BY+ · DOCUMENTATION MODULE

Module POS (Boutique & caisse) — document de définition

Version 2 du 2 octobre 2026 — corrigée à partir de la branche à jour de l'ancien code.

Statut : proposition à valider en atelier avant `php artisan byplus:make-module`.
Références : `docs/catalogue-modules.pdf` (§ 4 : module POS, requiert Catalogue, ancien code « Shop, Pos (branche shop) » ; § 6 : ponts `pos-entrepot`, `pos-facturation`, `pos-fidelite`, `pos-nps`), `docs/architecture.pdf` (règles des tables, 10 règles d'or), `docs/socle-mise-en-place.pdf`.
Ancien code disséqué : dépôt `core-api`, branche `by-plus-shop-mini-staging`, dossiers `app/v1/Shop` (15 modèles, 47 migrations, environ 18 200 lignes) et `app/v1/Pos` (4 contrôleurs, sessions et encaissement, environ 3 150 lignes).

Ce qui change par rapport à la version 1

La version 1 décrivait le Shop de la branche `develop` (6 modèles, 15 migrations, dernière modification du Shop en janvier 2026). La branche à jour est `by-plus-shop-mini-staging` (septembre 2026). Les corrections :

- **AJOUT** Caisses, sessions de vente, paiements multiples, taxes et attributs par ligne, stock par boutique, annulation partielle, ticket PDF et envois, normalisation DGI, téléphones multiples et message de facture : absents de la version 1, présents dans le code à jour.
- **CORRIGÉ** « La caisse n'est exposée nulle part » : vrai sur `develop`, faux sur la branche à jour (22 routes de caisse dans `Shop`, plus l'encaissement dans `Pos`).
- **CORRIGÉ** Le rattachement d'une boutique à une société n'est pas une nouveauté : la colonne `society_id` existe déjà.
- **CORRIGÉ** Tables préfixées `pos_` (règle 6 de l'architecture), clés UUID, devise et pays par leur code.
- **CORRIGÉ** Plus de clé étrangère vers Contacts, plus de vendeur en lien polymorphe, plus de renvoi à une « brique Paramètres » (elle a été retirée du socle).
- **CORRIGÉ** Droits détaillés (13 au lieu de 3) et test d'activation du module (règle 10).

1. Identité (prête pour `module.php`)

```
return [
    'key' => 'pos',
    'name' => 'POS (Boutique & caisse)',
    'version' => '0.1.0',
    'requires' => ['catalogue'],
    'sellable' => true,
    'permissions' => [
        'pos.boutiques.voir',     'pos.boutiques.gerer',
        'pos.membres.gerer',      'pos.caisses.gerer',
        'pos.assortiment.voir',   'pos.assortiment.gerer',
        'pos.sessions.voir',      'pos.sessions.ouvrir',   'pos.sessions.fermer',
        'pos.vendre',             'pos.ventes.voir',       'pos.ventes.annuler',
        'pos.statistiques.voir',
    ],
    'settings' => [
        'prefixe_facture' => 'FACT-',   // valeur par défaut d'une nouvelle boutique
        'prefixe_avoir' => 'FA-',
    ],
    'schedule' => [],
    'provider' => \Modules\Pos\PosServiceProvider::class,
];
```

Test des 4 questions de l'architecture : un commerçant s'en sert seul, avec le Catalogue activé automatiquement ✓ · menu et écrans propres (Boutiques, Caisse, Ventes, Statistiques) ✓ · droits propres ✓ · retirable sans casser un autre module, les ponts se désactivent seuls ✓ → module vendable.

Droits. L'ancien code en compte 28 (`shop_view`, `shop_sale_create`, `shop_cash_box_update`, `shop_sale_open_session`…), un par action et par objet. La version 1 en proposait 3, ce qui ne permet pas de donner la caisse à un vendeur sans lui donner l'annulation d'une vente. Les 13 droits ci-dessus regroupent les 28 par objet, en gardant à part ce qui engage de l'argent : ouvrir et fermer une session, vendre, annuler une vente ou émettre un avoir.

2. Ce que le module comporte

Pas de sous-modules : dix ensembles fonctionnels dans un seul module.

