuser tout message non signé.

C-04 · Paiement sans reprise en cas d'erreur

Gravité : Majeur · **Surface** : aucun attaquant nécessaire, une panne suffit · **Lot 2**

Preuve. Deux mécanismes confirment la réservation, et aucun n'est fiable :

1. La page de retour de Stripe (`/merci`) passe la réservation à `confirmed` depuis le navigateur. Si le client ferme l'onglet avant, rien ne se passe.

2. Le webhook (C-03) avale ses erreurs et répond toujours `200`. Stripe considère donc le message comme reçu et ne le relance pas, même quand le traitement a échoué.

```
try { await confirmBooking(event); } catch (e) { console.log(e); }
return new Response("ok"); // Stripe ne réessaiera pas
```

Aucun identifiant d'événement n'est conservé. Or Stripe peut livrer un même événement plusieurs fois, et la page `/merci` confirme aussi : la confirmation et l'e-mail peuvent partir deux fois. Sur les 30 derniers jours, j'ai relevé dans le tableau de bord Stripe 3 acomptes payés pour des réservations restées « en attente ».

Impact. Pour vos clients : ils ont payé, ne reçoivent pas de confirmation, et le créneau peut être donné à quelqu'un d'autre. Pour vous : appels au standard, remboursements manuels, et aucune alerte quand cela arrive.

Correction. Faire du webhook la seule source de vérité ; enregistrer chaque `event.id` dans une table `stripe_events` avec une contrainte d'unicité (traitement idempotent) ; renvoyer une erreur `500` en cas d'échec pour que Stripe réessaie ; envoyer une alerte après trois échecs ; ajouter un rapprochement quotidien entre paiements Stripe et réservations.

C-05 · Clé `service_role` dans l'historique Git

Gravité : Majeur · **Surface** : toute personne ayant eu accès au dépôt (5 comptes, dont 2 anciens prestataires) · **Lot 1**

Preuve.

```
$ git log --all --oneline -- .env
b71d02e remove env file
3c09f5a add supabase config
$ git show 3c09f5a:.env | cut -c1-54
SUPABASE_SERVICE_ROLE_KEY=eyJhbGciOiJIUzI1NiIsInR5cCI6
```

Le fichier a été supprimé, mais reste dans l'historique. La clé `service_role` contourne toutes les règles d'accès de la base, et elle n'a jamais été renouvelée.

Impact. Quiconque a cloné le dépôt dispose d'un accès total à la base, en lecture et en écriture, même après la correction de C-02.

Correction. Renouveler les clés du projet Supabase ; retirer les accès au dépôt devenus inutiles ; ajouter une détection de secrets dans l'intégration continue. Réécrire l'historique est facultatif une fois la clé renouvelée.

C-06 · Aucune sauvegarde testée

Gravité : Majeur · **Surface** : une erreur de manipulation ou une modification ratée · **Lot 1**

Preuve. Le projet Supabase est sur l'offre gratuite, qui ne comprend pas de sauvegarde automatique. Aucune copie de la base n'existe ailleurs. Aucune restauration n'a jamais été tentée.

Impact. Une requête mal écrite, une migration ratée ou une suppression malveillante (voir C-02 : la table `bookings` est modifiable par tous) effacerait l'historique de vos clients et vos rendez-vous à venir, sans retour possible.

Correction. Passer à une offre avec sauvegardes quotidiennes ; ajouter une copie chiffrée hors de Supabase, chaque nuit ; **restaurer une sauvegarde en préproduction** et vérifier le résultat, puis répéter ce test chaque trimestre. Le résultat de chaque test de restauration est noté dans la documentation d'exploitation.

C-07 · Pas de préproduction, déploiement à la main

Gravité : Majeur · **Surface** : chaque modification · **Lot 1**

Preuve. Lovable et le développement local pointent sur le même projet Supabase que l'application en ligne. Les 14 migrations de la base ont été appliquées directement en production ; 3 d'entre elles ne sont pas dans le dépôt.

