BY+ · DOCUMENTATION MODULE

Module POS (Boutique & caisse) — document de définition

Version finale du 2 octobre 2026.

Statut : version finale, à suivre pour `php artisan byplus:make-module pos`. Les décisions du § 8 sont celles retenues pour la version 0.1.0 du module ; l'atelier peut en modifier une, en mettant ce document à jour.
Références : `docs/catalogue-modules.pdf` (§ 4 : module POS, requiert Catalogue ; § 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 de référence : dépôt `core-api`, branche `by-plus-shop-mini-staging` (la branche à jour), dossiers `app/v1/Shop` (15 modèles, 47 migrations, environ 18 200 lignes) et `app/v1/Pos` (sessions et encaissement, environ 3 150 lignes). Il sert de référence métier, pas de base à copier.

> **En bref**
>
> - **Un module**, clé `pos`, vendable, qui requiert le module de base Catalogue.
> - **Onze ensembles fonctionnels** : boutiques, membres, caisses, sessions de vente, assortiment, stock de la boutique, encaissement, annulation et avoirs, ticket et envois, normalisation fiscale, statistiques.
> - **Douze tables** préfixées `pos_`, en UUID, dans la base de chaque client.
> - **Treize droits**, vérifiés sur chaque adresse de l'API, servie sous `/api/v1/pos`.
> - **Hors du module :** les produits (Catalogue), les clients (Contacts), les dépôts (Entrepôt), les comptes et la connexion (socle), l'inscription d'une boutique (console).

1. Identité (prête pour `module.php`)

```
return [
    'key' => 'pos',
    'name' => 'POS (Boutique & caisse)',
    'version' => '0.1.0',
    'requires' => ['catalogue'],
    'sellable' => true,
    'permissions' => [
        'pos.shops.view',           'pos.shops.manage',
        'pos.members.manage',       'pos.cash_registers.manage',
        'pos.products.view',        'pos.products.manage',
        'pos.sessions.view',        'pos.sessions.open',     'pos.sessions.close',
        'pos.sell',                 'pos.sales.view',        'pos.sales.cancel',
        'pos.statistics.view',
    ],
    'settings' => [
        'invoice_prefix' => 'FACT-',       // valeur par défaut d'une nouvelle boutique
        'credit_note_prefix' => '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, Sessions, Statistiques) ✓ · droits propres ✓ · retirable sans casser un autre module, les ponts se désactivent seuls ✓ → module vendable.

Droits. Seule la clé d'un droit est en anglais : c'est un identifiant, jamais affiché. Son libellé est traduit dans `modules/Pos/lang/fr.json` et `en.json` (`pos.sales.cancel` → « Annuler une vente » / « Cancel a sale ») ; l'API le renvoie dans la langue courante, et un droit sans traduction affiche sa clé. 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. Les 13 droits du module les regroupent 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. On peut ainsi donner la caisse à un vendeur sans lui donner l'annulation.

2. Ce que le module comporte

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

Ensemble	Contenu
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 `manager` (responsable) ou `staff`, actifs ou non ; « mes boutiques » pour l'utilisateur connecté ; les données visibles se limitent aux boutiques dont l'utilisateur est membre.
Caisses	Caisses d'une boutique, au nom unique dans la boutique : créer, modifier, supprimer.
Sessions de vente	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) du Catalogue vendables en boutique.
Stock de la boutique	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 ; un service n'a pas de stock.
Encaissement	Ticket `DRAFT` → `PAID` ; lignes avec prix et libellé photographiés ; taxes et variantes de chaque ligne ; montants HT, taxes, TTC ; un ou plusieurs paiements par ticket, chacun avec son moyen de paiement ; montant reçu et monnaie rendue ; client nommé et son IFU ; devise photographiée ; rattachement à la session de vente.
Annulation et avoirs	Seul un ticket `DRAFT` 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`, sinon à `CANCELED` ; l'avoir porte le statut `REVOKED`.
Ticket et envois	Ticket ou facture en PDF (voir, télécharger) ; envoi au client par mail ; envoi et lien de partage par WhatsApp quand le canal existe dans le socle.