Ensemble	Contenu repris de l'ancien code (branche à jour)
Boutiques	Créer, modifier, archiver une boutique : nom unique, adresse, pays, devise, société de rattachement, logo, téléphone principal et liste de téléphones (avec indicateur WhatsApp), message affiché sur les factures, préfixe et compteur de numérotation, boutique en ligne ou non.
Membres et vendeurs	Utilisateurs rattachés à une boutique avec un profil `responsable` ou `staff`, actifs ou non ; « mes boutiques » pour l'utilisateur connecté ; périmètre des données limité aux boutiques de l'utilisateur.
Caisses AJOUT	Caisses d'une boutique (nom unique par boutique) : créer, modifier, supprimer.
Sessions de vente AJOUT	Un vendeur ouvre une session sur une caisse avec un fond de caisse, vend, puis la ferme ; la session cumule le chiffre d'affaires ; liste des sessions, ventes d'une session, ventes d'une session par produit.
Assortiment de la boutique	Éléments de vente du Catalogue mis en vente dans une boutique : prix propre à la boutique, en ligne ou non, taxes par produit ; ajout produit par produit ou par catégorie ; lots (bundles) vendables en boutique.

Ensemble	Contenu repris de l'ancien code (branche à jour)
Stock de la boutique AJOUT	Stock initial, stock courant, seuil d'alerte et stock de sécurité par produit et par boutique ; ajout de stock ; contrôle de la quantité disponible avant la vente ; un produit sans stock n'est pas proposé à la caisse. Emplacement à trancher, § 8.5.
Encaissement	Ticket `DRAFT` → `PAID` ; lignes avec prix et libellé photographiés ; taxes et attributs (variantes) de chaque ligne AJOUT ; montants HT, taxes, TTC ; **plusieurs paiements par ticket** avec un moyen de paiement chacun AJOUT ; montant reçu et monnaie rendue ; client nommé et son IFU ; devise photographiée ; rattachement à la session de vente.
Annulation et avoirs	Un ticket `DRAFT` seul se modifie ou se supprime ; annuler une vente payée crée un avoir numéroté `FA-…`, lié au ticket d'origine ; l'avoir peut être **partiel** : le ticket passe alors à `PARTIALLY_REVOKED` AJOUT, sinon à `CANCELED`, et l'avoir porte le statut `REVOKED`.
Ticket et envois AJOUT	Ticket ou facture en PDF (voir, télécharger) ; envoi au client par mail ou par WhatsApp ; lien de partage WhatsApp.
Statistiques	Chiffre d'affaires, nombre de ventes, ventes par jour et meilleur jour de la semaine, ventes par catégorie et par produit, statistiques par client et par vendeur, listes paginées des ventes.

3. Hors périmètre (renvois explicites)

Besoin	Où il vit
Produits, variantes, catégories, marques, taxes, lots	Module de base **Catalogue** (requis, `sellable: false`), repris de `app/v1/Product`
Clients (carnet d'adresses)	Module **Contacts** — lien à trancher, § 8.1
Dépôts, mouvements de stock, inventaire	Module **Entrepôt** et pont `pos-entrepot`, qui écoute `VenteEffectuee` (ancienne table `shop_stores`)
Facture à partir d'un ticket, acomptes, relances	Module **Facturation** et pont `pos-facturation`
Points de fidélité · enquête après achat	Ponts `pos-fidelite` et `pos-nps`
Normalisation fiscale DGI Bénin (SYGMEF) AJOUT	À trancher, § 8.6. Dans l'ancien code : `app/v1/Normalization` et `ShopCheckoutService`
Envoi par WhatsApp AJOUT	Un canal à ajouter à la brique **Notifications** du socle (l'ancien code a quatre fournisseurs dans `Shop/Service`)
Connexion par téléphone et code (OTP), changement de téléphone, suppression de compte AJOUT	Brique **Auth** du socle : ce sont des comptes, pas de la vente
Inscription d'une boutique depuis le mobile AJOUT	Elle crée un client (`CreateTenantRegisterJob`) : c'est la **console** NetByUs, hors périmètre pour l'instant
Synchronisation hors ligne	Brique **Sync** du socle (phase 3). En attendant : adresses `pull` / `push` propres au module, comme avant

4. Entités et tables (base de chaque client)

Règles appliquées à toutes les tables : préfixe `pos_` ; clé primaire UUID (`uuid('id')->primary()` et trait `HasUuids`) ; clés étrangères seulement vers le socle ou vers le Catalogue ; devise et pays par leur code (briques Currencies et Countries), plus par un identifiant.

