# GLOUP — Plan de développement **piloté par LLM**

Stack cible : **React Native + Expo** · **TypeScript** · **Supabase** (Auth, DB, Realtime, Storage, Edge Functions) · EAS (Build/Submit/Updates)  
Design : **UI Kit “Bleu nuit”** figé + Présentation produit (4 onglets : Feed, Conseils, Messages & Groupes, Profil)

## 0) Objectifs & principes

* **Construire l’app 100% via LLM**, sous supervision humaine, avec **prompts standardisés**, garde‑fous, et **tests automatiques**.
* **Itérations rapides** (sprints 1–2 semaines), **A/B** sur UX critiques, **observabilité** (Sentry/analytics).
* **Contrats** (types/SQL/RPC) = source de vérité ; **RLS** et sécurité by default.
* **Points système** (réactions) : base = **10 pts**, **Couronne = 20 pts (×2)**, **Récompense Conseil = +200 pts**.

## 1) Organisation du travail (LLM‑Ops)

### 1.1 Rôles

* **Superviseur humain** : priorise, valide, merge.
* **Agents LLM** (prompts dédiés) :
  + **Architecte** (conception, schémas, RFC),
  + **Codegen** (écriture de code + tests),
  + **Reviewer** (revue PR + menaces sécu),
  + **Tester** (cas de tests, e2e Maestro/Detox),
  + **Doc** (README, ADR, guides).

### 1.2 Méthode

* **Branches** par feature (feat/…) + **PRs** obligatoires + **Revues LLM** + check humain.
* **Conventions** : Conventional Commits, Keep a Changelog, SemVer.
* **ADR** (Architecture Decision Records) pour choix majeurs.

### 1.3 Dossiers (monorepo simple)

/apps/mobile # Expo RN app  
/packages/ui # DS & composants  
/packages/theme # tokens & thèmes  
/packages/api # SDK client (Supabase + RPC typed)  
/supabase # migrations SQL, policies, seed, edge functions  
/e2e # Maestro/Detox scripts  
/.github/workflows # CI/CD  
/docs # ADR, RFC, UX specs

## 2) Pré‑projet & environnement

1. **Initialisation** : Expo (TS), EAS, NativeWind/Tamagui, React Query, Expo Router, FlashList, Reanimated, Gesture Handler.
2. **Qualité** : ESLint, Prettier, TypeCheck strict, Husky + lint‑staged.
3. **Secrets** : app.config.ts + dotenv EAS (profiles dev/preview/prod).
4. **CI/CD** : GH Actions — Lint/Test, Build Preview (EAS) → lien QR sur PR.
5. **Sentry/Analytics** : intégration dès le jour 1.

## 3) Modèle de données (Supabase)

### 3.1 Tables clés (simplifié)

* profiles(id uuid pk, username text unique, display\_name text, bio text, avatar\_url text, banner\_url text, created\_at)
* follows(follower uuid fk, followee uuid fk, created\_at, pk(follower,followee))
* posts(id uuid pk, author uuid fk, text text, media jsonb, created\_at, visibility)
* reaction\_types(key text pk, label text, emoji text, base\_points int)
* post\_reactions(id uuid pk, post uuid fk, reactor uuid fk, type text fk → reaction\_types, created\_at)
* advice(id uuid pk, post uuid fk, advisor uuid fk, body text, created\_at)
* advice\_rewards(id uuid pk, advice uuid fk, by\_author uuid fk, points int default 200, created\_at)
* messages(id uuid pk, conv\_id uuid, sender uuid, body text, media jsonb, created\_at)
* conversations(id uuid pk, type text enum('dm','group'), created\_at)
* conversation\_members(conv\_id uuid, user\_id uuid, role enum('admin','member'), joined\_at, pk(conv\_id,user\_id))
* notifications(id uuid pk, user\_id uuid, type text, payload jsonb, read\_at, created\_at)
* reports(id uuid pk, reporter uuid, entity\_type text, entity\_id uuid, reason text, created\_at, status)
* user\_points(user\_id uuid pk, total int, updated\_at)

### 3.2 Politiques (RLS) & index

* RLS **ON** pour toutes les tables user‑data.
* Policies : lecture publique contrôlée des posts/profils, écriture owner‑only, follows, etc.
* Index : posts(author, created\_at), post\_reactions(post, type), follows(follower), follows(followee).