Impact. Chaque essai se fait sur les données de vos vrais clients. Corriger C-02 dans ces conditions risquerait de couper l'application pendant la journée.

Correction. Un second projet Supabase pour la préproduction, alimenté par des données fictives ; toutes les migrations versionnées dans le dépôt ; un déploiement reproductible, la même procédure versionnée pour la préproduction et la production ; une application de préproduction à son adresse propre, où vous testez chaque livraison pendant la **semaine de test** du cycle.

C-08 · Données personnelles envoyées au fournisseur d'IA

Gravité : Majeur · **Surface** : chaque réservation · **Lot 1**

Preuve. Le message envoyé pour le résumé contient la fiche complète du client, alors que seule la description de la panne est utile.

```
const messages = [
  { role: "system", content: SUMMARY_PROMPT },
  { role: "user", content: JSON.stringify({ client, booking }) },
  // client = { full_name, phone, email, address, access_notes }
];
```

Impact. Noms, adresses, téléphones et codes d'accès partent chez le fournisseur d'IA, dont les serveurs sont situés hors de l'Union européenne, sans que votre politique de confidentialité le mentionne. Point de vigilance, sans valeur de conseil juridique : à faire examiner par votre conseil ou votre DPO (obligations RGPD possibles).

Correction. N'envoyer que la description et le type d'appareil, après masquage automatique des numéros de téléphone et des adresses e-mail ; vérifier les options de non-conservation et la région de traitement proposées par le fournisseur ; mettre à jour la politique de confidentialité. Détail en section 5.

C-09 · Créneaux : double réservation et décalage horaire

Gravité : Modéré · **Surface** : usage normal · **Lot 3**

Preuve. La disponibilité est vérifiée dans le navigateur, puis la réservation insérée : entre les deux, un autre client peut prendre le même créneau.

```
const { data: taken } = await supabase.from("bookings")
  .select("id").eq("technician_id", t).eq("starts_at", slot);
if (!taken?.length) await supabase.from("bookings").insert({ ... });
```

Les dates sont construites sans fuseau horaire (`new Date("2026-10-27T09:00")`) et stockées dans une colonne `timestamp` sans fuseau : depuis le passage à l'heure d'hiver du 25 octobre, les créneaux s'affichent avec une heure de décalage pour les rendez-vous pris avant cette date. La même logique est recopiée, avec des variantes, dans 4 composants.

Impact. Deux techniciens prévus au même endroit, ou un client qui attend une heure. Le risque grandit avec le nombre de réservations.

Correction. Voir le verdict « Prise de rendez-vous » en section 4 : reconstruire plutôt que corriger.

C-10 · Résumé recalculé à chaque pause de frappe

Gravité : Modéré · **Surface** : usage normal · **Lot 1**

Preuve. Le résumé est relancé après chaque pause de 300 ms pendant la saisie de la description. Mesure sur dix réservations de test : **23 appels en moyenne** pour un seul résumé utile.

Impact. Un coût d'IA multiplié par 23 (chiffrage en section 5), et des résumés incohérents si les réponses arrivent dans le désordre.

Correction. Un seul appel, côté serveur, à la validation de la réservation.

C-11 · Aucun test automatisé, aucune intégration continue

Gravité : Modéré · **Lots 1 et 2**

Preuve. Aucun fichier de test dans le dépôt ; aucun workflow d'intégration continue ; `tsc --noEmit` signale 47 erreurs de type ignorées au build.

Impact. Chaque modification peut casser un parcours sans que personne ne le voie avant un client. C'est la cause probable des régressions que vous constatez « quand on touche à quelque chose ».

Correction. Lot 1, avant toute correction : tests de bout en bout (Playwright) des parcours critiques (inscription, réservation, paiement) et tests des règles d'accès (C-02). Lot 2 : tests étendus (annulation, back-office) et une intégration continue qui les lance à chaque modification et bloque la mise en ligne s'ils échouent.