Modèle (`Modules\Pos\Models\`)	Table (ancienne table)	Colonnes principales et changements
`Shop`	`pos_shops` shops	`name` unique, `address`, `phone_code`, `phone`, `phones` (JSON), `invoice_message`, `prefix`, `invoice_number` (compteur), `online`, `last_sync_at`, archivage. **Changé :** `company_id` remplace `society_id` ; `currency_code` et `country_code` remplacent `currency_id` et `country_id` ; logo par `HasAttachments` ; le lien polymorphe `event` n'est pas repris.
`ShopMember`	`pos_shop_members` shop_users, shop_participators	`shop_id`, `user_id`, `profile` (`responsable`, `staff`), `active`. **Changé :** `user_id` pointe vers `core_users` ; plus de lien polymorphe ni de second type de compte (§ 8.2).
`CashRegister` **AJOUT**	`pos_cash_registers` cash_registers	`shop_id`, `name` (unique par boutique).
`SaleSession` **AJOUT**	`pos_sale_sessions` shop_sale_sessions	`shop_id`, `cash_register_id`, `operator_id` (utilisateur), `opened_at`, `closed_at`, `starting_cash` (fond de caisse), `sales_turnover`.
`ShopProduct`	`pos_shop_products` shop_products	`shop_id`, `product_sale_element_id` (Catalogue), `price` décimal, `online`, et le stock **AJOUT** : `stock_initial`, `stock_variable`, `stock_alert`, `stock_security` (§ 8.5).
`ShopProductTax`	`pos_shop_product_taxes` shop_product_taxes	`shop_product_id`, `tax_id` (taxe du Catalogue).
`Checkout`	`pos_checkouts` shop_checkouts	`number` (`FACT-…`, `FA-…` pour un avoir), `ref_invoice_number`, `status` (`DRAFT`, `PAID`, `CANCELED`, `REVOKED`, `PARTIALLY_REVOKED`), `shop_id`, `sale_session_id`, `seller_id` (utilisateur), `customer_name`, `ifu`, `note`, `amount_ht`, `taxes`, `amount_ttc`, `received_amount`, `change_amount`, `currency_code`. **Changé :** pas de `customer_id` (§ 8.1) ; vendeur par `seller_id` au lieu d'un lien polymorphe ; code de devise au lieu de son nom et de son symbole. Les colonnes de normalisation (`is_normalized`, `normalization_data`, `normalized_at`) suivent la décision § 8.6.
`CheckoutLine`	`pos_checkout_lines` shop_checkout_product_sale_elements	`checkout_id`, `product_sale_element_id`, `name` (photo), `quantity` décimal, `price`.
`CheckoutLineTax` **AJOUT**	`pos_checkout_line_taxes` shop_checkout_product_taxes	`checkout_line_id`, `tax`, `tax_title`, `tax_type` (pourcentage ou montant), `is_dgi` : la taxe est photographiée sur la ligne.
`CheckoutLineAttribute` **AJOUT**	`pos_checkout_line_attributes` shop_checkout_product_attributes	`checkout_line_id`, `attribute`, `attribute_value` : la variante vendue, photographiée.

Modèle (`Modules\Pos\Models\`)	Table (ancienne table)	Colonnes principales et changements
`CheckoutPayment` **AJOUT**	`pos_checkout_payments` shop_checkout_payments	`checkout_id`, moyen de paiement (§ 8.4), `amount`. Un ticket a un ou plusieurs paiements.

Non reprises : `shop_checkout_currencies` (remplacée par `currency_code` sur le ticket) ; `shop_stores` (pont `pos-entrepot`) ; `account_deletion_requests` (brique Auth) ; `shop_product_dgi_taxes` (suit la décision § 8.6) ; `shop_checkout_taxes` (créée puis supprimée dans l'ancien code).

Règles reprises : seul un ticket `DRAFT` se modifie ou se supprime ; la validation passe à `PAID` et augmente le chiffre d'affaires de la session ; un avoir, total ou partiel, est un nouveau ticket lié à l'original par `ref_invoice_number` ; les prix, libellés, taxes, variantes et la devise sont photographiés sur le ticket ; une vente n'est possible qu'avec une session ouverte ; la quantité vendue ne dépasse pas le stock de la boutique, sauf pour un service.

5. Vitrine `Public/` (ce que les ponts et les autres modules utilisent)

```
modules/Pos/Public/
├── Events/VenteEffectuee.php        # un ticket passe à PAID : boutique, lignes, montants,
vendeur
├── Events/VenteAnnulee.php          # un avoir, total ou partiel, vient d'être émis
├── Events/SessionCloturee.php       # une session de vente est fermée, avec son chiffre
d'affaires
├── Contracts/PosContext.php         # les boutiques de l'utilisateur connecté, sa session
ouverte
└── Hooks/NumerotationTicket.php     # point d'extension : numéro du prochain ticket d'une
boutique
```

Aucun `use Modules\Pos\Models\…` hors du module. Les ponts écoutent `VenteEffectuee` et `VenteAnnulee` avec un écouteur qui étend `Core\Modules\Public\ModuleListener` (méthode `whenActive`) : le pont `pos-entrepot` sort le stock à la vente et le remet à l'annulation.

