0) Objectif V1 (déployable vite, extensible)

Livrer en V1

1. **Site public**
    - Page d'accueil **animée** (hero + slider photos + sections)
    - Liste des logements
    - Page détail logement (photos, équipements, prix indicatif, dispo)
    - Formulaire **demande de réservation** (request)
2. **Auth**
    - Inscription / login client (email + password)
    - Espace client (dashboard)
3. **Espace client**
    - Mes demandes / réservations (passées, en cours, à venir)
    - Statut paiement (au début: "en attente / payé / remboursé" géré admin)
    - Points fidélité (basique)
4. **Admin**
    - CRUD logements
    - Upload / ordre / suppression photos
    - Publier / dépublier logements (visible homepage)
    - Gestion demandes/réservations (accepter/refuser)
    - Marquer paiement (payé / non payé)
5. **Déploiement**
    - Supabase (DB + RLS + Storage)
    - Vercel (build OK, env OK)

Hors scope V1 (mais prévu)

- Paiement en ligne Stripe (V2)
- Channel manager iCal/Airbnb/Booking sync (V3)
- Tarifs saison + promos + coupons (V2)
- Contrat PDF / cautions / inventaire / ménage avancé (V2+)

- Avis publics et modération (V2)

---

1) UX / Pages (routes)

Public

- / : landing animée
- /logements : liste + filtres (ville, capacité, prix)
- /logements/[slug] : détail logement + galerie + demande réservation
- /login /signup : auth client
- /legal /privacy : textes (placeholder V1)

Client (protégé)

- /app : dashboard
- /app/reservations : liste
- /app/reservations/[id] : détail (statut, paiement)
- /app/profile : profil (téléphone, adresse, préférences)

Admin (protégé + caché)

- /admin/login : login admin (email/password)
- /admin : dashboard
- /admin/logements : liste
- /admin/logements/new : création
- /admin/logements/[id] : édition + photos + publish
- /admin/reservations : demandes/réservations
- /admin/reservations/[id] : détail + accepter/refuser + paiement

Caché par raccourci clavier sur /

- Sur la home: **Ctrl+C** (ou mieux Ctrl+Shift+A pour éviter le copier normal) → affiche un petit bouton discret “Admin”.
- Ce bouton redirige vers /admin/login.
- **Important**: Même si quelqu’un découvre l’URL, **RLS + rôle admin** bloque tout.

---

2) Style / Landing animée (spéc UI)

Home / sections

1. **Hero**
    - Background: slider 3–5 photos (fullscreen)
    - Titre + CTA “Voir les logements”
    - Petite animation (fade-in + parallax léger)
2. **Section “Nos logements”**
    - Cards (photo, titre, capacité, prix à partir de)
3. **Section “Pourquoi nous”**
    - 3–5 avantages (wifi, parking, vue, proche pistes, etc.)
4. **Section “Avis”** (mock V1)
5. **Footer**

Animations attendues

- Transition douce au scroll (IntersectionObserver)
- Cards hover (scale léger)
- Slider auto (interval)
- Skeleton loading

Interdiction Copilot: ne pas inventer 5000 logements, ne pas ajouter des stats bidons style “98% satisfaction”. Tu connais la musique.

---

3) Modèle data Supabase (V1)

On part sur un modèle **mono-client** mais déjà structuré **multi-tenant** (pour plus tard). Ça évite de tout casser à V2.

Tables (minimum viable)

orgs

- id uuid pk
- name text
- created_at

org_members

- org_id uuid fk
- user_id uuid fk auth.users
- role text check in ('admin','staff')
- PK (org_id, user_id)

profiles

- id uuid pk (= auth.uid)
- email text
- full_name text
- phone text
- created_at

properties (logements)

- id uuid pk
- org_id uuid fk
- title text
- slug text unique
- description text
- address_line1 text
- city text
- postal_code text
- country text
- lat numeric null
- lng numeric null
- capacity_adults int
- capacity_children int
- bedrooms int
- beds int
- bathrooms numeric
- surface_m2 int null

- amenities jsonb (ex: ["wifi","parking","sauna"])
- base_price_night numeric
- cleaning_fee numeric default 0
- deposit_amount numeric default 0
- is_published boolean default false
- created_at, updated_at

property_photos

- id uuid pk
- property_id uuid fk
- storage_path text (Supabase Storage)
- caption text null
- sort_order int default 0
- is_cover boolean default false
- created_at

availability_blocks (bloquages manuels)

- id uuid pk
- property_id uuid fk
- date_from date
- date_to date
- reason text null

booking_requests (demandes)

- id uuid pk
- property_id uuid fk
- user_id uuid fk
- date_from date
- date_to date
- adults int
- children int

- message text null
- status text check in ('pending','accepted','rejected','cancelled')
- created_at

bookings (réservation confirmée)

- id uuid pk
- property_id uuid fk
- user_id uuid fk
- request_id uuid null fk booking_requests
- date_from date
- date_to date
- nights int (computed côté app)
- price_nights numeric
- cleaning_fee numeric
- deposit_amount numeric
- total_amount numeric
- status text check in ('confirmed','cancelled','completed')
- created_at

payments (V1 manuel)

- id uuid pk
- booking_id uuid fk
- method text check in ('bank_transfer','cash','card_manual')
- status text check in ('pending','paid','refunded')
- amount numeric
- paid_at timestamptz null
- created_at

loyalty_accounts

- user_id uuid pk fk
- points int default 0

- updated_at

loyalty_ledger

- id uuid pk
- user_id uuid fk
- booking_id uuid null fk
- delta int (+/-)
- reason text
- created_at

---

4) RLS (sécurité) – règles simples et béton

Règle d'or

- **Public**: lecture uniquement sur properties publiés + photos associées.
- **Client**: ne voit que ses booking_requests, bookings, payments, loyalty_*.
- **Admin**: via org_members.role='admin' peut tout gérer pour son org_id.