### 3.3 Triggers & RPC

* Trigger after insert on post\_reactions → incrémente user\_points.total selon reaction\_types.base\_points.
* RPC reward\_advice(advice\_id) → +200 pts au conseiller (auth = auteur du post).
* RPC feed\_for\_user(user\_id, mode) → timeline (PourToi/Suivis) avec scoring simple (récence + points).

### 3.4 Seed

* **reaction\_types** : (crown, “Couronne”, “👑”, **20**), (style, “Style”, “👗”, **10**), (fitness, “Fitness”, “💪”, **10**), (confidence, “Confiance”, “😎”, **10**), (care, “Soins”, “🧼”, **10**), (wellbeing, “Bien‑être”, “😊”, **10**).

## 4) Backlog produit → Epics & DoD

### E1 — Auth & Onboarding

* **Livrables** : Sign in/up magic link, profil minimal, choix sujets, consentements.
* **DoD** : tests e2e (flow complet), RLS vérifiées, analytics événements auth\_\*.

### E2 — Composer & Médias

* **Livrables** : texte riche, upload images/vidéos (Storage), alt‑text obligatoire.
* **DoD** : contraintes taille/ratio, retry upload, tests unit + e2e, modération pré‑envoi.

### E3 — Feed (Pour toi / Suivis)

* **Livrables** : FlashList, pagination, pull‑to‑refresh, skeletons, barres d’actions.
* **DoD** : 60fps liste 1k+ items simulés, scroll perf profillée, offline state.

### E4 — Réactions & Points

* **Livrables** : ReactionPicker (6), compteurs, +points auteur, anti‑spam (cooldown).
* **DoD** : tests RPC/trigger, idempotence, UI haptique, analytics reaction\_\*.

### E5 — Conseils & Récompenses

* **Livrables** : page Conseils (Top/Pour vous/En hausse), récompense +200 pts, historique.
* **DoD** : tri par score, filtres, abuse checks, e2e (récompense autorisée ↔ auteur du post).

### E6 — Messages & Groupes

* **Livrables** : DMs, groupes, fichiers, réactions inline, mentions, liens d’invit.
* **DoD** : Realtime fiable, pagination, notifications locales, demande‑box.

### E7 — Profil & Personnalisation

* **Livrables** : header, onglets, stats, badges Mentor, thèmes via points.
* **DoD** : éditions sécurisées, cache images, tests screenshot.

### E8 — Modération & Sécurité

* **Livrables** : filtres toxicité (pré/post), signalements, blocage/silence.
* **DoD** : faux‑positifs < seuil, SLA traitement, journal d’audit.

### E9 — Recherche & Hashtags

* **Livrables** : recherche unifiée, explore, filtres.
* **DoD** : temps de réponse p95 < 300ms (cache/edge function), pertinence validée.

### E10 — Notifications

* **Livrables** : push (Expo), inbox notifs, réglages.
* **DoD** : opt‑in/opt‑out, deep links, cold start correct.

## 5) Flux LLM par étape (prompts prêts à l’emploi)

### 5.1 Gabarits universels

* **Codegen**

RÔLE: Ingénieur RN/Expo + Supabase.  
BUT: Implémente la feature {feature}.  
CONTRAINTES: TS strict, pas de TODO, fichiers complets avec chemins, hooks/test inclus.  
ENTRÉES: UI Kit (tokens), schémas DB, contrats types.  
SORTIES: PR squelette (patch), explications brèves, cas de test.