CORRIGÉ Le point d'extension `RemiseVendeur` de la version 1 est retiré : l'ancien code n'a pas de remise. Un point d'extension se crée quand un besoin réel d'un client apparaît (architecture, chapitre 9).

6. Droits, écrans, routes

- **Écrans :** Boutiques · Caisse · Ventes · Sessions · Statistiques, plus un onglet ajouté par chaque pont actif.
- **Routes :** `modules/Pos/routes/tenant.php`, servies sous `/api/v1/pos` sur le sous-domaine du client. Le socle y pose trois gardes : le client (tenancy), la connexion (`RequireUser`, 401 sinon) et le module (`module:pos`, 403 sinon). Chaque action vérifie ensuite son droit par `$this->authorize('pos.…')`.
- **Un seul type de compte :** un vendeur est un utilisateur du socle (`core_users`), membre d'une boutique. Les gardes `auth:api` et « participator » de l'ancien code n'existent plus.
- **Périmètre :** un utilisateur ne voit que les boutiques dont il est membre, dans les sociétés qui lui sont accessibles ; celui qui a tous les droits les voit toutes.

Adresse	Rôle · droit
`GET·POST /pos/shops`, `GET·PATCH·DELETE /pos/shops/{id}`	Boutiques · `pos.boutiques.voir`, `pos.boutiques.gerer`
`GET·POST·DELETE /pos/shops/{id}/members`	Membres et profils · `pos.membres.gerer`

Adresse	Rôle · droit
`GET·POST /pos/shops/{id}/cash-registers`, `PATCH·DELETE …/{id}`	Caisses · `pos.caisses.gerer`
`GET·POST /pos/shops/{id}/products`, `PATCH·DELETE …/{id}`	Assortiment, prix, stock · `pos.assortiment.voir`, `pos.assortiment.gerer`
`POST /pos/sessions`, `POST /pos/sessions/{id}/close`, `GET /pos/sessions`	Ouvrir, fermer, lister · `pos.sessions.ouvrir`, `pos.sessions.fermer`, `pos.sessions.voir`
`GET /pos/sale/products`, `POST /pos/checkouts`	Ce que la caisse peut vendre, encaisser · `pos.vendre`
`GET /pos/checkouts`, `GET /pos/checkouts/{id}`, `…/pdf`, `…/send`	Ventes, ticket PDF, envoi · `pos.ventes.voir`
`POST /pos/checkouts/{id}/credit-note`	Annuler une vente, avoir total ou partiel · `pos.ventes.annuler`
`GET /pos/statistics/…`	Statistiques · `pos.statistiques.voir`

Ces adresses remplacent celles de l'ancien code (`get/shops`, `store/shop`, `pos/open/session`, `pos/checkout`…). Les applications web et mobile actuelles devront être adaptées : décision § 8.7.

7. Dépendances au socle (guichets, jamais les modèles privés)

Brique	Guichet ou trait	Usage par POS
Kernel	`TenantController`, `Context`	Contrôleurs du module, utilisateur et société de la requête, format des réponses et des erreurs
Auth	Modèle `User` (lecture), droits `pos.*`	Vendeurs et membres ; les droits du manifeste rejoignent le catalogue des droits tout seuls
Companies	`CompanyDirectory`, `CurrentCompany`, trait `BelongsToCompany`	Une boutique appartient à une société ; l'ancienne colonne `society_id` devient `company_id`
Files	`Storage`, `Attachments`, trait `HasAttachments`	Logo de la boutique, tickets PDF (remplace l'ancien `FileManager`)
Audit	`Journal`	Chaque action est journalisée : boutique, session ouverte et fermée, vente, avoir
Notifications	`Notifier`	Ticket envoyé au client par mail ; par WhatsApp quand le canal existera
Currencies, Countries	`Currencies`, `Countries`	Devise de la boutique et du ticket, pays et indicatif téléphonique
Units	`Units`	Unité des quantités vendues, si le Catalogue en porte une
Modules	`ModuleListener`, les cinq portes	Routes, traitements en file et écouteurs du module sont refusés au client qui ne l'a pas
Sync (phase 3)	—	Transport `pull` / `push` ; en attendant, adresses propres au module

CORRIGÉ La version 1 renvoyait la numérotation à une « brique Paramètres ». Cette brique a été retirée du socle (décision sur le guide de mise en place) : la numérotation reste dans le module.