Normalisation fiscale	Pour les boutiques qui l'activent : envoi de la vente et de l'avoir à la DGI du Bénin (SYGMEF), groupe de taxe DGI par produit, conservation de la réponse sur le ticket, normalisation à la validation ou relancée à la main.
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, groupes de taxe DGI, lots	Module de base **Catalogue** (requis, `sellable: false`), repris de `app/v1/Product`
Clients (carnet d'adresses)	Module **Contacts** et pont `pos-contacts` (§ 8.1)
Dépôts, mouvements de stock, inventaire	Module **Entrepôt** et pont `pos-entrepot`, qui écoute `SaleCompleted` (§ 8.5)
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`
Envoi par WhatsApp	Un canal à ajouter à la brique **Notifications** du socle ; POS l'utilise par le guichet `Notifier`
Connexion par téléphone et code (OTP), changement de téléphone, suppression de compte	Brique **Auth** du socle : ce sont des comptes, pas de la vente
Inscription d'une boutique depuis le mobile	Elle crée un client : c'est la **console** NetByUs
Synchronisation hors ligne	Brique **Sync** du socle (phase 3). En attendant : adresses `pull` / `push` propres au module

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).

Modèle (`Modules\Pos\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Shop`	`pos_shops` shops	`name` unique, `address`, `phone_code`, `phone`, `phones` (JSON), `invoice_message`, `prefix`, `invoice_number` (compteur), `online`, `is_normalization_enabled`, `dgi_tax_id` (groupe DGI par défaut), `last_sync_at`, archivage. **Change :** `company_id` remplace `society_id` ; `currency_code` et `country_code` remplacent les identifiants ; logo par `HasAttachments` ; le lien polymorphe `event` n'est pas repris.
`ShopMember`	`pos_shop_members` shop_users, shop_members	`shop_id`, `user_id`, `profile` (`manager`, `staff`), `active`. **Change :** une seule table et un seul type de compte ; `user_id` pointe vers `core_users` (§ 8.2).
`CashRegister`	`pos_cash_registers` cash_registers	`shop_id`, `name` (unique par boutique).
`SaleSession`	`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`.

Modèle (`Modules\Pos\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`ShopProduct`	`pos_shop_products` shop_products	`shop_id`, `product_sale_element_id` (Catalogue), `price` décimal, `online`, `dgi_tax_id`, `stock_initial`, `stock_variable`, `stock_alert`, `stock_security`.
`ShopProductTax`	`pos_shop_product_taxes` shop_product_taxes	`shop_product_id`, `tax_id` (taxe du Catalogue).
`PaymentType`	`pos_payment_types` payment_types (Setting)	`name`, `active`. Livrée avec Espèces, Carte et Paiement mobile ; le client en ajoute (§ 8.4).
`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`, `is_normalized`, `normalization_data`, `normalized_at`. **Change :** pas de `customer_id` (§ 8.1) ; vendeur par `seller_id` au lieu d'un lien polymorphe ; code de devise au lieu d'une table de devises du ticket.
`CheckoutLine`	`pos_checkout_lines` shop_checkout_product_sale_elements	`checkout_id`, `product_sale_element_id`, `name` (photo), `quantity` décimal, `price`.
`CheckoutLineTax`	`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`	`pos_checkout_line_attributes` shop_checkout_product_attributes	`checkout_line_id`, `attribute`, `attribute_value` : la variante vendue, photographiée.
`CheckoutPayment`	`pos_checkout_payments` shop_checkout_payments	`checkout_id`, `payment_type_id`, `amount`. Un ticket a un ou plusieurs paiements.

Anciennes tables non reprises : `shop_checkout_currencies` (remplacée par `currency_code` sur le ticket) ; `shop_stores` (pont `pos-entrepot`) ; `shop_product_dgi_taxes` (remplacée par `dgi_tax_id` sur le produit en boutique) ; `account_deletion_requests` (brique Auth) ; `shop_checkout_taxes` (créée puis supprimée dans l'ancien code).

Règles métier reprises :

- Une vente n'est possible qu'avec une session ouverte ; la validation passe le ticket à `PAID` et augmente le chiffre d'affaires de la session.
