FLARAVEL

PLATEFORME RESTAURANT

Plus qu'une caisse, une plateforme complète pour votre restaurant

MANUEL D'ARCHITECTURE

Architecture de la plateforme FLARAVEL

Backend · Mobile · Livraison · Caisse

ONLINE & OFFLINE
Continuez à travailler même sans connexion Internet.



Gestion Centrale

Stock
Suivi en temps réel de vos produits

Comptabilité
Vos finances sous contrôle

RH
Gestion de votre équipe

FLARAVEL · VOTRE RESTAURANT, NOTRE PRIORITÉ

© Caisse Pro | Architecture de la plateforme | i

Sommaire

Plateforme Flaravel — Architecture et relations

Un manuel complémentaire au guide Caisse Pro pour comprendre les quatre applications, les échanges de données et l'exploitation d'une installation.

Avant propos et vue d'ensemble....2 à 3

Installation par entreprise et responsabilités....4 à 5

Backend données authentification et API....6 à 9

Application client et parcours de commande....10 à 11

Prix catalogue et réservations....12 à 14

Architecture et travail de la caisse....15 à 16

Synchronisation identifiants et paiements....17 à 19

Application livreur affectation et QR....20 à 22

Livraison retour et actions hors connexion....23 à 24

Rapprochement de la recette du livreur....25

Site web rapports et actualisation....26 à 28

Sécurité réseau et mise en service....29 à 31

Déploiement sauvegarde et maintenance....32 à 33

Dépannage et recette complète....34 à 35

Exemple complet et vocabulaire....36 à 37

Repères dans les sources....38 à 39

Périmètre vérifié et édition du manuel....40

Choisir son parcours

Gérant et responsable de restaurant : vue d'ensemble, parcours métier, rapports et dépannage.
Intégrateur et équipe technique : backend, API, identités, synchronisation, sécurité et annexes de sources. Les NB et les points de vigilance reprennent les couleurs du manuel Caisse Pro.

Avant propos

Dirigeant gérant responsable technique et équipe de déploiement

Comprendre une seule plateforme à plusieurs entrées

La plateforme Flaravel réunit la gestion centrale, la vente en restaurant, la commande client et le travail des livreurs. Ce manuel explique le rôle de chaque application, les informations qu'elle conserve et la manière dont les opérations circulent entre elles. Il complète le guide utilisateur Caisse Pro, qui décrit les boutons et les gestes du personnel en détail.

Le backend Laravel constitue le point de rencontre des applications connectées. La caisse garde aussi une base locale pour son fonctionnement sur le terminal. L'application client et l'application livreur ont des responsabilités différentes : commander pour la première, recevoir et traiter les livraisons affectées pour la seconde.

Lire le document selon son besoin

Un gérant peut commencer par la vue d'ensemble, les rôles et les parcours métier, puis consulter les pages sur les rapports et le dépannage. Un intégrateur suivra aussi les chapitres sur les API, les identifiants, les files de synchronisation et la configuration des serveurs. Les annexes donnent les emplacements du code correspondant aux explications.

Les diagrammes montrent des relations et des flux vérifiés dans les sources. Les captures proviennent des images déjà fournies pour la caisse. Les exemples de montants, d'identifiants et de restaurants sont fictifs et servent uniquement à comprendre les mécanismes.

NB. Ce manuel décrit les sources présentes le 19 octobre 2026. La lecture du code confirme l'existence des composants ; la validation d'une installation réelle demande aussi la recette décrite en fin de document. Une connexion locale, une imprimante ou un téléphone n'ont pas été testés pendant la rédaction.

Repère de lecture. Vert émeraude : plateforme et parcours communs. Indigo : applications, caisse et échanges. Ambre : gestion et administration. Rouge : points de vigilance métier.

Sources : B01 B02 C01 M01 L01.

Vue générale de la plateforme

Les quatre applications principales et le site web

Le backend relie les différents métiers

Le projet **backend_platform** expose les API, le back-office d'administration et les pages web de commande. Il reçoit les demandes, contrôle les droits et applique les règles métier avant de persister les informations. Les autres applications communiquent avec lui par des requêtes HTTP et des réponses JSON.

soyabox_mobile sert au client : découverte des restaurants, catalogue, panier, commande, réservations et avantages. **caisse_soya** sert à l'équipe du restaurant : tables, saisie des commandes, encaissement, impressions, sessions et synchronisation. **livraison_app** sert au livreur : liste des affectations, scan du QR, réception, livraison ou retour.

Les applications ne partagent pas directement un fichier de base de données. La caisse utilise sa propre base locale ; les informations centrales sont conservées par Laravel. Un changement devient visible ailleurs après un échange réussi et l'actualisation de la vue concernée.

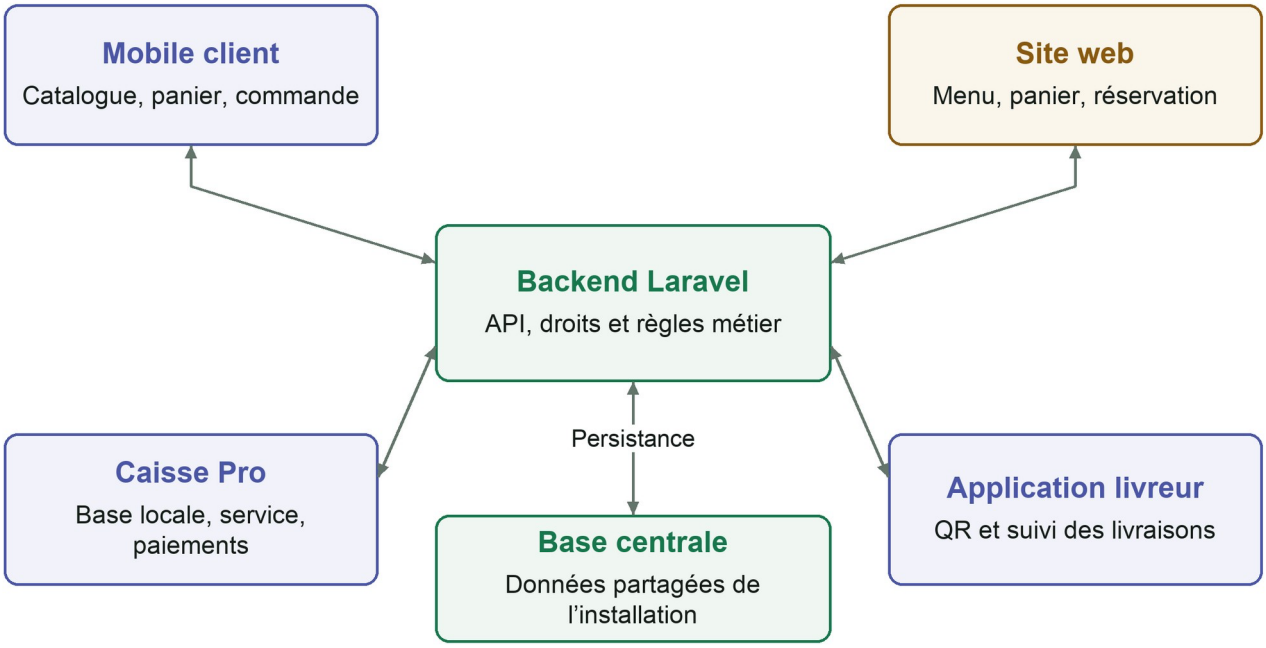


Figure 1 Les applications communiquent avec le backend central

À retenir. Une commande peut être créée par plusieurs canaux, mais elle doit garder une identité et un restaurant cohérents pendant tout son traitement. Une notification ou un ticket accompagne cette opération ; la preuve de son état reste l'enregistrement correspondant.

Sources : B01 B02 C01 C03 M01 M02 L01 L02.

Une installation par entreprise

Organisation et séparation des restaurants

Une entreprise peut gérer plusieurs restaurants

L'architecture de référence retient une installation indépendante par entreprise cliente : son backend, sa configuration, ses comptes et ses données. À l'intérieur de cette installation, plusieurs restaurants peuvent fonctionner ensemble. Les applications d'une entreprise doivent pointer vers le backend qui lui correspond.

Le code actuel organise les principales données opérationnelles autour de **restaurant_id**. Les produits, catégories, tables, commandes, réservations, appareils, livreurs et sessions de caisse disposent de leur périmètre restaurant. Les comptes utilisateurs peuvent être associés aux restaurants par leurs appartenances.

Le mot « entreprise » décrit ici l'organisation propriétaire de l'installation. Certains noms historiques de rôles ou de classes subsistent dans le projet ; ils ne suffisent pas à démontrer une plateforme SaaS à plusieurs entreprises dans une base commune.

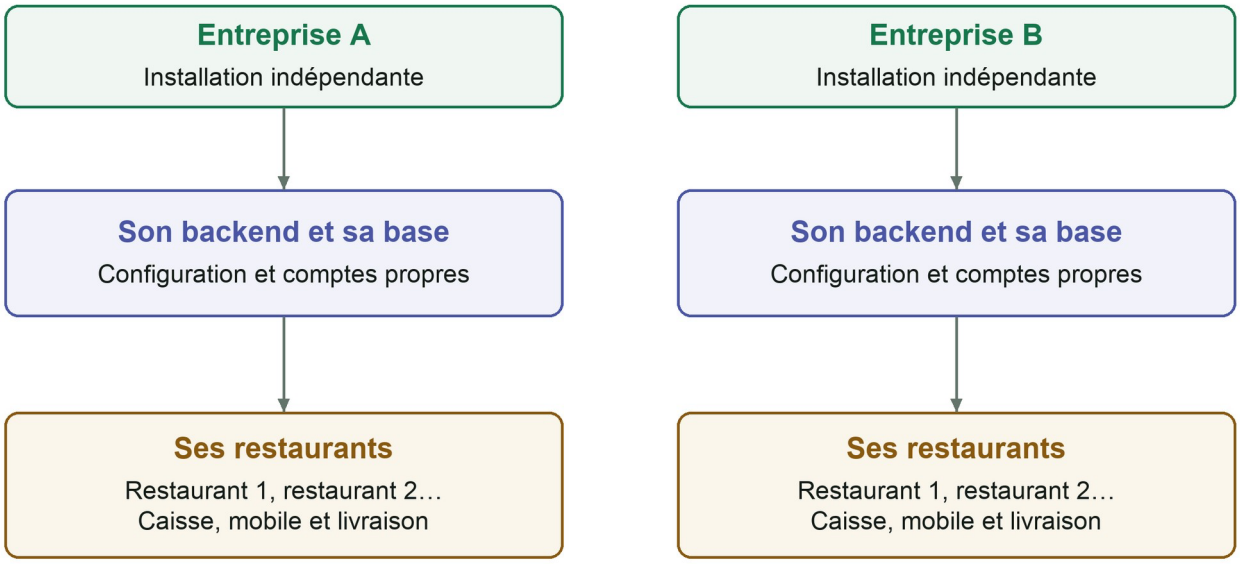


Figure 2 Une installation indépendante peut contenir plusieurs restaurants

NB. Le super administrateur peut disposer d'une vue sur plusieurs restaurants de son installation. Le choix « Tous » ne doit pas être interprété comme un accès à d'autres entreprises indépendantes. L'isolation des bases et des accès d'hébergement reste à organiser au déploiement.