8. Décisions à trancher en atelier

1. **Clients nommés.** POS ne requiert que le Catalogue : une table `pos_` ne peut pas porter de clé étrangère vers Contacts (règle 6). *Recommandation :* POS seul garde `customer_name` et `ifu` en texte, comme avant ; si le client a Contacts, un pont `pos-contacts`, à ajouter au catalogue, porte le lien ticket-contact dans sa propre table et ajoute l'onglet « Achats » sur la fiche du contact.
2. **Comptes des vendeurs.** L'ancien code a deux sortes de comptes et une connexion mobile par téléphone et code. *Recommandation :* un seul type de compte, celui du socle ; un vendeur est un utilisateur membre d'une boutique. Si l'application mobile doit garder la connexion par téléphone, elle s'ajoute à la brique Auth, pour tous les modules.
3. **Numérotation.** Reprise telle quelle : préfixe et compteur par boutique, `FACT-` et `FA-` par défaut. À corriger au passage : l'ancien calcul (`'FACT-' + nombre de tickets`) peut donner deux fois le même numéro si deux ventes arrivent en même temps ; le compteur doit être incrémenté de façon sûre.
4. **Moyens de paiement.** CORRIGÉ La branche à jour a plusieurs paiements par ticket, chacun avec un moyen de paiement tiré d'une liste (ancien module `Setting`). *À décider :* cette liste devient une table du module (`pos_payment_types`), ou une liste commune que Facturation utilisera aussi.
5. **Stock.** AJOUT La branche à jour tient un stock par boutique et refuse une vente sans stock. Le catalogue confie le stock au module Entrepôt. *Deux voies :* (a) garder un stock simple dans POS pour ne pas dépendre d'Entrepôt, au prix d'une donnée tenue à deux endroits le jour où Entrepôt arrive ; (b) POS sans stock, et le pont `pos-entrepot` dès la première version. *Recommandation :* (a) pour livrer vite, avec une règle écrite : dès qu'un client a Entrepôt, c'est lui qui fait foi.
6. **Normalisation DGI.** AJOUT Absente de la version 1, elle est utilisée en production au Bénin. *À décider :* dans la première version ou juste après, et son emplacement : partie de POS, ou module de base partagé avec Facturation, qui en aura besoin aussi.
7. **Adresses de l'API.** Nouvelles adresses REST (§ 6), ou adresses compatibles avec les applications actuelles le temps de la bascule.
8. **Ticket PDF.** Il faut un paquet de génération de PDF : une dépendance à valider.
9. **Clé du module.** `pos`, cohérente avec les ponts `pos-*` du catalogue, plutôt que `shop`.

9. Ordre d'intégration recommandé

1. **Valider ce document** en atelier, et reporter les décisions dans le catalogue des modules.
2. **Mettre à jour le gabarit** `byplus:make-module` : test d'activation généré, contrôleur de base, droits. Le guide de mise en place le prévoit avant le premier module.
3. **Module de base Catalogue** : produits, éléments de vente, catégories, taxes, repris de `app/v1/Product` (environ 10 700 lignes). POS ne peut rien vendre sans lui.
4. **POS, lot 1 :** boutiques, membres, caisses.
5. **POS, lot 2 :** assortiment, prix et stock de la boutique.
6. **POS, lot 3 :** sessions de vente et encaissement, avec les paiements.
7. **POS, lot 4 :** annulation et avoirs, ticket PDF, envois, statistiques.
8. **Ensuite :** ponts `pos-entrepot`, `pos-facturation`, `pos-fidelite`, `pos-nps` au fil des modules voisins ; normalisation DGI ; synchronisation.
9. **Reprise des données** des clients existants : `society_id` → `company_id`, identifiants de devise et de pays → codes, vendeurs → utilisateurs du socle, tickets repris avec leurs montants photographiés.

> **Terminé quand**
>
> Le **test d'activation** du module passe (règle 10 de l'architecture) : sur un client neuf, on active POS, ce qui active le Catalogue ; les écrans de POS répondent, ceux des autres modules sont refusés ; un vendeur ouvre une session, encaisse une vente à deux paiements, émet un avoir partiel et ferme sa session. Les tests du module sont dans `modules/Pos/tests` ; PHPStan et Deptrac sont verts.

Version 1 : document de définition du 2 octobre 2026, établi sur la branche `develop`. Version 2 : même plan, contenu revu sur la branche `by-plus-shop-mini-staging` et aligné sur les règles du socle en place sur `new-develop`.