- Seul un ticket `DRAFT` se modifie ou se supprime.
- Un avoir, total ou partiel, est un nouveau ticket lié à l'original par `ref_invoice_number`.
- Prix, libellés, taxes, variantes et devise sont photographiés sur le ticket : une vente passée ne change pas quand le Catalogue change.
- La quantité vendue ne dépasse pas le stock de la boutique, sauf pour un service.
- La somme des paiements d'un ticket couvre son montant TTC.

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

```
modules/Pos/Public/
├── Events/SaleCompleted.php        # un ticket passe à PAID
├── Events/SaleCancelled.php        # un avoir, total ou partiel, est émis
├── Events/SaleSessionClosed.php    # une session de vente est fermée
├── Contracts/PosContext.php        # boutiques et session de l'utilisateur
└── Hooks/TicketNumbering.php       # numéro du prochain ticket d'une boutique
```

Les classes sont nommées en anglais, comme tout le code ; les textes affichés, eux, sont traduits en français et en anglais. `SaleCompleted` porte la boutique, les lignes, les montants et le vendeur ; `SaleSessionClosed` porte le chiffre d'affaires de la session. Aucun `use Modules\Pos\Models\…` hors du module. Les ponts écoutent `SaleCompleted` et `SaleCancelled` 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.

Un point d'extension ne se crée que pour un besoin réel d'un client (architecture, chapitre 9). La numérotation en est un : certains clients imposent leur propre format de numéro.

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.
- **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 (sous `/api/v1`)	Rôle · droit
`GET·POST /pos/shops`, `GET·PATCH·DELETE /pos/shops/{id}`	Boutiques · `pos.shops.view`, `pos.shops.manage`
`GET·POST·DELETE /pos/shops/{id}/members`	Membres et profils · `pos.members.manage`
`GET·POST /pos/shops/{id}/cash-registers`, `PATCH·DELETE …/{id}`	Caisses · `pos.cash_registers.manage`
`GET·POST /pos/payment-types`, `PATCH …/{id}`	Moyens de paiement · `pos.shops.manage`
`GET·POST /pos/shops/{id}/products`, `PATCH·DELETE …/{id}`, `POST …/{id}/stock`	Assortiment, prix, stock · `pos.products.view`, `pos.products.manage`
`POST /pos/sessions`, `POST /pos/sessions/{id}/close`, `GET /pos/sessions`	Ouvrir, fermer, lister · `pos.sessions.open`, `pos.sessions.close`, `pos.sessions.view`
`GET /pos/sale/products`, `POST /pos/checkouts`	Ce que la caisse peut vendre, encaisser · `pos.sell`
`GET /pos/checkouts`, `GET /pos/checkouts/{id}`, `…/pdf`, `POST …/send`	Ventes, ticket PDF, envoi · `pos.sales.view`
`POST /pos/checkouts/{id}/credit-note`	Annuler une vente, avoir total ou partiel · `pos.sales.cancel`
`POST /pos/checkouts/{id}/normalize`	Relancer la normalisation DGI · `pos.sell`
`GET /pos/statistics/…`	Statistiques · `pos.statistics.view`

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 seuls le catalogue des droits
Companies	`CompanyDirectory`, `CurrentCompany`, trait `BelongsToCompany`	Une boutique appartient à une société
Files	`Storage`, `Attachments`, trait `HasAttachments`	Logo de la boutique, tickets PDF
Audit	`Journal`	Chaque action est journalisée : boutique, session ouverte et fermée, vente, avoir
Notifications	`Notifier`	Ticket envoyé au client par mail, puis par WhatsApp
Currencies, Countries	`Currencies`, `Countries`	Devise de la boutique et du ticket, pays et indicatif téléphonique
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

Dépendance à installer : `barryvdh/laravel-dompdf`, le paquet de génération de PDF que l'ancien code utilise déjà pour les tickets.