Sources : B02 et ARCHITECTURE_REFERENCE.md.

Les acteurs et leurs responsabilités

Donner le bon accès à la bonne personne

Le rôle et le restaurant se complètent

Le rôle définit les opérations possibles. Le restaurant détermine les données sur lesquelles ces opérations peuvent porter. Un compte actif peut encore être refusé s'il n'a pas accès au restaurant demandé ou à l'action concernée.

Acteur	Outil principal	Responsabilité
Super administrateur	Back-office et administration caisse	Préparer l'installation et gérer les restaurants et les accès autorisés.
Gérant ou administrateur	Back-office et caisse	Organiser le restaurant, son catalogue, son équipe et le suivi de ses opérations selon ses permissions.
Caissier	Caisse Pro	Ouvrir le service, contrôler les commandes, enregistrer les règlements et préparer la clôture.
Serveur ou personnel	POS de la caisse	Saisir et suivre les commandes dans son périmètre. Certaines corrections restent réservées au responsable.
Livreur	Application de livraison	Consulter ses affectations, scanner leur QR puis déclarer la livraison ou le retour.
Client	Application mobile et site	Consulter l'offre, commander, réserver et retrouver ses informations personnelles.

Les libellés visibles et les valeurs techniques peuvent différer. Le backend normalise notamment « caissier » vers **cashier**, plusieurs anciens rôles de personnel vers **staff** et les anciens rôles de livraison vers **livreur**. Le serveur applique ensuite les permissions et l'accès au restaurant.

Préparer un compte utilisable

1. Créer le compte avec son rôle et son état actif.
2. Vérifier son rattachement au restaurant et, pour un livreur, sa fiche de livraison.
3. Tester la connexion dans l'application correspondant à son métier.
4. Vérifier une lecture puis une action autorisée, ainsi que le refus d'un accès hors périmètre.

Bonne pratique. Utiliser des comptes personnels pour attribuer les opérations à la bonne personne. Éviter de prêter un compte administrateur pour contourner un problème de droit.

Sources : B01 B02 B05 C07 L01.

Architecture interne du backend

Du point d'entrée HTTP à la base de données

Des couches avec des responsabilités distinctes

Les routes déclarent les URL disponibles. Les middleware vérifient notamment l'authentification, les permissions ou l'accès à une surface métier. Les contrôleurs valident les paramètres et préparent la réponse. Les services portent les calculs et les actions réutilisables ; les modèles représentent les données et leurs relations.

Cette organisation permet à la caisse, au mobile client et au site web de réutiliser des règles centrales. Par exemple, le prix d'une commande client passe par **OrderPricingService**, les transitions de commande par **OrderStatusTransitionService** et la livraison par **DeliveryService**.

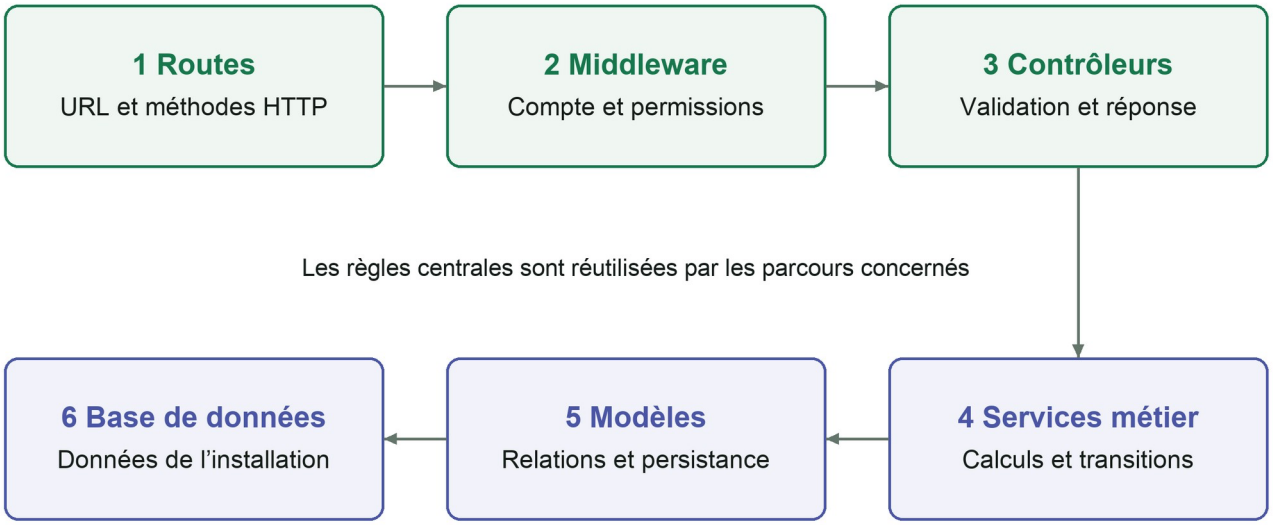


Figure 3 Les couches du backend et leurs responsabilités

Les services métier à connaître

OrderService crée les commandes selon leur origine. **PaymentService** encadre la confirmation, l'échec et le remboursement enregistrés dans le système. **SyncService** reçoit ou restitue les données de synchronisation. **ProductPriceResolver** choisit le prix correspondant au type demandé. Les contrôleurs de réservation et de catalogue appliquent les validations de leurs domaines.

NB. L'existence d'un service commun ne signifie pas que tous les parcours utilisent exactement le même contrat. La création client et l'import des commandes POS conservent des règles de compatibilité distinctes, détaillées aux pages sur les prix et la synchronisation.

Sources : B01 B03 B04 B05 B06 B07 B08.

Les données et leurs relations

Comprendre les objets partagés entre les applications

La commande est un point de liaison central

Un restaurant possède des commandes. Une commande contient des lignes d'articles et peut être associée à un client, à un utilisateur opérateur et à une table. Son origine et son mode de service restent enregistrés pour distinguer une commande POS, web ou mobile et un service sur place, à emporter ou en livraison.

Le modèle **Delivery** représente la fiche de livreur rattachée à un compte et à un restaurant. **OrderDelivery** représente l'affectation d'une commande à cette fiche : statut, clé d'affectation et dates des étapes. Le compte utilisateur, la fiche de livreur et l'affectation sont donc trois objets différents.

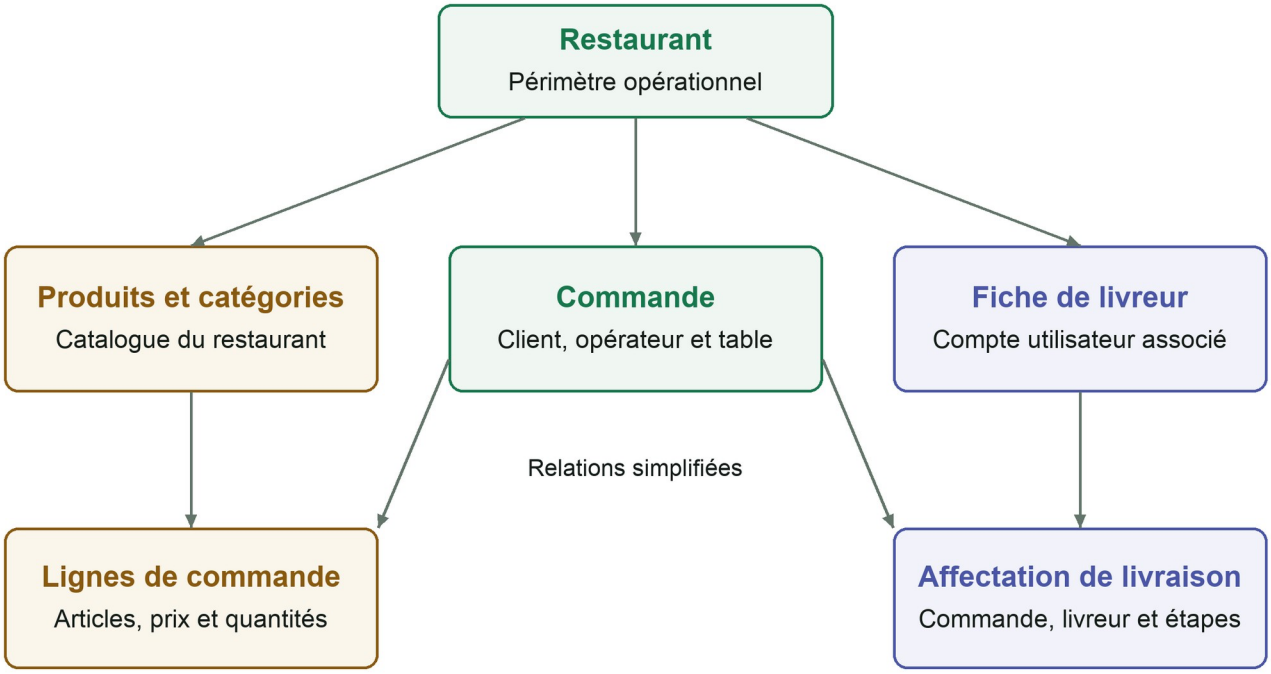


Figure 4 Les principales relations autour du restaurant et de la commande

Une réservation possède son propre cycle de validation et peut recevoir une table. Un crédit correspond à une créance. Une session de caisse regroupe les informations d'ouverture et de clôture. Les usages de promotions et les événements de paiement assurent le suivi des opérations concernées.

À retenir. Le schéma est volontairement simplifié : il explique les liens utiles au fonctionnement. Les migrations et les modèles restent la référence pour les colonnes exactes, les relations facultatives et les contraintes techniques.

Sources : B02 B04 B05 B07.

Authentification et contexte de connexion

Identifier la personne et autoriser sa demande

Deux familles de connexion

Les interfaces web utilisent une session de navigateur. Les API protégées utilisent l'authentification Sanctum et un jeton envoyé dans l'en-tête **Authorization: Bearer**. La possession d'un jeton ne dispense pas des contrôles de rôle, de permission et de restaurant.

La caisse dispose de parcours de connexion par PIN ou badge en plus des mécanismes de compte. L'application livreur utilise le point d'entrée de connexion v1 et refuse un rôle autre que livreur. L'application client conserve son propre parcours de compte et les contrats historiques encore exposés par Laravel.

Une connexion complète en cinq étapes

1. L'application utilise l'adresse du backend de l'entreprise.
2. Elle envoie les informations du compte ou du parcours PIN/badge.
3. Laravel vérifie le compte, son état et le contexte demandé.
4. L'application reçoit la réponse de session ou le jeton utilisable pour les requêtes suivantes.
5. Chaque opération protégée fait encore l'objet des contrôles prévus côté serveur.

Bien interpréter les erreurs

Une erreur **401** demande généralement de vérifier la session ou le jeton. Une erreur **403** indique un refus d'accès : reconnecter indéfiniment le même compte ne corrige pas son rôle ou son rattachement. Une réponse **422** signale souvent des données ou une transition non acceptées. Une erreur réseau ne prouve pas que le compte est invalide.

