

Dossier d'analyse technique — Tontine Éthique (Tontetic)

Analyse de 12 fichiers (Flutter + Firebase + Stripe) — 2026-01-30

Objectif : fournir un diagnostic exploitable pour un déploiement production (Android + Web) en identifiant les causes des bugs actuels, les incohérences d'architecture, les risques sécurité et les actions correctives prioritaires.

1. Résumé exécutif

État global : le socle Firebase/Firestore est présent, mais la couche paiement et certains services restent incompatibles Web ou incohérents (résidus Supabase / démos). Le blocage actuel le plus critique est la crash Web lors du changement de plan (Stripe).

1.1 Priorités (ordre conseillé)

1. Stabiliser Stripe sur Web : interdire flutter_stripe côté Web et basculer sur Stripe Checkout via Cloud Functions.
2. Verrouiller la cohérence App ↔ Back Office : mêmes collections Firestore, mêmes règles/claims admin, même projet Firebase.
3. Éliminer les derniers artefacts “demo/mock” et les dépendances incohérentes (ex: Supabase).
4. Mettre en place un protocole de tests end-to-end (checkout → webhook → Firestore → back office).

1.2 Causes racines probables du bug abonnement Web

- Initialisation StripeService sur Web utilise des appels non supportés (Platform._operatingSystem / dart:io) → crash.
- Absence de séparation “Stripe mobile” (PaymentSheet) vs “Stripe web” (Checkout redirect).
- Deep links gérés trop tôt (context null) : navigation perdue possible après checkout.

2. Périmètre et méthode

Périmètre analysé : 12 fichiers fournis (Flutter/Dart + Cloud Functions JS). Analyse statique (lecture du code), orientée production : compatibilité Web, architecture Firebase, paiements Stripe, qualité (mocks), et sécurité de base.

2.1 Fichiers analysés

Fichier	Nb. lignes	Signaux détectés
stripe_service.dart	427	Clés Stripe live détectées (risque), Présence de secrets / clés (à vérifier), Logs via print() (à désactiver en prod)
index.js	456	Présence de secrets / clés (à vérifier), Usage functions.config() (ok pour secrets CF)
user_provider.dart	662	Logs via print() (à désactiver en prod), Indices de mock/démo
auth_service.dart	443	Présence de secrets / clés (à vérifier), Logs via print() (à désactiver en prod)
auth_provider.dart	18	Logs via print() (à désactiver en prod)
user_subscription_service.dart	563	Logs via print() (à désactiver en prod), Indices de mock/démo
circle_service.dart	222	Logs via print() (à désactiver en prod), TODO/FIXME restant
create_tontine_screen.dart	1206	Logs via print() (à désactiver en prod), TODO/FIXME restant, Indices de mock/démo
mobile_money_service.dart	283	Présence de secrets / clés

		(à vérifier), Logs via print() (à désactiver en prod), Indices de mock/démo
biometric_auth_service.dart	284	Présence de secrets / clés (à vérifier), Utilisation crypto (hash/signature)
log_sanitizer.dart	146	Logs via print() (à désactiver en prod)
tontine_rule_history_service.dart	245	Logs via print() (à désactiver en prod), Indices de mock/démo, Dépendance Supabase (incohérent si Firebase)

Note : les “signaux” sont des heuristiques (mots-clés). Une validation finale nécessite exécution, logs et tests.

3. Architecture recommandée (production)

3.1 Abonnement : séparation Web vs Android

- Android : App → Cloud Function → flutter_stripe (PaymentSheet) → webhook → Firestore → back office.
- Web : App → Cloud Function (createCheckoutSession) → redirection Stripe Checkout → retour app → webhook → Firestore.

3.2 Paiements de cotisations (principe)

- L'app n'orchestre pas “l'argent” : elle orchestre des intentions (documents/requests).
- Cloud Functions valide et crée les objets Stripe.
- Le webhook Stripe est la source de vérité et met à jour Firestore.

4. Constats détaillés

4.1. Incompatibilité Stripe sur Web (crash lors du changement de plan)

Sévérité : Critique

Impact : Le parcours d'abonnement payant casse sur Web, empêchant monétisation et déploiement Web stable.

Éléments observés :

- Logs Web : “Unsupported operation: Platform._operatingSystem” + “Fatal: Failed to initialize StripeService”.
- Le code Stripe est très probablement partagé entre Web et Android (même service).