8. Décisions retenues pour la version 0.1.0

#	Sujet	Décision	Pourquoi
1	**Clients nommés**	Le ticket garde `customer_name` et `ifu` en texte. Le lien avec un contact passe par un pont `pos-contacts`, à ajouter au catalogue, qui porte sa propre table et l'onglet « Achats » de la fiche contact.	POS ne requiert que le Catalogue : une table `pos_` ne peut pas pointer vers Contacts (règle 6).
2	**Comptes des vendeurs**	Un seul type de compte, celui du socle. Un vendeur est un utilisateur membre d'une boutique. Ses droits viennent de ses rôles ; la boutique limite ce qu'il voit.	Le socle n'a qu'une sorte de compte. La connexion par téléphone, si le mobile la garde, s'ajoute à la brique Auth pour tous les modules.
3	**Numérotation**	Préfixe et compteur par boutique, `FACT-` et `FA-` par défaut. Le compteur est incrémenté sous verrou.	L'ancien calcul (préfixe + nombre de tickets) peut donner deux fois le même numéro quand deux ventes arrivent en même temps.
4	**Moyens de paiement**	Une table du module, `pos_payment_types`, livrée avec Espèces, Carte et Paiement mobile.	L'ancienne liste vivait dans les paramètres généraux, qui n'existent plus. Si Facturation a besoin de la même liste, elle deviendra un module de base commun.

#	Sujet	Décision	Pourquoi
5	**Stock**	Stock simple tenu par POS, par produit et par boutique, comme dans l'ancien code. Dès qu'un client a Entrepôt, le pont `pos-entrepot` prend le relais et c'est Entrepôt qui fait foi.	POS doit pouvoir se vendre seul et refuser une vente sans stock, sans attendre le module Entrepôt.
6	**Normalisation DGI**	Dans POS, activable par boutique, livrée dans le dernier lot du module.	Elle est utilisée en production au Bénin. Elle sera sortie dans un module de base le jour où Facturation en aura besoin.
7	**Adresses de l'API**	Nouvelles adresses REST sous `/api/v1/pos` (§ 6). Les anciennes (`get/shops` , `pos/checkout` …) ne sont pas reprises.	Toute l'API du nouveau socle suit cette forme. Les applications web et mobile sont adaptées module par module.
8	**Ticket PDF**	Généré avec `barryvdh/laravel-dompdf` , enregistré par la brique Files.	C'est le paquet de l'ancien code : les modèles de ticket se reprennent sans réécriture.
9	**Clé du module**	`pos` .	Cohérente avec le catalogue et ses ponts `pos-*` .

9. Ordre d'intégration

1. **Mettre à jour le gabarit** `byplus:make-module` : test d'activation généré et contrôleur de base. Les droits `{clé}.view` / `{clé}.manage` et leurs libellés traduits sont déjà générés.
2. **Module de base Catalogue** : produits, éléments de vente, catégories, taxes, groupes de taxe DGI, repris de `app/v1/Product` (environ 10 700 lignes). POS ne peut rien vendre sans lui.
3. **POS, lot 1 :** boutiques, membres, caisses, moyens de paiement.
4. **POS, lot 2 :** assortiment, prix et stock de la boutique.
5. **POS, lot 3 :** sessions de vente et encaissement, avec les paiements.
6. **POS, lot 4 :** annulation et avoirs, ticket PDF, envoi par mail, statistiques.
7. **POS, lot 5 :** normalisation DGI.
8. **Ensuite :** ponts `pos-contacts` , `pos-entrepot` , `pos-facturation` , `pos-fidelite` , `pos-nps` au fil des modules voisins ; canal WhatsApp dans Notifications ; synchronisation hors ligne.
9. **Reprise des données** des clients existants : `society_id` → `company_id` , identifiants de devise et de pays → codes, vendeurs → utilisateurs du socle, profil `responsable` → `manager` , tickets repris avec leurs montants photographiés.

Chaque lot est livré avec ses tests dans `modules/Pos/tests` , vérifié sur une vraie base client, puis fusionné : le module n'attend pas le lot 5 pour être utilisable.

> **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 ; un utilisateur sans le droit `pos.sales.cancel` ne peut pas émettre l'avoir. PHPStan et Deptrac sont verts.