NB. Les mécanismes de stockage diffèrent entre les applications. Le mobile client utilise notamment SharedPreferences, l'application livreur utilise flutter_secure_storage et la caisse possède sa gestion de session locale. On ne doit pas annoncer une protection identique pour tous ces emplacements sans examen propre à chaque application.

Bonne pratique. En cas d'incident, noter l'application, le rôle, le restaurant et le code de réponse. Ne pas transmettre de mot de passe, de jeton complet ou de secret QR dans une capture destinée au support.

Sources : B01 B02 C07 M01 L01.

Les API et le sens des échanges

Lire les contrats sans confondre les versions

Une API est un contrat entre deux composants

Une requête indique une action, les données nécessaires et l'identité de son auteur. La réponse fournit un résultat ou une erreur. Les applications n'écrivent pas directement dans les tables de Laravel : elles passent par ces contrats et leurs contrôles.

Le backend expose des routes historiques sous **/api/** et des routes versionnées sous **/api/v1/**. Le client mobile utilise encore plusieurs routes historiques, notamment pour ses commandes et réservations. La livraison utilise les routes v1 dédiées. La caisse dispose de services qui normalisent les chemins et prennent en charge certaines variantes de compatibilité.

Échange	Point d'entrée représentatif	Résultat attendu
Connexion API	POST /api/v1/auth/login	Identité et jeton autorisé.
Commande client mobile	POST /api/orders	Commande créée ou réponse correspondant à une demande déjà traitée.
Devis serveur	POST /api/v1/orders/quote	Lignes et détail du calcul autorisé.
Synchronisation générale	POST /api/v1/sync/push et GET /api/v1/sync/pull	Résultats des opérations et données récupérables.
Affectation de livraison	Route v1 restaurant puis delivery/orders/{order}/assign	Affectation de la commande au livreur.
Travail du livreur	Routes v1 delivery/orders puis receive, deliver ou return	Statuts acceptés par le backend.

Les URL de ce tableau sont des repères de lecture. Les paramètres, permissions et formats complets doivent être vérifiés dans les routes et les contrôleurs lors d'une intégration. Une ancienne URL présente dans un service n'est pas automatiquement un endpoint disponible dans toutes les versions du serveur.

Attention. Changer l'adresse du serveur et changer le contrat d'API sont deux opérations différentes. Une application peut atteindre un serveur tout en recevant une erreur de version, de permission ou de format.

Sources : B01 B05 B06 C03 C04 M01 M02 L02.

Architecture de l’application client

Le parcours de commande dans soyabox_mobile

Une interface client connectée au restaurant choisi

L’application est organisée en écrans, contrôleurs GetX, dépôts de données et client API. Les écrans présentent les produits et les actions. Les contrôleurs suivent l’état du panier, du restaurant ou des commandes. Les dépôts appellent les endpoints appropriés ; **ApiClient** prépare les en-têtes et interprète les réponses réseau.

RestaurantContext conserve le restaurant actif. Ce contexte est indispensable : le catalogue, les zones de livraison et la commande doivent correspondre au même établissement. Les données de panier et d’historique présentes localement facilitent l’usage de l’interface, mais ne créent pas à elles seules une commande centrale.

Les principales fonctions repérées

Le projet contient la découverte de restaurants, les catégories et produits, les favoris, la gestion du panier, les adresses, la commande et son historique. Il contient aussi les réservations, les avantages et promotions ainsi qu’une couche de notifications et de liens entrants.

La page de confirmation d’une commande dépend du résultat reçu du backend. Le contrôleur d’ordre enregistre localement la référence retournée après une réponse acceptée ; il remonte une erreur si la création ne reçoit pas de réponse exploitable.

Les informations à vérifier avant de commander

1. Le restaurant actif et sa disponibilité dans l’application.
2. Le contenu et les quantités du panier.
3. L’adresse ou le mode de retrait et la zone de livraison applicable.
4. L’avantage demandé, les frais et le montant affiché avant envoi.
5. La référence de confirmation retournée après l’envoi.

NB. Un panier sauvegardé, un historique en cache et une commande confirmée sont trois états différents. Le manuel ne présente pas le mobile client comme une caisse autonome hors connexion.

Sources : M01 M02 M03 M04 M05.

De la commande mobile au restaurant

Suivre le passage du panier à la commande centrale

Un parcours avec un accusé de réception

Le client compose son panier dans le restaurant choisi, renseigne le service et valide l'envoi. L'application prépare notamment l'identifiant du restaurant, les produits, les quantités et une clé de demande. Laravel contrôle les données et construit la commande avec ses lignes.

Dans le parcours mobile, le service de commande résout les prix et les avantages côté serveur. Il peut retrouver une commande déjà créée avec la même clé pour éviter un doublon. La réponse donne la référence centrale que l'application utilise pour la confirmation et le suivi.

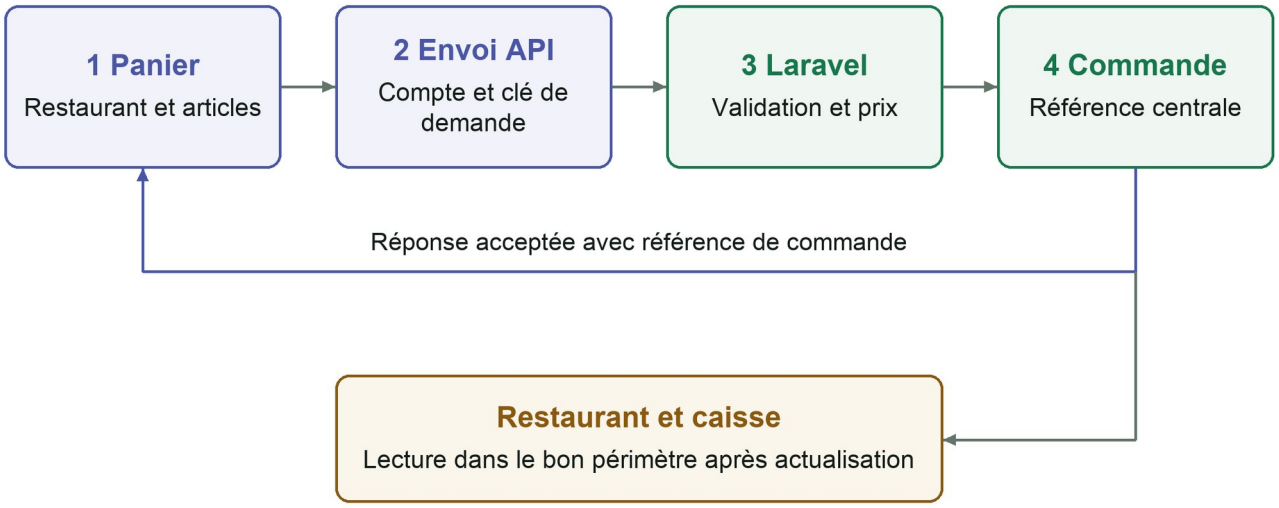


Figure 5 Le parcours d'une commande client et sa confirmation

Le restaurant consulte ensuite la commande dans les interfaces et services qui lui sont autorisés. La caisse possède un service de récupération des commandes distantes et une correspondance entre leurs identifiants centraux et locaux. La visibilité dépend de la connexion, du périmètre et de l'actualisation de l'écran.

En cas de délai. Si l'envoi semble bloqué, vérifier la référence et l'historique avant de créer une autre commande. Pour une intégration, conserver la clé de la même tentative logique pendant ses reprises ; une nouvelle clé peut être interprétée comme une nouvelle demande.

Sources : B03 B06 C03 M02.

Prix frais promotions et avantages

Comprendre les calculs qui accompagnent une commande

Le backend calcule les commandes client

Le service de tarification lit les produits du restaurant, leur prix applicable et les quantités. Il évalue la visibilité du canal, les conditions des promotions et des avantages, puis les règles de livraison. Les montants envoyés par le client ne suffisent pas à imposer le total final de ce parcours.

La formule du calcul client est : **sous-total moins remises sur produits moins remise de commande plus frais de livraison moins remise de livraison plus taxe éventuelle**, avec un total borné à zéro. Les réductions de livraison issues du seuil, d'une promotion et d'un avantage sont rapprochées selon les règles du service ; elles ne s'additionnent pas librement.

Exemple fictif	Montant
Articles après prix et quantités	200 MAD
Remise de commande autorisée	-20 MAD
Frais de livraison	+15 MAD
Réduction de livraison autorisée	-15 MAD
Taxe de cet exemple	0 MAD
Total de l'exemple	180 MAD

Validation puis consommation

L'aperçu d'une promotion vérifie sa pertinence pour le panier. L'utilisation enregistrée dépend de la création effective de la commande. Les codes, tokens QR et récompenses ont leurs conditions : restaurant, période, produit, quantité ou limites d'usage selon leur configuration.

Attention. Un QR promotionnel et un QR d'affectation au livreur ont des buts et des contrats différents. Ils ne sont pas interchangeables.

Cas de la caisse. Les commandes POS importées ou synchronisées conservent un parcours de compatibilité, notamment identifié par **legacy-pos-v1**. Le calcul central des commandes client ne doit donc pas être présenté comme une recalculation identique de tous les encaissements locaux.

Sources : B03 B08 M02 M04.

Le catalogue selon le canal de vente

Produits disponibles et visibilité POS mobile web

Un même produit peut avoir plusieurs destinations

Le catalogue contient des catégories, des produits et des prix complémentaires par type. Les indicateurs de visibilité **POS**, **mobile** et **web** permettent de choisir où un produit peut apparaître. La disponibilité et le périmètre restaurant interviennent aussi dans les requêtes de catalogue.

Un produit créé depuis la caisse peut être destiné seulement au POS selon les valeurs de création et les options envoyées. Le retrouver dans la caisse ne démontre donc pas sa publication sur le site ou dans l'application client.

Situation	Contrôle utile
Visible en caisse seulement	Vérifier la visibilité mobile et web ainsi que le restaurant.
Présent mais non commandable	Contrôler disponibilité, catégorie, prix et règles du canal.
Prix différent entre deux parcours	Comparer le type de prix, le restaurant et la version de catalogue.
Ancienne image après modification	Vérifier le fichier publié, son accès et le cache de l'application.

