

Nouveau projet : JuraBreak Immobilier (site public + admin + annonces + estimation + analytics).

Objectif : livrer un repo propre, déployable, avec Supabase configuré correctement (SQL migrations + Storage + RLS + politiques + fonctions).

Règles strictes :

1. Tu fournis **des fichiers SQL de migration** (pas juste du texte dans le chat).
 2. SQL **sans erreur de syntaxe**, compatible Supabase/Postgres.
 3. RLS activé partout où nécessaire + politiques **cohérentes** (public read contrôlé, public insert limité, admin full).
 4. Tu ajoutes une **checklist de validation** et tu exécutes mentalement les cas “public”, “admin”, “non-auth”.
 5. Tu n’inventes pas d’éléments non demandés. Si incertitude, tu crées TODO.md.
-

A) Stack Projet (obligatoire)

- Next.js App Router en **JavaScript (.js)** uniquement.
- Supabase (Auth + DB + Storage).
- Stripe Checkout + webhook pour paiements estimations.
- Génération PDF côté serveur.
- Déploiement Vercel.

Créer un repo avec ces dossiers:

- /supabase/migrations/
 - /src/app/
 - /src/lib/
 - /src/components/
 - /src/server/ (helpers server actions / stripe / pdf)
-

B) Supabase: Schéma DB (SQL complet) + RLS + Politiques

B1) Exigences globales

- Toutes les tables listées ci-dessous doivent exister.
- **RLS activé** sur toutes les tables sensibles.

- Le site public peut lire uniquement:
 - annonces publiées (non supprimées)
 - événements
 - settings nécessaires (logo, contact)
- Le public peut seulement insérer:
 - leads (contact + demande visite)
 - analytics_events
 - estimations (DRAFT)
- Seul **admin** peut tout modifier.

B2) Définition du rôle admin

On utilise `profiles.role = 'admin'`.

Créer une fonction helper sécurisée:

Fonction `is_admin()`

- SECURITY DEFINER
- `search_path` fixé
- retourne boolean si `profiles.role='admin'` pour `auth.uid()`

B3) Tables requises (DDL)

Créer migrations SQL dans l'ordre:

0001_init.sql

- extensions si besoin
- tables:
 - profiles
 - agence_settings
 - annonces
 - annonce_photos
 - events
 - leads

- analytics_events
- estimations

Contraintes importantes:

- annonces.slug unique
- annonce_photos.position entre 0 et 7
- max 8 photos par annonce (trigger)
- annonces.statut CHECK IN ('EN_VENTE','SOUS_OFFRE','COMPROMIS','VENDU','RETIRE')
- soft delete annonces.is_deleted

0002_rls_policies.sql

- activer RLS
- policies détaillées (voir section C)

0003_triggers.sql

- trigger enforce_max_8_photos
- trigger set_updated_at sur tables principales

C) Politiques RLS (obligatoires, exactes)

C1) profiles

- SELECT: user peut lire sa ligne
- INSERT: via server/admin only (ou allow insert si auth.uid = id)
- UPDATE: user peut update sa ligne (optionnel), admin full

C2) agence_settings

- SELECT: public autorisé (read-only)
- INSERT/UPDATE/DELETE: admin only

C3) annonces

- SELECT: public peut lire seulement si is_deleted=false et published_at is not null
- SELECT: admin peut tout lire
- INSERT/UPDATE/DELETE: admin only

C4) annonce_photos

- SELECT public: seulement photos liées à annonces publiées/non deleted
- SELECT admin: tout
- INSERT/UPDATE/DELETE: admin only

C5) events

- SELECT public: ok
- INSERT/UPDATE/DELETE: admin only

C6) leads

- INSERT public: autorisé (sans auth)
- SELECT/UPDATE/DELETE: admin only

C7) analytics_events

- INSERT public: autorisé (sans auth)
- SELECT: admin only

C8) estimations

- INSERT public: autorisé (status DRAFT)
- UPDATE: uniquement backend/admin après paiement (admin only)
- SELECT: admin only
- IMPORTANT: empêcher public update d'un record.

D) Storage Buckets + règles

Créer buckets:

- annonces (images)
- public (logo/signature/cachet)
- estimations (pdf)

Règles:

- lecture public autorisée sur annonces et public (images affichées sur site)
- écriture uniquement admin sur annonces et public
- estimations lecture non publique (admin only), accès via lien signé si nécessaire

Fournir policies Storage (SQL) si nécessaire ou instructions Supabase.

E) Scripts & Vérifications automatiques (zéro surprise)

Créer un dossier /scripts avec:

- verify_rls.sql (requêtes qui listent tables sans RLS / policies manquantes)
- seed_admin.sql (créer un profil admin pour un email donné)

Ajouter un fichier CHECKLIST.md obligatoire avec:

- migration apply ok
 - RLS enabled ok
 - public can read published annonces only
 - public cannot read draft annonces
 - public can insert lead and analytics
 - public cannot read leads/analytics
 - admin CRUD annonces + photos ok
 - upload 9e photo refusé
 - stripe webhook met estimation PAID et génère PDF
 - vercel build ok
-

F) Déploiement (Vercel + Supabase) sans erreurs

Créer .env.example complet:

- NEXT_PUBLIC_SUPABASE_URL
- NEXT_PUBLIC_SUPABASE_ANON_KEY
- SUPABASE_SERVICE_ROLE_KEY (server only)
- STRIPE_SECRET_KEY
- STRIPE_WEBHOOK_SECRET
- STRIPE_PRICE_ID_FORMULE1
- STRIPE_PRICE_ID_FORMULE2
- EMAIL_PROVIDER_API_KEY (Resend/Sendgrid)

- `BASE_URL`

Créer README.md avec:

1. setup Supabase
 2. run migrations
 3. create admin user
 4. run local dev
 5. deploy vercel + env vars
-

G) Interdictions

- Ne pas changer le scope.
 - Ne pas ajouter de fonctionnalités “bonus”.
 - Ne pas supprimer des sections demandées.
 - Si doute, écrire TODO.md et s’arrêter là.
-

H) Livrables attendus

- Repo Next.js fonctionnel
 - Migrations Supabase dans `/supabase/migrations`
 - RLS + policies validées
 - Storage buckets + règles
 - Checklist + README + `.env.example`
 - Déploiement Vercel OK
-

Notes (à respecter)

- Tout doit être en `.js`
 - Pas de TypeScript
 - Pas de SQL approximatif: chaque migration doit passer sans erreur.
-

Fin