* **Reviewer**

RÔLE: Senior reviewer (perf/sécu/UX).  
VÉRIFIE: accessibilité, RLS, erreurs race conditions, perfs FlashList, mémoire.  
RÉSULTAT: liste d’issues concrètes + patchs.

* **Tester**

RÔLE: QA.  
PRODUIS: tests Jest + @testing-library/react-native, scripts Maestro/Detox, datasets mock.  
COUVERTURE: happy path + edge + échec réseau.

### 5.2 Prompts spécialisés

* **Schema/SQL (Supabase)**

RÔLE: DBA Supabase.  
OBJECTIF: Créer/mettre à jour tables, index, RLS, triggers et RPC pour {feature}.  
PRODUIS: fichier migration SQL + commentaires + tests SQL (pgTAP si dispo).

* **Edge Function**

RÔLE: Dev Edge Function.  
OBJECTIF: Implémenter RPC {nom} avec validation d’input, auth, et retours typés.

* **Composants UI**

RÔLE: UI engineer.  
OBJECTIF: Créer {Component} conforme tokens.  
INCLUT: story (Storybook/Expo Storybook), tests de rendu, variantes.

## 6) Qualité, tests & sécurité

* **Unit** : Jest + RTL (components, hooks).
* **Intégration** : mocks Supabase (msw), tests RPC.
* **E2E** : Maestro (scénarios auth → post → réaction → conseil → récompense).
* **Perf** : profiling Flipper/Perf Monitor, mémoire images, jank < 1%.
* **Sécu** : RLS exhaustive, rate‑limit (Edge), validation Zod côté client & server.
* **CI** :
  + Job 1: Lint/Typecheck/Unit
  + Job 2: Build Preview EAS + publie QR en commentaire PR
  + Job 3: Migrations supabase db push + tests sqitch/pgTAP

## 7) Observabilité & analytics

* **Sentry** : erreurs JS & performances (traces).
* **PostHog/Amplitude** : funnels (onboarding, réaction, récompense).
* **Dash** : métriques points/jour, rétention, abus détectés.

## 8) Performance & UX

* **Listes** : FlashList (estimatedItemSize), windowing agressif, keyExtractor stable.
* **Images/Vidéos** : thumbnails, lazy, cache, pause off‑screen.
* **Network** : React Query (staleTime, cacheTime, backgroundRefetch).
* **Motion** : Reanimated pour interactions, disable si « réduire animations ».

## 9) Sécurité & conformité

* **RLS** partout, audit logs, prévention enumeration IDs.
* **Politique contenu** + flows report/ban.
* **Privacy** : minimisation, consentements, export/suppression compte.

## 10) Livraisons

* **Alpha (Semaine 6–8)** : Auth, Feed basique, Composer, Réactions (pts), Profil.
* **Beta (Semaine 10–12)** : Conseils (+200), Messages/Groupes v1, Notifs, Modération v1.
* **1.0 (Semaine 14+)** : Personnalisation profil, Recherche, stabilité/perf, durcissement sécu.

## 11) Definition of Done (par PR)

* Code + tests (unit/inté/e2e) + docs (README section).
* A11y OK, perfs profilées si liste/animation.
* Migrations & RLS vérifiées.
* Build Preview réussi + QA passé.

## 12) Annexes — Checklists prêtes à copier

* **PR Template** : description, captures, tradeoffs, risques, tests, checklist a11y.
* **Issue Template** : user story, critères d’acceptation, métriques.
* **Release Checklist** : migrations, env, notes, rétrocompat.

## 13) Prochaines actions (jour 1–3)

1. Bootstrap repo + CI + Expo preview.
2. Générer migrations SQL (schema + seed reaction\_types).
3. Intégrer UI Kit dans /packages/theme & /packages/ui.
4. Implémenter Auth + Onboarding (E1) via LLM prompts ci‑dessus.
5. Écrire scénarios e2e Maestro init (auth → feed vide).