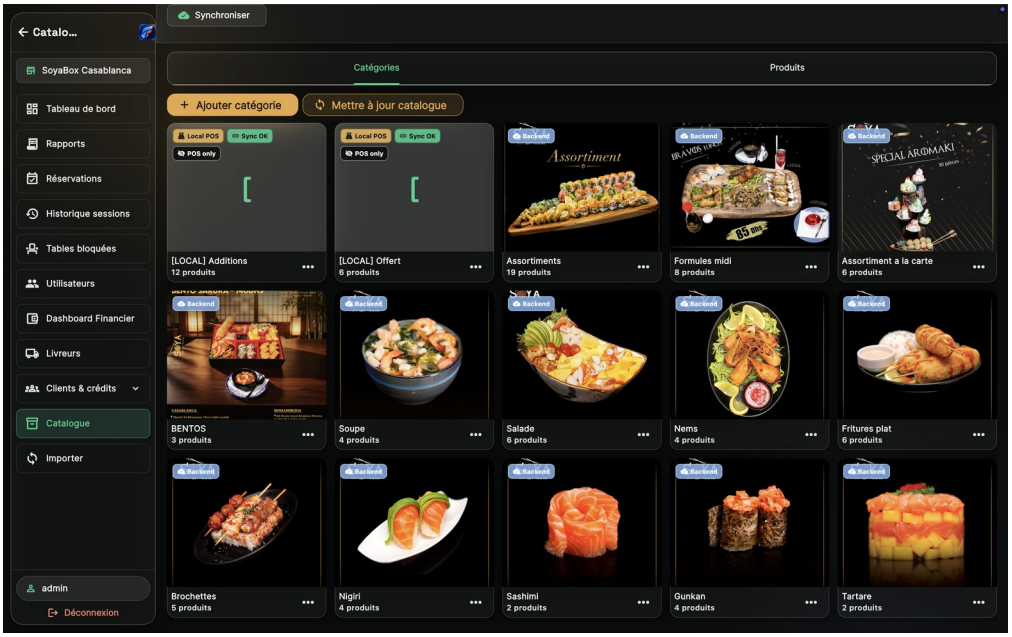


Figure 6 Le catalogue local de la caisse fourni pour le manuel

Bonne pratique. Après une modification, contrôler un produit représentatif dans chaque canal réellement utilisé. La réussite d'un import ou la présence d'une ligne dans le back-office ne remplace pas ce contrôle.

Sources : B08 C03 M03.

Les réservations entre client et restaurant

Une demande distincte de la commande

Recevoir puis décider

Le client transmet un restaurant, une date, un horaire selon le formulaire, le nombre de personnes et ses coordonnées. Le backend crée une réservation en attente. Le personnel autorisé peut la consulter, décider de son statut et lui associer une table du même restaurant.

Le cycle serveur repéré autorise notamment le passage d'en attente à confirmée, annulée ou rejetée. Une réservation confirmée peut être terminée ou annulée. Une demande initiale ne constitue donc pas à elle seule une réservation confirmée.

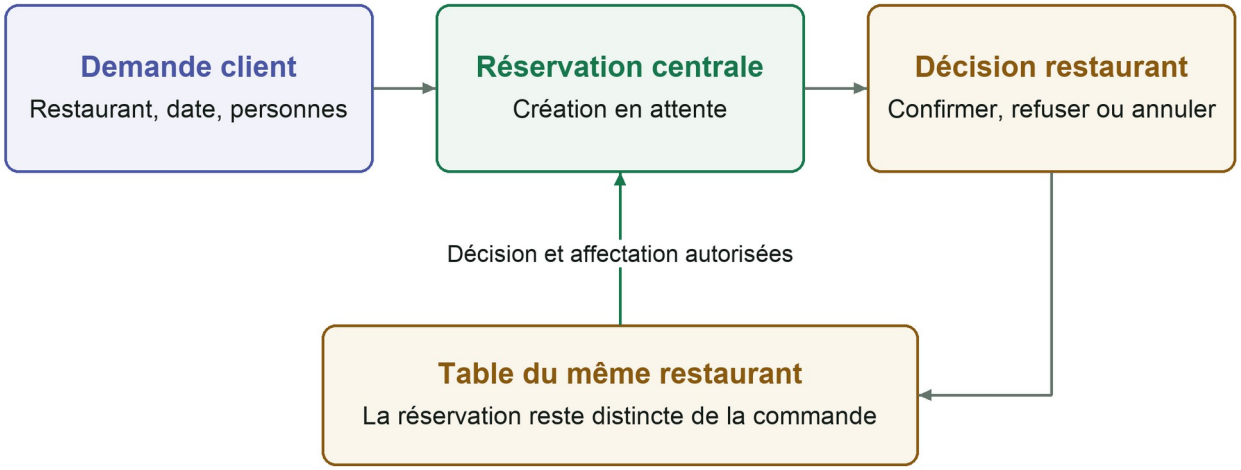


Figure 7 La demande de réservation et la décision du restaurant

Les réservations et les commandes sont conservées séparément. Affecter une table à une réservation ne prouve pas que la commande des convives existe ni qu'une addition a été réglée. La caisse possède ses services de réception et de suivi des réservations connectées.

NB. La disponibilité de la vue dépend du serveur et des droits du compte. En cas de coupure, ne pas interpréter une liste vide comme l'absence de demandes. Vérifier aussi le restaurant, les dates et les filtres avant de ressaisir une réservation.

Sources : B07 C07 M04.

Architecture de la caisse locale

Le terminal Windows ou macOS et ses services

Une application métier avec sa propre persistance

La caisse associe des vues Flutter, des contrôleurs et des services locaux. **DatabaseService** et les modèles Isar conservent les entités nécessaires au terminal. Les paramètres déterminent notamment le restaurant, le serveur et les imprimantes. Les files d'impression et de synchronisation ont des responsabilités séparées.

La saisie, l'encaissement et le suivi de session utilisent le contexte local de la caisse. Les services connectés récupèrent des données et transmettent les opérations vers Laravel lorsque les conditions d'accès sont réunies. Les informations présentes localement peuvent donc être plus récentes que leur copie serveur pendant une coupure.

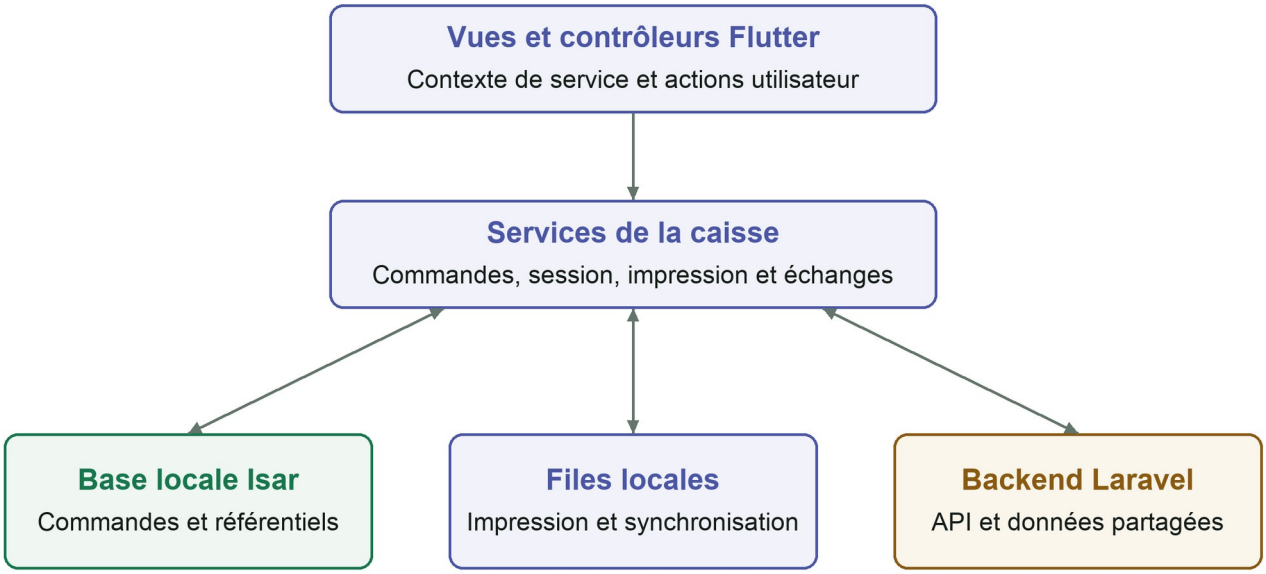


Figure 8 Les couches locales de la caisse et leurs connexions

Préserver le terminal comme un ensemble

La base locale, les fichiers de file, les correspondances d'identifiants, les paramètres et les secrets nécessaires ne remplissent pas le même rôle. Copier uniquement l'exécutable ne transfère pas l'état complet de travail. Une réinstallation ou une migration de poste demande une procédure de sauvegarde et de reprise préparée.

À retenir. Un ticket imprimé prouve une action d'impression. Il ne prouve pas automatiquement que le backend a reçu la commande, que le paiement est encaissé ni que le livreur a pris en charge la livraison.

Sources : C01 C02 C03 C04 C06 C07.

Le travail du restaurant dans la caisse

Service commandes et coordination de l’équipe

La caisse transforme le service en opérations tracées

Le personnel choisit le type de commande, prépare le panier et rattache les informations utiles : table, client, notes, type de plat ou référence externe selon le parcours. La commande enregistrée peut être retrouvée, complétée ou suivie selon les droits de l’utilisateur et son état.

Le serveur suit son travail, le caissier contrôle les règlements et le responsable intervient sur les actions qui demandent une autorisation supplémentaire. Le guide Caisse Pro détaille les gestes de saisie, les appuis longs, les paiements partiels, les remises et les impressions.

Comprendre les trois axes d’une commande

Axe	Exemple	Utilité
Canal d’origine	POS, web, API ou mobile	Identifier l’origine de la vente et lire les rapports par canal.
Mode de service	Sur place, à emporter ou livraison	Déterminer le traitement opérationnel.
État de traitement	En attente, préparation, prête ou terminée	Suivre l’avancement du travail.

Le statut de paiement est encore un autre axe. Une commande prête peut rester non payée ; une commande déjà payée peut nécessiter une préparation ou un acheminement. Les actions autorisées dépendent de ces états et des contrôles du parcours concerné.

Coordonner les écrans

1. Vérifier le restaurant et la session de travail.
2. Identifier la commande par sa référence et son origine.
3. Contrôler son état actuel avant une action.
4. Réaliser l’action avec le compte autorisé.
5. Vérifier le résultat local puis la remontée serveur si cette opération doit être partagée.

Attention. Réimprimer ou rouvrir un écran ne doit pas conduire à recréer une vente existante. En cas de doute, rechercher la commande et ses références avant de ressaisir son panier.

Sources : C02 C05 C06 B04.

Synchronisation entre caisse et backend

Envoyer récupérer et rapprocher les opérations

Plusieurs mécanismes complémentaires

La caisse contient une file persistante d'envoi, un service de récupération des commandes distantes, des services de mise à jour et des échanges dédiés aux rapports ou à la synchronisation quotidienne. Le backend expose aussi un contrat général push/pull. Ces mécanismes doivent être décrits selon leur rôle plutôt que comme une unique copie instantanée de toute la base.

SyncQueueService conserve des opérations sur disque, gère des reprises et isole certains échecs dans une file dédiée. **ApiOrderPullService** maintient l'état de correspondance des commandes récupérées. Le backend contrôle les appareils, les restaurants et les opérations ; le contrat général sait identifier des doublons ou des versions en conflit.