Recommandations (actionnables) :

5. Créer StripeServiceWeb (Checkout URL + redirection) et StripeServiceMobile (PaymentSheet).
6. Bloquer toute initialisation flutter_stripe si kIsWeb.
7. Assurer le flux webhook : success_url/cancel_url → Firestore (status) → UI refresh.

Fichiers concernés : stripe_service.dart, user_subscription_service.dart, index.js

4.2. Cohérence App ↔ Back Office : projet Firebase, collections, règles et rôles admin

Sévérité : Élevée

Impact : Données visibles côté app mais pas côté admin si divergence de projet/collections ou règles Firestore.

Éléments observés :

- Erreur observée : “Missing or insufficient permissions”.
- Risque fréquent : Admin UI pointe vers un autre projectId ou une autre collection.

Recommandations (actionnables) :

8. Vérifier que l'admin et l'app utilisent le même projectId et les mêmes firebase_options.
9. Définir un “contrat de données” : collections/champs requis.
10. Mettre un rôle admin (custom claims) OU utiliser Admin SDK côté serveur pour le back office.

Fichiers concernés : auth_service.dart, user_provider.dart, circle_service.dart

4.3. Supabase présent dans le projet (incohérent si backend Firebase)

Sévérité : Élevée

Impact : Risque de feature cassée, divergence de données, et complexité inutile.

Éléments observés :

- Référence à Supabase dans un service d'historique de règles.
- Commentaire indiquant un hash “demo” non destiné à la prod.

Recommandations (actionnables) :

11. Migrer cette feature sur Firestore OU la désactiver proprement si hors scope.
12. Remplacer le hash par SHA-256 si l'intégrité est requise.
13. Supprimer toute dépendance Supabase si l'app est 100% Firebase.

Fichiers concernés : tontine_rule_history_service.dart

4.4. Artefacts de démo et features futures (risque de parcours incomplets)

Sévérité : Moyenne

Impact : Augmente la dette technique et crée des comportements surprise en prod.

Éléments observés :

- Présence possible de liens /demo (fallback) lors de création de tontine.
- Service mobile money présent alors que reporté.

Recommandations (actionnables) :

14. Mettre mobile money derrière un feature-flag OFF en prod.
15. Remplacer tout fallback /demo par un deep link réel basé sur circleId.
16. Ajouter une règle CI : refuser build release si “demo/mock/simulate” détecté.

Fichiers concernés : create_tontine_screen.dart, mobile_money_service.dart

4.5. Biométrie : garde-fous plateforme (Web)

Sévérité : Faible à Moyenne

Impact : Peut générer erreurs ou UX dégradée sur Web si exposé.

Éléments observés :

- Biométrie utile surtout sur mobile.

Recommandations (actionnables) :

17. Désactiver/masquer l'option biométrie sur Web.
18. Ajouter garde-fou kIsWeb et gérer erreurs proprement.

Fichiers concernés : biometric_auth_service.dart

4.6. Hygiène logs & données sensibles

Sévérité : Moyenne

Impact : Logs trop verbeux peuvent exposer des identifiants ; bon point : sanitizer existe.

Éléments observés :

- Présence de log_sanitizer (bon signal).
- Dans le projet global, plusieurs print() existent (vu dans tes sorties flutter analyze).

Recommandations (actionnables) :

19. Centraliser un logger avec niveaux et sanitation obligatoire.
20. Éliminer print() en prod (ou neutraliser en release).

Fichiers concernés : log_sanitizer.dart

5. Analyse par fichier (synthèse actionnable)

5.1 stripe_service.dart

Points clés :

- Service Stripe côté app ; utilisé pour abonnements/paiements.
- Suspect principal du crash Web.

Risques / limites :

- Incompatibilité Web si Platform/dart:io/flutter_stripe est touché.
- Initialisation globale non conditionnée.

Actions recommandées :

21. Séparer web/mobile (imports conditionnels).
22. Web → Checkout URL, Mobile → PaymentSheet.
23. Ajouter tests E2E checkout/webhook.

5.2 index.js

Points clés :

- Cloud Functions : meilleur endroit pour créer checkout sessions et webhooks.
- Peut devenir source de vérité paiement.

Risques / limites :

- Idempotence webhook à confirmer.
- Schéma Firestore paiements/subscriptions à standardiser.

Actions recommandées :

24. Ajouter idempotence (eventId).
25. Vérifier signature webhook Stripe.
26. Normaliser l'écriture Firestore (status, ids).

5.3 user_provider.dart

Points clés :

- Synchronisation user via Firestore.
- Centralise zone, statut, ids Stripe.

