Intégration Fidélité dans Shop

Contexte

Décision actée : — un tenant fidelity-api = une Société core-api(isolation par la base de données elle-même, pas par un filtre applicatif).

Étape 1 — core-api : mapping Société → tenant fidelity

1- Nouvelle migration tenants :Cree une table (fidelity_domain, fidelity_activated_at,) pour stocker les infos de la societe recu cote fidelity

Étape 2 — Provisioning : activer la fidélité pour une société

shop-core-api

1. api pour activate fidelité pour une societe — , avec fallback fidelity-api /api/v1/register/company (endpoint public existant).
2. La création du tenant étant asynchrone côté fidelity-api (CreateCompanyTenantJob en queue), ne pas bloquer la réponse : marquer la société "provisioning en cours", et soit (a) poller GET company/{id} (déjà existant) via un job différé jusqu'à obtenir le domaine, soit (b) exposer un webhook si fidelity-api en a un pour la création de tenant (à vérifier à ce moment-là — sinon partir sur le polling, plus simple).
3. Une fois le domaine confirmé, PATCH la société core-api (étape 1) avec fidelity_domain + fidelity_activated_at.

Étape 3 — FidelityService dans shop-core-api

Nouveau module app/v1/Fidelity dans shop-core-api. Relaie le Bearer token OAuth du caissier connecté (request()->bearerToken()), URL de base résolue dynamiquement via le fidelity_domain de la société courante (étape 1).

Aucune exception bloquante : si fidelity-api est indisponible ou répond en erreur, la vente doit continuer sans fidélité (dégradation), jamais bloquer le caissier.

Méthodes prévues dès maintenant : searchCustomer(), prepareSale(), confirmSale(), status() — implémentées au fur et à mesure des chantiers suivants.

Hors scope de ce plan (chantiers suivants, à planifier après validation)

1. Synchronisation Shop / User / ShopUser vers fidelity-api à l'activation d'une boutique ou d'un caissier.
2. Recherche client (email/téléphone) et flux de vente sales/prepare, puis QR/OTP, puis sales/confirm au checkout ; ajout d'une colonne fidelity_sale_id sur la vente shop.
3. Résilience : retry / file d'attente si fidelity-api est indisponible au moment d'une vente.

Nettoyage prévu (tâche de fin de chantier)

Retirer les colonnes society_id déjà présentes côté fidelity-api et le filtrage getSociety() / X-Society-Id associé, puis arrêter d'envoyer ce header depuis shop-core-api.

Vérification

— core-api : migration appliquée, colonnes visibles sur la table societies, l'endpoint société renvoie bien fidelity_domain et fidelity_activated_at (même à vide).
— shop-core-api : php artisan route:list liste les routes du nouveau module Fidelity.
— Test manuel de bout en bout : activer la fidélité pour une société de test, vérifier côté fidelity-api qu'un tenant est bien créé, et que le domaine revient bien s'enregistrer sur la société core-api.
— Test manuel du FidelityService : vérifier que le Bearer token du caissier connecté est bien relayé et accepté par fidelity-api via l'introspection.