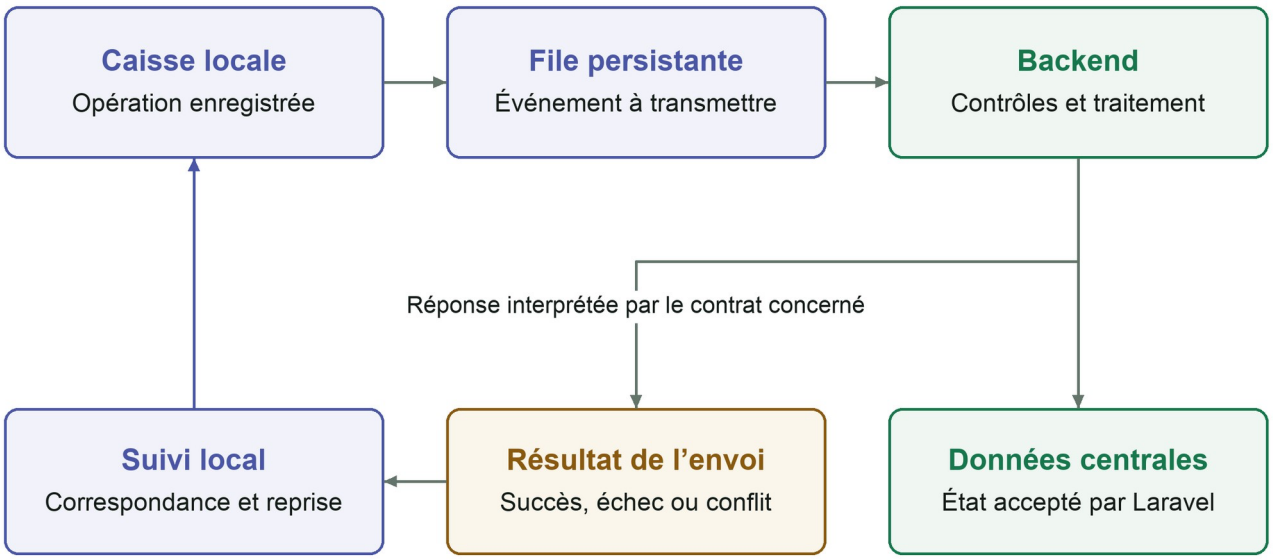


Figure 9 L'envoi local et le retour d'information du backend

NB. Une opération en attente reste locale jusqu'à son traitement accepté. Une opération en échec ou en conflit appelle un diagnostic ; son absence dans la vue centrale ne justifie pas de supprimer la base locale. Les chemins de reprise et la prise en compte des suppressions varient selon les entités et les contrats.

Sources : B06 C03 C06.

Identifiants correspondances et doublons

Garder le lien entre une opération locale et sa représentation centrale

Des numéros différents peuvent désigner la même opération

Un identifiant Isar désigne un objet du terminal. Un identifiant Laravel désigne un enregistrement central. Ils n'ont pas l'obligation d'être égaux. La caisse possède des correspondances pour retrouver l'objet distant à partir du local, et inversement.

Identifiant ou clé	Rôle
id local	Identité de l'objet dans la base du terminal.
id central ou backendOrderId	Référence de l'objet correspondant dans Laravel.
source_local_id	Référence locale transmise dans certains contrats d'échange.
client_order_id ou idempotency_key	Reconnaître une même demande de création et limiter les doublons selon le contrat.
operation_id	Reconnaître une opération du contrat général de synchronisation.
assignment_key	Identifier l'affectation associée au QR de livraison.
localEventId	Identifier un événement local de l'application livreur lors de ses reprises.

Par exemple, une commande locale 42 peut correspondre à la commande centrale 845. L'opération de livraison doit atteindre la bonne référence centrale tout en gardant le lien avec la commande locale. Le compte du livreur et sa fiche **Delivery** demandent également un rapprochement explicite.

L'idempotence rend une reprise reconnaissable

Si une demande a été traitée mais que sa réponse a été perdue, la même clé permet au contrat concerné de retrouver le résultat ou de refuser un contenu incohérent. Ce mécanisme ne remplace pas les permissions, les contrôles de restaurant ni les règles de statut.

Attention. Remplacer une identité locale par un numéro serveur peut casser les liens entre objets. Une réparation doit examiner la correspondance concernée et préserver les références existantes. La clé d'une opération ne doit pas être changée à chaque reprise de cette même opération.

Sources : B03 B05 B06 C03 C04 C05 M02 L03.

Paiement crédit et clôture de caisse

Relier l’argent aux commandes sans confondre leurs états

Enregistrer un règlement est une opération distincte

La caisse propose les moyens de règlement prévus par ses parcours et conserve le détail nécessaire au service. Le backend possède ses statuts de paiement et un service de confirmation, d’échec ou de remboursement, avec événements associés pour ces actions centrales.

Un crédit représente un montant à recouvrer. Une remise ou un offert réduit la valeur demandée selon les règles du parcours. Une commande livrée n’est pas automatiquement payée. De même, le libellé TPE désigne le mode enregistré ; il ne démontre pas une transaction bancaire automatique entre le logiciel et un terminal bancaire.

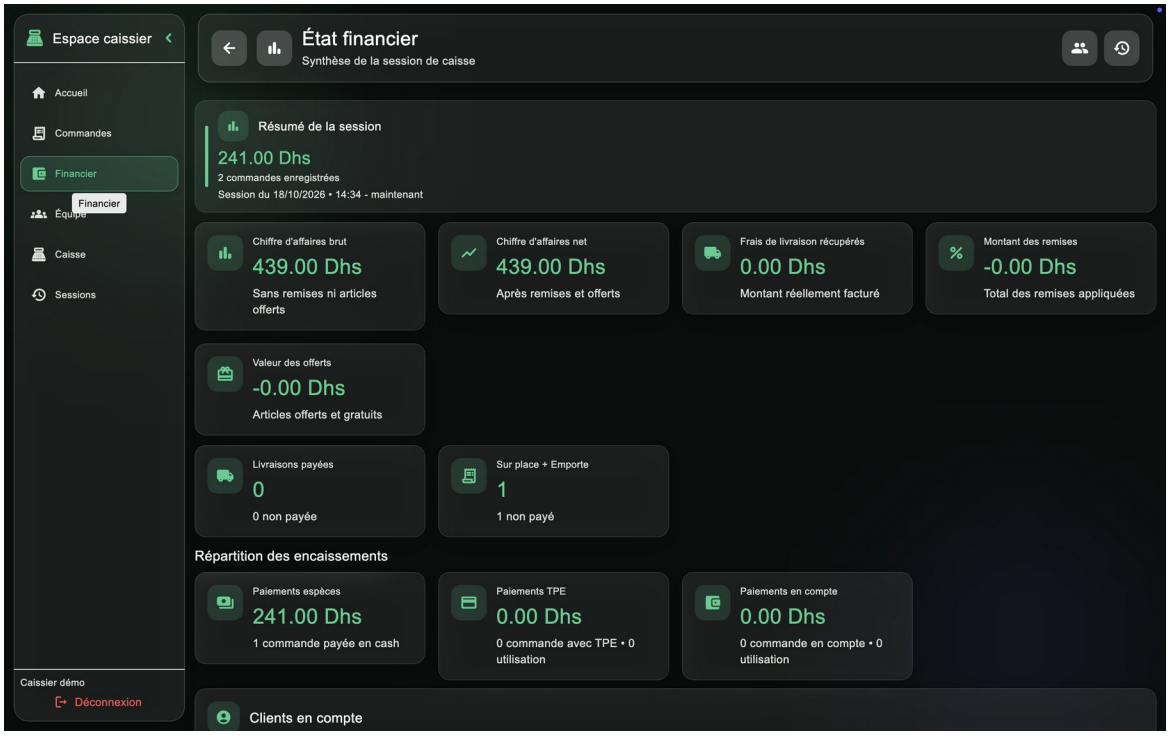


Figure 10 L’état financier de la caisse et ses informations de rapprochement

La clôture doit rapprocher les commandes, les règlements, les crédits et les opérations encore ouvertes. Les rapports locaux et centraux peuvent être comparés une fois leurs périmètres et leur fraîcheur alignés : même restaurant, même session ou période, même traitement des annulations et mêmes catégories de montants.

NB. La clôture d’une session, la fin d’une livraison et le versement de la recette d’un livreur sont des événements différents. La procédure d’exploitation doit préciser qui les contrôle et dans quel outil l’information de paiement devient définitive.

Sources : B04 B09 C06.

Architecture de l’application livreur

Une application dédiée aux affectations personnelles

Des services simples autour d’une session livreur

L’application de livraison est structurée en fonctionnalités : authentification, accueil, liste des commandes, détail et scan QR. Un service HTTP dialogue avec Laravel. Le stockage sécurisé conserve la session et les actions de reprise. Les modèles interprètent les commandes et rapprochent les informations serveur des événements locaux.

Le service **ReceiveSyncQueue** organise la réception et les déclarations de livraison ou de retour. Il conserve l’auteur, la commande et l’affectation concernés. L’interface peut montrer une action locale en attente de synchronisation sans modifier de façon arbitraire le statut officiel reçu de Laravel.

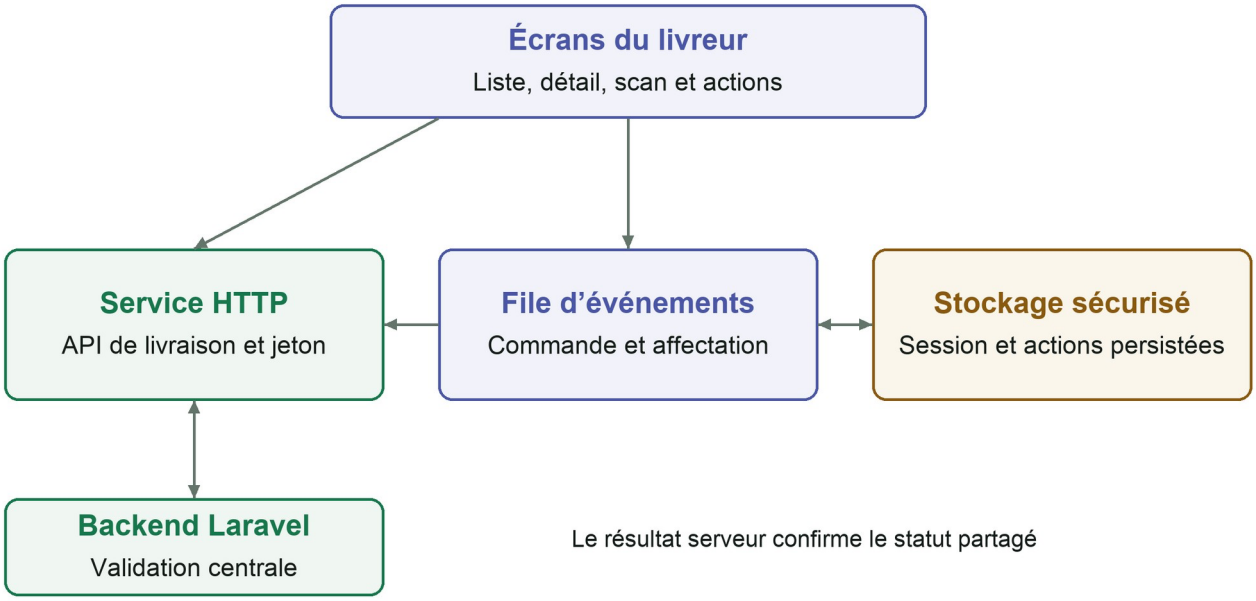


Figure 11 Les composants de l’application livreur

La liste serveur est construite pour le livreur actif et les affectations du jour selon l’horloge du backend. Elle utilise les affectations avec QR et le lien entre la fiche de livreur et son compte. Un livreur ne doit donc pas retrouver toutes les commandes de l’entreprise.