Risques / limites :

- Si imports Riverpod/Firebase/Auth manquent → analyse casse.
- copyWith incomplet peut casser UI.

Actions recommandées :

27. Finaliser refactor StateNotifier propre.
28. Gérer absence doc user sans crash.
29. Tests login/signup + refresh.

5.4 auth_service.dart

Points clés :

- Signup/login + écriture user dans Firestore.
- Champs sensibles chiffrés (approche).

Risques / limites :

- Admin doit lire même collection.
- Chiffrement E2E ≠ chiffrement au repos : attentes à cadrer.

Actions recommandées :

30. Documenter schéma user.
31. Aligner admin UI.
32. Éviter champs “demo”.

5.5 auth_provider.dart

Points clés :

- Expose état auth à l'app.

Risques / limites :

- Risque faible si aligné avec user_provider.

Actions recommandées :

33. Valider séparation des flux auth admin vs app.

5.6 user_subscription_service.dart

Points clés :

- Orchestration plans/upgrade/downgrade.

Risques / limites :

- Couplage StripeService → crash Web si non séparé.

Actions recommandées :

34. Dépendre d'une abstraction Stripe (web/mobile).

35. Gérer loading/error sans crash.

5.7 circle_service.dart

Points clés :

- CRUD cercles via Firestore.

Risques / limites :

- Index Firestore et permissions à valider.

- Contrat de données nécessaire pour admin.

Actions recommandées :

36. Définir collections/champs officiels.

37. Créer index requis.

38. Tester règles (user vs admin).

5.8 create_tontine_screen.dart

Points clés :

- UI création cercle, invitations.

Risques / limites :

- Fallback /demo possible.

- Logique métier potentiellement côté UI.

Actions recommandées :

39. Invitations réelles (deep link circleId).
40. Validations sensibles côté CF.
41. Nettoyage artefacts demo.

5.9 mobile_money_service.dart

Points clés :

- Feature future.

Risques / limites :

- Appel accidentel en prod.
- Confusion produit.

Actions recommandées :

42. Feature-flag OFF.
43. Retirer parcours UI tant que non implémenté.

5.10 biometric_auth_service.dart

Points clés :

- Biométrie mobile.

Risques / limites :

- Web : support limité.

Actions recommandées :

44. Masquer sur Web.
45. Garde-fous kIsWeb.

5.11 log_sanitizer.dart

Points clés :

- Sanitization logs (bon).

Risques / limites :

- À adopter partout sinon inutile.

Actions recommandées :

- 46. Brancher au logger global.
- 47. Remplacer print() partout.

5.12 tontine_rule_history_service.dart

Points clés :

- Historique de règles.

Risques / limites :

- Supabase incohérent.
- Hash demo non prod.

Actions recommandées :

- 48. Migrer Firestore ou désactiver.
- 49. SHA-256 si nécessaire.

6. Plan d'action (roadmap)

6.1 Sprint 1 — Débloquer Web (Stripe)

50. Refactor StripeService : séparation web/mobile.
51. Créer Cloud Function createCheckoutSession + routes success/cancel.
52. Mettre à jour UI abonnement : redirection Web + gestion retour + erreurs.
53. Valider webhook → Firestore → UI.

6.2 Sprint 2 — Back Office relié et fiable

54. Vérifier firebase_options : admin et app = même projectId.
55. Définir contrat de données (collections/champs) + index Firestore.
56. Rôle admin : custom claims ou Admin SDK.
57. Health checks admin (compteurs, dernières transactions).

6.3 Sprint 3 — Nettoyage final

58. Supprimer/migrer Supabase.
59. Feature-flag mobile money OFF.
60. CI anti-mock/demo/TODO en release.
61. Logger global + sanitizer + suppression print().

7. Check-list de validation finale

7.1 Paiements

- Web : clic plan → Stripe Checkout → retour OK → Firestore abonnement à jour (webhook).
- Android : PaymentSheet OK → Firestore à jour (webhook).
- Webhooks idempotents (même event 2x = 1 effet métier).

7.2 App ↔ Admin

- Signup user dans app → visible dans admin.
- Création cercle → visible dans admin.
- Admin peut lire/filtrer sans erreurs permissions.

7.3 Qualité prod

- flutter analyze : 0 erreurs.
- flutter build web --release : OK.
- Aucun secret hardcodé dans repo.
- Aucune mention mock/demo/simulate sur parcours prod.