Policies essentielles (résumé)

- properties:
    - SELECT: is_published = true (public)
    - INSERT/UPDATE/DELETE: admin org only
- property_photos:
    - SELECT: public si property publié
    - write: admin org only
- booking_requests:
    - INSERT: user connecté
    - SELECT/UPDATE: user propriétaire OU admin org
- bookings:
    - SELECT: user propriétaire OU admin org
    - INSERT/UPDATE: admin org only (V1)
- payments:

    - SELECT: user propriétaire OU admin org
    - INSERT/UPDATE: admin org only
- loyalty_*:
    - SELECT: user propriétaire
    - UPDATE: admin org only (ou trigger)

---

5) Storage Supabase (photos)

Bucket: property-photos (public read OK, mais write protégé)

- Path convention: org/{org_id}/property/{property_id}/{uuid}.jpg

Règles:

- Upload/delete: admin org only
- Public read: OK (sinon tu te bats avec signed URLs partout)

---

6) Logique métier V1 (workflow)

Demande → Acceptation → Réservation → Paiement

1. Client envoie booking_request (pending)
2. Admin accepte:
    - vérifie dispo (pas de chevauchement avec bookings confirmed + blocks)
    - crée booking + payment (pending)
    - passe request = accepted
3. Admin marque payment paid
4. Points fidélité:
    - à la confirmation “completed” (ou au paiement) créditer X points / nuit
    - V1: règle simple: points += nights * 10

Contrôle disponibilité (V1 simple)

- Un logement est “dispo” si:
    - aucune bookings.confirmed qui overlap
    - aucun availability_blocks qui overlap

---

7) App (structure projet) – JS only

Tu m'as déjà dit: **pas TypeScript**. Donc:

Stack recommandée V1

- Next.js (App Router) en **JavaScript**
- Supabase JS client
- UI: Tailwind + petits composants maison
- Animations: Framer Motion (facile, propre)
- Déploiement: Vercel

Arborescence

```
/app
  /(public)
    page.js              // home animée
    logements/page.js
    logements/[slug]/page.js
    login/page.js
    signup/page.js
  /(client)
    app/page.js
    app/reservations/page.js
    app/reservations/[id]/page.js
    app/profile/page.js
  /(admin)
    admin/login/page.js
    admin/page.js
    admin/logements/page.js
    admin/logements/new/page.js
    admin/logements/[id]/page.js
```

admin/reservations/page.js

admin/reservations/[id]/page.js

/lib

supabaseClient.js

auth.js

rlsHelpers.js

availability.js

/components

Navbar.js

Footer.js

PropertyCard.js

PhotoCarousel.js

BookingRequestForm.js

AdminGuard.js

ClientGuard.js

AdminShortcutListener.js

/styles (si besoin)

/public (images placeholders)

---

8) “Admin caché par Ctrl+C” (spéc technique)

Comportement

- Sur la home /, écouter un raccourci:
  - Option strict user: **Ctrl+C**
  - Option saine: **Ctrl+Shift+A** (recommandé)
- À la détection:
  - afficher un petit toast “Mode admin”
  - afficher bouton “Admin” pendant 10s

- clic → /admin/login

Contraintes

- Ne pas casser le copier normal dans les champs input/textarea.
- Ne pas déclencher si l'utilisateur est en train de sélectionner du texte dans un champ.

---

9) Seed / Données démo V1

- 3 logements
- 10 photos (placeholders)
- 1 admin (compte)
- 1 client (compte)
- 1 demande pending

---

10) Déploiement Vercel (checklist)

Supabase

1. Créer projet
2. Créer tables + RLS + bucket storage
3. Créer compte admin (Auth) puis l'insérer dans org_members role admin

Vercel env

- NEXT_PUBLIC_SUPABASE_URL
- NEXT_PUBLIC_SUPABASE_ANON_KEY

Optionnel:

- ADMIN_SHORTCUT=ctrl+c ou ctrl+shift+a

---

11) Tests d'acceptation (V1)

Public

- Home charge, animée, slider OK
- Logements listés uniquement si is_published=true

- Détail logement affiche galerie + infos

Client

- Signup/login OK
- Envoyer demande réservation OK
- Dashboard liste demandes/réservations
- Paiement visible (pending/paid)

Admin

- Raccourci affiche bouton admin
- Login admin OK
- CRUD logement OK
- Upload photo OK
- Publish logement → visible public
- Accept request → crée booking + payment
- Mark paid → client voit “paid”