À retenir. La commande, l’affectation et le compte actif doivent être cohérents. Une ancienne affectation ou un autre livreur ne doit pas récupérer la confirmation d’une action locale qui ne lui appartient pas.

Sources : B05 L01 L02 L03 L04.

Affecter une commande à un livreur

Le restaurant prépare le passage vers l'application de livraison

Une affectation relie trois objets

Pour affecter une commande, le restaurant doit disposer de la commande correspondante sur le serveur et d'une fiche de livreur utilisable. Cette fiche est liée au compte qui se connecte dans l'application de livraison. Le nom visible du livreur ne suffit pas à établir ce lien : les identifiants et le restaurant doivent correspondre.

La caisse prépare une **assignment_key** et le secret du QR avec son service dédié. Le backend contrôle la demande et conserve l'affectation. Le code central conserve une empreinte de vérification ainsi qu'une représentation chiffrée du QR ; les champs secrets ne sont pas exposés par la sérialisation ordinaire du modèle.

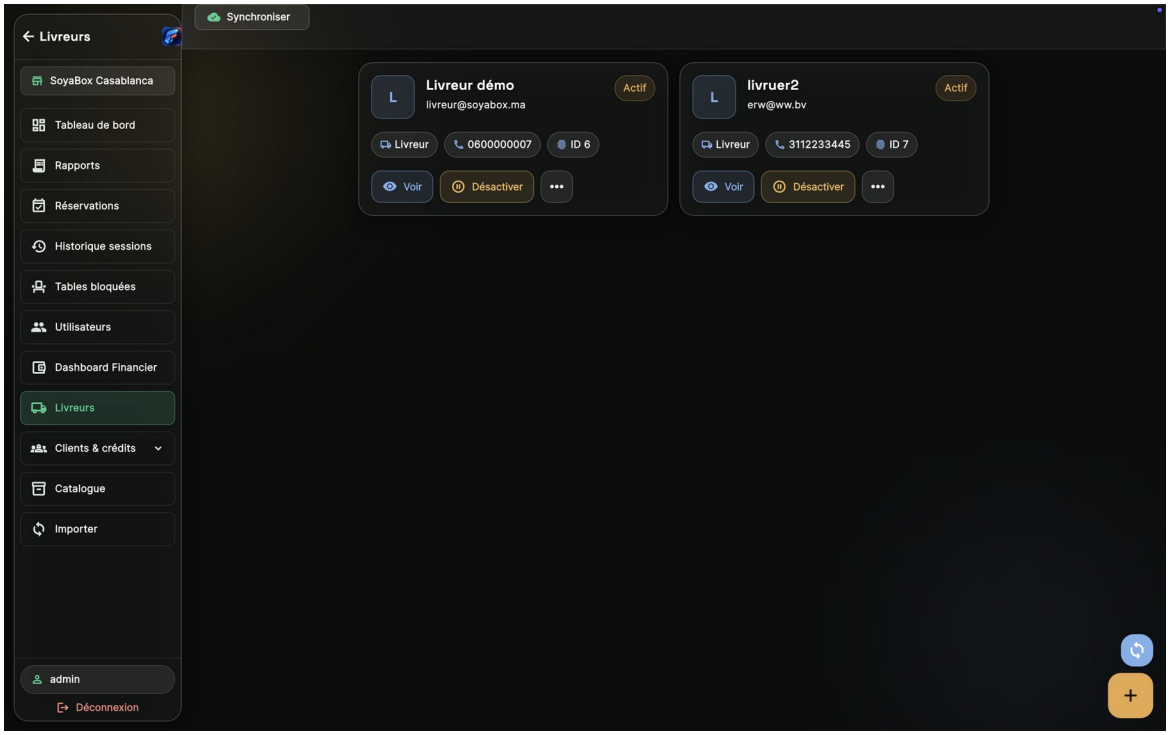


Figure 12 La liste des livreurs dans la caisse fournie pour le manuel

Les contrôles utiles au comptoir

1. Retrouver la commande et vérifier qu'elle est bien en livraison.
2. Choisir la fiche du livreur correspondant à la personne qui partira.
3. Vérifier que la commande et le livreur ont leurs correspondances centrales.
4. Contrôler le résultat de l'affectation et l'état de synchronisation du QR.
5. Remettre au livreur le ticket ou le support correspondant à cette affectation.

Attention. Un QR imprimé alors que l'affectation n'a pas encore été acceptée par le backend peut rester inutilisable côté livreur. Après un changement de livreur, reprendre le support de l'affectation active et vérifier l'état serveur avant le départ.

Sources : B05 C04 C05 L02.

La réception de commande par QR

Confirmer la prise en charge au restaurant

Le scan déclenche une demande contrôlée

Le QR de livraison contient une clé d'affectation et un secret selon le format prévu par le projet. L'application transmet cette preuve au point d'entrée de réception. Laravel vérifie notamment la correspondance de la clé, le secret, le livreur autorisé et l'état courant de la commande.

Si la demande est acceptée, le statut de l'affectation devient **picked_up** et celui de la commande **out_for_delivery**. La réception est donc un changement de responsabilité tracé, distinct de la simple lecture d'une image QR par la caméra.

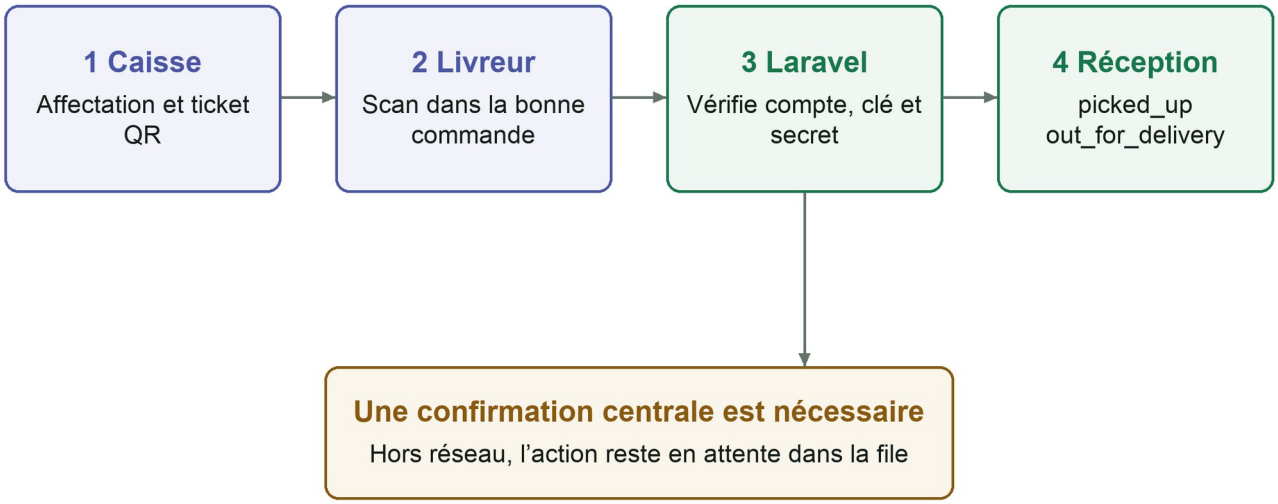


Figure 13 Du QR d'affectation à la prise en charge confirmée

L'application peut d'abord conserver la demande localement et afficher « Reçue — synchronisation en attente ». Elle ne présente une confirmation centrale qu'après une réponse serveur acceptée. Le même événement local sert de clé lors des reprises pour éviter de transformer chaque tentative réseau en nouvelle action logique.

NB. Un QR valide d'une autre commande ou d'une ancienne affectation n'autorise pas la réception de la commande actuellement ouverte. En cas de refus, contrôler le compte, la commande et l'affectation avant de rescanner plusieurs supports.

Sources : B05 C04 L02 L03 L04.

Livraison retour et statuts de suivi

Terminer le parcours opérationnel du livreur

Le backend valide les transitions

L’écran de détail permet les déclarations « Livrée au client » et « Retournée au restaurant » selon les conditions de la vue. Le serveur vérifie à nouveau le livreur, son affectation et le statut courant. Il ne se contente pas du libellé affiché sur le téléphone.

Étape	Affectation de livraison	Commande centrale
Affectation acceptée	assigned	assigned
Réception QR acceptée	picked_up	out_for_delivery
Livraison acceptée	delivered	delivered
Retour accepté	returned	refused dans ce parcours

Dans le service actuel, la livraison réussie exige une prise en charge cohérente. Le retour peut être accepté depuis une affectation ou une prise en charge autorisées. La transition générale de commande peut normaliser des libellés historiques ; cette normalisation ne doit pas effacer la raison métier d’un retour.

Traiter un retour proprement

1. Identifier la commande concernée et vérifier le client et l’affectation.
2. Déclarer le retour dans l’application et confirmer le dialogue prévu.
3. Attendre la synchronisation ou conserver la mention d’attente visible.
4. Informer le restaurant selon son organisation opérationnelle.
5. Rapprocher séparément le montant à encaisser, les frais et tout règlement déjà enregistré.

Attention. « Retournée », « annulée », « remboursée » et « non payée » ne désignent pas la même opération. Le retour met à jour le parcours de livraison et la commande ; un remboursement nécessite le traitement de paiement qui lui correspond.

Bonne pratique. Après une réponse de conflit, recharger l’état serveur et comparer l’affectation avant de forcer une nouvelle action. Le serveur peut déjà connaître une étape différente de celle affichée localement.

Sources : B04 B05 L02 L03 L04.

Les actions du livreur hors connexion

Garder l’intention locale et reprendre l’envoi

Une file persistante propre à l’application

La file conserve un événement avec son **localEventId**, l’action, la commande, le livreur, l’affectation, la date et l’état de synchronisation. Le QR nécessaire à une réception est conservé via le stockage sécurisé prévu par l’application, puis retiré après succès ou après une erreur définitive selon le traitement du service.

Les actions de réception, de livraison et de retour empruntent cette file. Le retour du réseau peut déclencher les reprises. Le service prévoit aussi des délais entre tentatives et tient compte des réponses qui demandent d’attendre.