C-12 · Réglages divers

Gravité : Mineur · **Lot 1** · Confirmation de l'e-mail désactivée, mots de passe de 6 caractères, session et fiche client affichées dans la console du navigateur. Correction : confirmation de l'e-mail et protection contre les mots de passe compromis activées, 10 caractères minimum, journaux retirés du code publié.

4. Verdict par partie

Je ne propose de reconstruire une partie que si cela coûte moins cher que de la corriger.

Partie	Verdict	Pourquoi	Constats
Inscription et connexion	À garder	Parcours clair, Supabase Auth bien intégré ; deux réglages à durcir.	C-12
Interface (écrans, composants)	À garder	Composants cohérents, bonne tenue sur téléphone. Aucune reprise nécessaire.	—
Données et accès	À corriger	Le schéma est sain ; ce sont les règles d'accès qui manquent.	C-02, C-05
Résumé par IA	À corriger	Le prompt et le rendu sont bons ; l'appel doit passer côté serveur, une seule fois, avec moins de données.	C-01, C-08, C-10
Paiement	À corriger	Stripe Checkout est le bon choix ; le traitement du retour doit devenir fiable.	C-03, C-04
Exploitation	À corriger	Sauvegardes, préproduction, déploiement, tests et alertes sont à créer.	C-06, C-07, C-11
Prise de rendez-vous (créneaux)	À reconstruire	Moins cher que de corriger, voir ci-dessous.	C-09

Prise de rendez-vous : les deux options côte à côte

	Corriger l'existant	Reconstruire le module
Ce qui est fait	Corriger les 4 copies de la logique une par une, convertir les dates existantes, ajouter un verrou applicatif	Une fonction Postgres unique qui calcule les créneaux et réserve ; une contrainte d'exclusion qui rend la double réservation impossible ; dates stockées avec fuseau horaire ; les écrans existants sont conservés
Risque restant	Une cinquième copie oubliée, ou ajoutée plus tard	Faible : la base refuse elle-même deux réservations qui se chevauchent
Durée et prix	2 cycles, 10 000 €	**Moins d'un cycle, compris dans le lot 3 (5 000 €)**

Recommandation : reconstruire. Le cœur de la reconstruction tient en une contrainte :

```
create extension if not exists btree_gist;
alter table public.bookings
  add constraint no_overlap exclude using gist
  (technician_id with =, tstzrange(starts_at, ends_at) with &&)
  where (status <> 'cancelled');
```

5. L'IA : coût par appel et données envoyées

Coût. Calcul sur la base d'un tarif indicatif retenu pour l'exemple (2,50 $ par million de jetons en entrée, 10 $ en sortie), converti au taux de 1 $ = 0,90 €, lui aussi retenu pour l'exemple. Le rapport réel utilise le tarif du fournisseur et le taux de change à la date de l'audit.

	Aujourd'hui	Après correction
Jetons par appel	environ 1 900 en entrée (fiche client complète) et 250 en sortie	environ 600 en entrée, 150 en sortie
Coût par appel	environ 0,0065 €	environ 0,0027 €
Appels par réservation	23 en moyenne	1
Coût par réservation	environ 0,15 €	environ 0,0027 €
Pour 400 réservations par mois	environ 60 €	environ 1,10 €
Plafond en cas d'abus	**aucun** (clé exposée, C-01)	plafond mensuel chez le fournisseur et limites par utilisateur

Le coût normal reste faible dans les deux cas. Le vrai risque financier est l'absence de plafond combinée à la clé exposée.

Données envoyées au fournisseur

Donnée	Envoyée aujourd’hui	Nécessaire au résumé	Après correction
Nom, e-mail, téléphone	oui	non	non
Adresse et codes d’accès	oui	non	non
Description de la panne	oui	oui	oui, numéros et e-mails masqués
Type et marque de l’appareil	oui	oui	oui

À vérifier avec le fournisseur choisi : durée de conservation des requêtes, utilisation pour l’entraînement, région de traitement. Héberger l’application en Europe ne garantit pas que les appels à l’IA y restent.