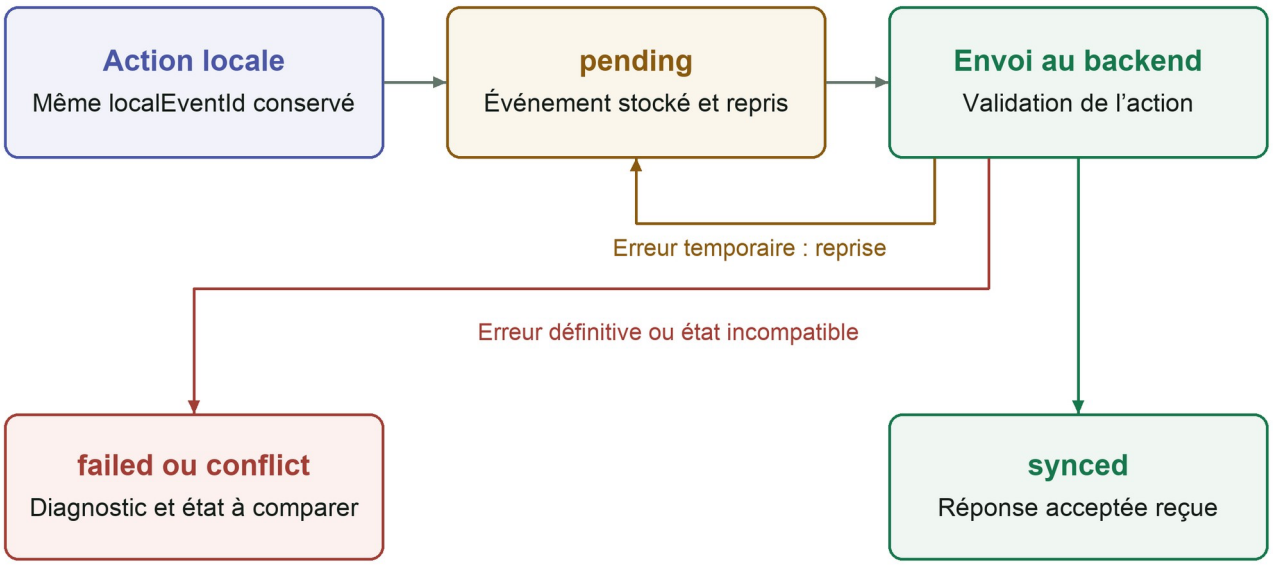


Figure 14 Une action locale conserve sa clé pendant les reprises

État de la file	Signification opérationnelle
pending	L’événement attend encore une réponse acceptée ; il peut être repris.
synced	Le backend a accepté l’action et le résultat central a été reçu.
failed	Le traitement a échoué sans reprise automatique de cette erreur.
conflict	La demande entre en conflit avec l’état ou l’affectation connus du serveur.

NB. Une file d’actions ne rend pas toutes les données disponibles hors connexion. Découvrir une nouvelle affectation, recharger tous les détails ou vérifier une nouvelle session peut encore nécessiter le réseau. Les éléments déjà affichés et les événements locaux doivent rester distingués de la confirmation centrale.

Sources : L02 L03 L04 B05.

Montants du livreur et rapprochement de recette

Lire les totaux sans créer de paiement implicite

Une somme de commandes reste un indicateur de commandes

L’accueil du livreur regroupe les commandes dans ses catégories de suivi et calcule des totaux à partir de leurs montants. Certains éléments peuvent aussi rester visibles grâce au rapprochement avec les actions locales. Ces totaux ne remplacent pas le journal des règlements de caisse.

Le détail transmis au livreur comprend notamment le total, les frais de livraison, le moyen et l’état de paiement, les coordonnées utiles et les articles. Il faut lire ces informations ensemble pour déterminer ce qui reste à remettre ou à encaisser selon l’organisation du restaurant.

Une procédure de fin de tournée

1. Fixer le livreur, le restaurant et la période ou journée à rapprocher.
2. Retrouver les affectations et leurs commandes centrales.
3. Séparer livraisons acceptées, retours et actions toujours en attente.
4. Vérifier le paiement déjà enregistré pour chaque commande concernée.
5. Comparer la recette réellement remise avec les montants qui devaient encore être encaissés.
6. Traiter un écart dans le parcours autorisé et en conserver l’explication.

Exemple de lecture

Une liste de commandes totalise 600 MAD. Elle peut inclure une commande déjà réglée, une commande à payer à réception et une action de retour en attente. Afficher 600 MAD ne signifie donc pas que le livreur doit remettre 600 MAD en espèces. Le montant à remettre dépend des règlements et des états effectifs de ces commandes.

Attention. Ne pas rendre une commande payée simplement parce qu’elle apparaît « livrée » sur un téléphone. Le paiement doit être confirmé dans le parcours habilité et rapproché du règlement réel.

NB. Les affectations du jour utilisent l’horloge du backend. Une comparaison autour de minuit doit tenir compte du fuseau horaire, de la date d’affectation et de la session de caisse.

Sources : L02 L04 B04 B05 B09 C06.

Le site web et les outils de présentation

Une autre entrée client vers le même système central

Le site de commande fait partie du backend

Les routes web de **backend_platform** exposent l'accueil, les menus, les pages restaurant, le panier, la commande et les réservations. Le panier web est géré dans le contexte du restaurant et de la session du navigateur. Le passage de commande utilise le service central prévu pour ce canal.

Le back-office dispose de fonctions de contenu pour l'accueil et de réglages de présentation. Les données catalogue et les indicateurs de visibilité web déterminent les produits affichables. Un changement graphique n'équivaut pas à une modification des prix, des permissions ou de la structure de commande.

La place de theme_platform

Le dossier **theme_platform** contient un autre projet Laravel avec ses propres routes, son authentification et son éditeur de sections d'accueil. Sa présence dans le même dossier de travail ne prouve pas que sa base ou sa publication sont reliées automatiquement au backend opérationnel.

Lors d'une installation, il faut identifier le site réellement publié et son mode d'administration avant de modifier un contenu. Le site exposé par backend_platform peut déjà assurer la partie publique sans passer par ce projet distinct.

Le parcours promotionnel web

Le projet prévoit une URL de scan promotionnel et une génération locale de QR SVG. Le scan doit conserver le contexte de promotion jusqu'au panier puis au calcul et à la création de commande. L'ouverture réussie de la page de scan ne suffit pas à prouver l'application de la réduction.

NB. Les interfaces web POS et livraison existent aussi dans les routes du backend. Elles sont des surfaces distinctes des applications Flutter : leurs écrans et leurs fonctions ne doivent pas être supposés identiques sans comparaison.

Sources : B01 B03 B08 B10 T01.

Pilotage et rapports de gestion

Rassembler les opérations avec un périmètre explicite

Des rapports centraux organisés par domaine

Le back-office possède une page de rapports avec les axes synthèse, ventes, caisse, canaux, livraison, clients, promotions et synchronisation. Les filtres portent notamment sur le restaurant, la période, le canal, le mode de service et les statuts.

Le chiffre d'affaires encaissé de cette page est calculé à partir des commandes payées retenues, avec exclusion des états d'annulation ou de refus prévus par le contrôleur. D'autres montants ont leur propre définition : ventes totales, remises, frais de livraison, crédits ou usages de promotions.

Indicateur	Ce qu'il faut lire avec le montant
Ventes ou commandes	Canal, état, période et restaurant.
Encaissement	Statut et moyen de paiement, annulations exclues selon la vue.
Articles vendus	Quantité multipliée par prix ; traitement des offerts et filtres.
Crédits	Création, solde ouvert, règlement et dates correspondantes.
Livraison	Affectation, état, frais et réduction de livraison.
Caisse	Session, attendu, compté et différence, selon les données reçues.
Synchronisation	Appareil, opération, statut, erreur et dernier échange.

Comparer un rapport local et un rapport central

Commencer par aligner le restaurant et la période. Identifier ensuite si l'écran se base sur la création de commande, l'affectation, le paiement ou la session. Vérifier la fraîcheur de synchronisation et la présence d'opérations locales en attente.

Attention. La somme des totaux de commande ne prouve pas le contenu physique du tiroir-caisse. Les espèces, les paiements TPE enregistrés, les crédits, les offerts et les règlements de plateformes externes doivent être rapprochés selon leur nature.

Bonne pratique. Conserver le nom du rapport et les filtres avec tout export ou capture transmis pour expliquer un écart. Deux captures sans périmètre explicite peuvent montrer des montants différents tout en décrivant des ensembles différents.

Sources : B09 C06 L04.

Actualisation notifications et fraîcheur des données

Comprendre pourquoi deux écrans ne changent pas toujours ensemble

Le réseau et le rafraîchissement ont des rôles distincts

Une opération peut être acceptée par Laravel alors qu'une liste déjà ouverte montre encore une ancienne réponse. La mise à jour d'un écran dépend du chargement de ses données, de ses filtres et de ses mécanismes de rafraîchissement. Les files de la caisse et du livreur ajoutent aussi un temps de reprise en cas de coupure.

Le mobile client contient une intégration Firebase Messaging et un service de notifications, avec enregistrement du token du terminal dans le parcours prévu. Le backend enregistre également des notifications d'administration lors de certaines créations ou modifications de commande.

La présence de ces composants ne démontre pas à elle seule un envoi push opérationnel sur chaque installation ni une liaison WebSocket permanente. Le projet doit être configuré et testé sur les appareils concernés pour garantir le comportement attendu.

Si une modification tarde à apparaître

1. Contrôler que l'action a reçu une confirmation exploitable.
2. Examiner la file locale si l'interface indique une attente.
3. Vérifier que les deux écrans consultent le même restaurant et la même période.
4. Actualiser la liste concernée et ouvrir le détail par sa référence.
5. Comparer l'état central avec l'événement local avant de rejouer l'action.

Les notifications facilitent l'attention

Une alerte sonore, une notification système ou une pastille peut attirer l'attention du personnel. Son absence ne prouve pas que la commande n'existe pas. À l'inverse, une ancienne notification ne prouve pas que la commande est encore dans le même état.

NB. Les permissions de notification du téléphone et l'accès réseau sont deux conditions différentes. Une application peut accéder à ses commandes alors que le système refuse l'affichage de notifications.

Sources : B02 C03 M05 L03 L04.

Sécurité et protection des informations

Contrôles présents et pratiques d’exploitation à organiser

Le serveur reste responsable des actions partagées

Les endpoints protégés contrôlent l’authentification et les autorisations prévues. Les accès restaurant et les règles de transition limitent les opérations possibles. Les services de livraison et de paiement utilisent des transactions pour encadrer leurs changements ; certains traitements verrouillent l’enregistrement concerné.

Pour les QR de livraison, le backend vérifie une empreinte du secret et masque les champs sensibles dans le modèle. L’application livreur conserve les événements via son stockage sécurisé. La caisse possède un stockage dédié au secret QR. Ces mesures n’autorisent pas la diffusion du QR ni des jetons d’accès.

Mesures à prévoir lors d’une installation

1. Utiliser une adresse HTTPS pour les échanges exposés hors d’un réseau de recette.
2. Attribuer les permissions selon les responsabilités et limiter les comptes partagés.
3. Restreindre l’accès au serveur, aux sauvegardes et aux données du terminal.
4. Protéger les clés de chiffrement et les identifiants techniques de l’installation.
5. Définir une procédure de révocation en cas de départ d’un employé ou de perte d’un appareil.
6. Vérifier les droits caméra, notifications et stockage nécessaires à chaque application.