6. Plan de lots chiffré

Chaque lot dure un cycle de deux semaines : une semaine je développe, une semaine vous testez en préproduction. Avant chaque cycle, nous écrivons ensemble sa spécification ; des tests automatiques vérifient qu'elle est tenue. Si ces tests ne passent pas, le solde n'est pas dû, et l'acompte vous est remboursé si l'échec dépend de moi. Vous pouvez vous arrêter après n'importe quel lot.

Lot	Contenu	Constats	Durée	Prix ferme
1 · Mise en sécurité	Avant toute correction : sauvegardes, préproduction, déploiement reproductible, tests des parcours critiques. Puis les autres constats du lot, voir section 3	C-01 à C-03, C-05 à C-08, C-10 à C-12	1 cycle	**5 000 €**
2 · Paiement fiable et tests	Webhook qui ne traite jamais deux fois le même paiement, et que Stripe relance en cas d'erreur ; rapprochement quotidien Stripe / réservations ; tests étendus ; intégration continue ; alertes en cas d'erreur	C-04, C-11	1 cycle	**5 000 €**
3 · Créneaux et passation	Module de créneaux reconstruit (option recommandée en section 4) ; reprise des rendez-vous existants ; documentation d'exploitation ; fichier `AGENTS.md` pour continuer à développer avec des agents IA dans un cadre ; séance de passation	C-09	1 cycle	**5 000 €**
			3 cycles	**15 000 €**

Déduction de l'audit : 1 500 € sur le lot 1 s'il est commandé dans les 30 jours suivant la remise de ce rapport, soit 13 500 € pour les trois lots.

Paiement : pour chaque lot, un acompte de 30 % au démarrage du cycle et le solde de 70 % à la fin. Seul le cycle réalisé est dû.

Mise en production : rien ne part en production sans votre accord écrit. Par défaut, chaque lot est mis en ligne à la fin de son cycle, après votre semaine de test.

Après chaque livraison : les bugs du code livré sont corrigés gratuitement pendant 12 mois. Chaque lot est livré avec une note « ce qui a été corrigé, ce qui reste », et fonctionne sans moi.

Hors de ce plan : rappels de rendez-vous par SMS, tableau de bord d'activité, application pour les techniciens. Ils pourront se chiffrer après le lot 3, en cycles supplémentaires.

7. Ce que vous pouvez faire cette semaine, sans moi

1. **Fixer un plafond de dépense** mensuel bas sur votre compte du fournisseur d'IA, et activer l'alerte par e-mail. Cela borne le risque de C-01 en attendant le lot 1.
2. **Faire une copie de la base** (`supabase db dump`, ou demander l'export à la personne qui gère le projet) et la conserver hors de Supabase, chiffrée.
3. **Retirer l'accès au dépôt et au projet Supabase** des comptes qui n'en ont plus besoin, en particulier les anciens prestataires.

Ne désactivez pas l'accès public à la base sans avoir écrit les règles d'accès : l'application cesserait de fonctionner pour vos clients.

8. Restitution et suite

Ce rapport est présenté pendant une restitution d'une heure en visio : nous reprenons la synthèse, vous posez vos questions, et nous regardons ensemble l'ordre des lots. Le rapport vous appartient, que vous continuiez avec Sparkforge ou non ; il est écrit pour qu'un autre développeur puisse s'en servir.

Si vous souhaitez lancer le lot 1, nous écrivons sa spécification ensemble ; le cycle démarre à la date convenue.

Demander un appel gratuit

Exemple fictif. Ce document illustre le rapport remis à l'issue de l'audit du forfait « Reprise de POC » de Sparkforge. Entreprise, application, données, extraits de code et clés sont inventés. Les conditions de travail détaillées sont sur sparkforge.fr/conditions.