Ne pas confondre les stockages

SharedPreferences n’a pas le même rôle ni les mêmes garanties que flutter_secure_storage. Un fichier de queue JSON, une base Isar et un export de rapport peuvent contenir des informations sensibles. Leur protection doit être évaluée selon le système et les pratiques d’exploitation effectivement retenus.

Attention. Une sauvegarde serveur sans les clés nécessaires peut ne pas suffire à relire les valeurs chiffrées. Une restauration doit être testée dans un environnement isolé avec le responsable de l’installation.

Ce manuel décrit les mécanismes observés et les mesures à organiser ; il ne constitue pas une certification de sécurité, une homologation de caisse ou une garantie de conformité réglementaire.

Sources : B01 B02 B04 B05 B06 C04 C07 M01 L01 L03.

Adresses serveur et topologie réseau

Faire pointer toutes les applications vers la bonne installation

Une adresse commune et des réglages propres à chaque application

Le mobile client et l'application livreur prévoient **API_BASE_URL** à la compilation. La caisse possède un réglage d'adresse de backend et des services qui normalisent les chemins API. Chaque application doit atteindre le backend de la bonne entreprise, avec le restaurant et le compte appropriés.

127.0.0.1 désigne l'appareil lui-même. Sur un téléphone physique, cette adresse n'atteint pas le Mac ou le PC qui héberge Laravel. L'application livreur prévoit aussi l'adresse spéciale **10.0.2.2** pour le contexte émulateur Android ; cela ne doit pas être réutilisé comme adresse universelle de production.

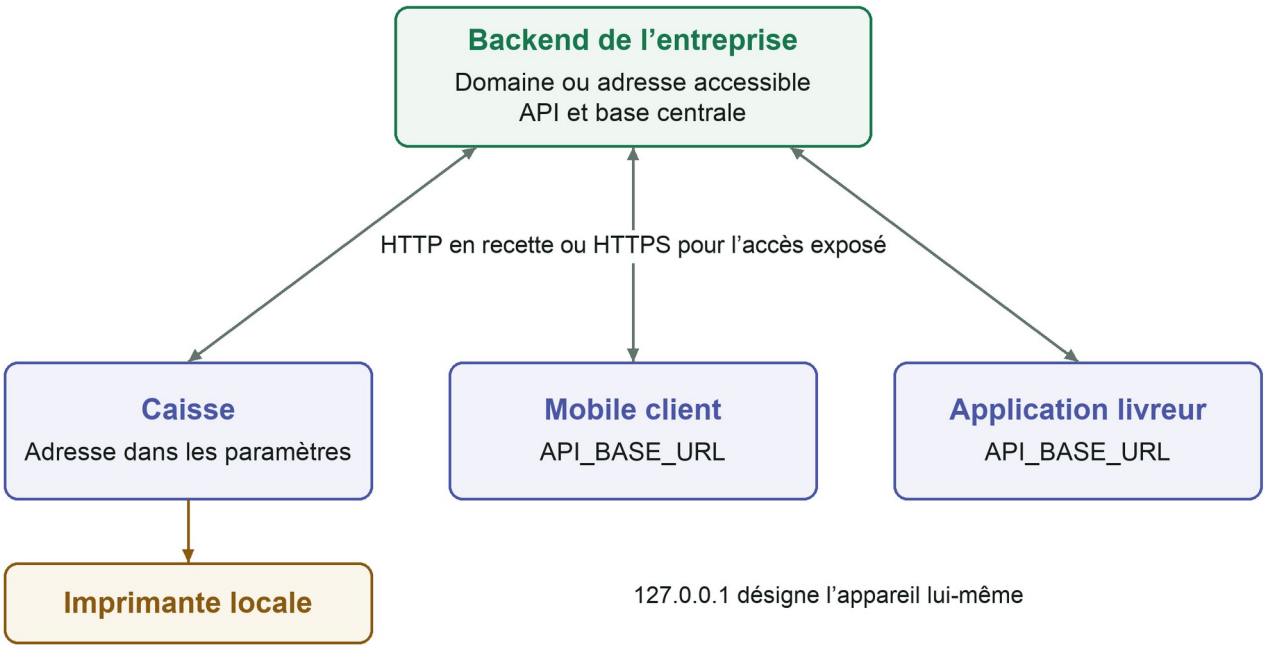


Figure 15 Les connexions locales et distantes d'une installation

Les points à contrôler ensemble

Le domaine ou l'adresse IP, le protocole et le port doivent être accessibles. Le serveur web doit desservir la bonne application et les bonnes routes. Les images doivent être accessibles aux applications qui les affichent. Les règles du réseau et les permissions du système peuvent empêcher un échange même si le logiciel est correctement configuré.

NB. La configuration par défaut présente dans le code de recette n'est pas une preuve d'adresse de production active. Ne pas diffuser un paquet client sans vérifier sa destination réelle.

Sources : M01 L01 C07 B01.

Mettre une entreprise en service

Une séquence de préparation vérifiable

Commencer par le référentiel central

1. Identifier l’entreprise, l’installation Laravel et sa base dédiée.
2. Créer les restaurants et vérifier leurs informations, leur état actif et leurs accès.
3. Préparer les comptes, leurs rôles, leurs appartenances et les fiches de livreurs reliées aux bons comptes.
4. Préparer les catégories, produits, prix, disponibilités et visibilités par canal.
5. Définir les règles de livraison et les promotions réellement proposées.

Brancher les applications

6. Configurer le backend dans la caisse et sélectionner son restaurant.
7. Importer ou rapprocher les données nécessaires, puis contrôler un échantillon de produits, tables et personnes.
8. Configurer les imprimantes, logos et options de tickets nécessaires au site.
9. Préparer le mobile client et l’application livreur avec la bonne adresse de serveur.
10. Vérifier les connexions et les permissions de chaque profil.

Faire une recette avant le premier service

11. Réaliser une commande mobile et une commande POS d’essai identifiées.
12. Vérifier leurs références centrales et locales, puis leurs règles de paiement.
13. Affecter une commande de livraison, scanner le QR avec le bon compte et vérifier les étapes côté serveur.
14. Tester une réservation, les mises à jour de catalogue et le retour d’une coupure sur un jeu d’essai.
15. Comparer les rapports sur le même périmètre et préparer la sauvegarde.

Conserver la version des applications, l’adresse de l’installation, les responsables et le résultat des contrôles dans une fiche de mise en service. Les données d’essai doivent rester reconnaissables pour ne pas se mélanger aux opérations réelles.

NB. Installer un exécutable ou réussir une connexion ne remplace pas la préparation du catalogue, du restaurant, des appareils et des correspondances de personnes. Les actions de suppression ou de réinitialisation ne sont pas des étapes normales de mise en service.

Sources : B01 B02 B08 C01 C03 C04 C07 M01 L01.

Déploiement sauvegarde et reprise

Organiser l’exploitation de l’installation complète

Préserver les données et leur contexte

Le déploiement du backend comprend le code, les dépendances, les paramètres d’environnement, la base relationnelle et les fichiers publiés nécessaires. Les applications distribuées doivent être compatibles avec ses contrats. L’utilisation d’une version récente de l’une d’elles ne garantit pas la compatibilité avec un ancien backend.

La sauvegarde doit être définie pour chaque couche. Côté serveur, prévoir les données et fichiers utiles ainsi que les clés nécessaires à la lecture des valeurs chiffrées. Côté caisse, préserver la base locale, les paramètres et les fichiers de synchronisation ou de correspondance en suivant une procédure cohérente avec l’état du terminal.

Élément	But de la protection	Vérification attendue
Base centrale et fichiers publiés	Retrouver les données et les médias métier	Restauration sur environnement isolé.
Configuration et clés serveur	Reconstituer l’accès et le chiffrement nécessaires	Lecture des données utiles après reprise.
Données locales de caisse	Préserver les opérations et liens du terminal	Ouverture et rapprochement cohérents.
Files locales et correspondances	Reprendre les échanges sans recréer les ventes	Même identité d’opération lors du renvoi.
Applications distribuées	Retrouver une version compatible	Connexion et parcours métier vérifiés.

Une reprise doit être organisée

Préciser qui sauvegarde, à quelle fréquence, où les copies sont protégées et qui peut restaurer. Tester la procédure à part avant d’en dépendre pendant un incident. Une migration de poste peut exiger un traitement particulier des secrets stockés par le système ; une simple copie de fichiers ne le démontre pas.

Attention. Ne pas restaurer arbitrairement une ancienne base de production au milieu d’échanges actifs. Examiner les opérations déjà acceptées et les files encore présentes pour éviter une perte ou une recréation de transactions.

Sources : B05 B06 C01 C03 C04 C06 L03.

Maintenance et évolution des contrats

Faire évoluer une composante sans perdre la cohérence d’ensemble

Commencer par le parcours touché

Une modification de catalogue peut concerner la caisse, le mobile et le site. Un changement de statut touche les contrôleurs, les services et les libellés d’écran. Une évolution du QR doit respecter la caisse émettrice, le backend validateur et l’application livreur qui le lit.

L’existence de plusieurs projets impose donc une vérification transversale. Les routes historiques et les routes v1 peuvent coexister pour préserver les clients installés. Un renommage de colonne ou de paramètre peut casser un contrat même si les pages web continuent de fonctionner.

Une méthode de changement

1. Nommer le problème concret et le parcours concerné.
2. Repérer les écrans, contrats, services, modèles et données persistées utilisés par ce parcours.
3. Conserver une copie et un moyen de retour adaptés au changement.
4. Préserver les identités locales, les clés de demande et les données déjà enregistrées.
5. Vérifier les autorisations et les cas d’erreur avec les applications concernées.
6. Tester la reprise réseau et les anciennes données si le changement touche une file ou un modèle local.
7. Documenter les versions compatibles et le résultat de la recette.

Les zones à coordonner en priorité

Les prix et promotions, la synchronisation de commandes, les rattachements de livreurs, la réception QR, les paiements et les rapports traversent plusieurs composants. Les modifier isolément peut déplacer un problème vers une autre application.

NB. Les tests de contrat présents dans les projets peuvent aider à vérifier une évolution, mais leur présence ne prouve pas qu’ils ont été exécutés ni que les téléphones, imprimantes et serveurs de production ont été validés. La recette doit couvrir le contexte réel d’utilisation.

Bonne pratique. Mettre à jour le manuel et ses repères de code avec les changements de comportement validés. Conserver la différence entre une fonction disponible, une procédure recommandée et une évolution envisagée.

Sources : B01 B03 B04 B05 B06 C03 C04 M02 L03.

Dépannage des relations entre applications

Rechercher le premier point de divergence

Un diagnostic par référence et par périmètre

Noter la référence de commande, le restaurant, les versions des applications, le compte utilisé, l'heure et l'étape observée. Comparer ensuite le dernier résultat accepté avec l'état de l'application qui paraît en retard.

Symptôme	Premiers contrôles
Le mobile ne voit pas le catalogue	Adresse serveur, restaurant actif, authentification si requise, visibilité mobile, disponibilité.
La commande mobile n'apparaît pas en caisse	Confirmation centrale, bonne référence, restaurant, filtres et récupération des commandes distantes.
Une vente POS manque dans le back-office	File locale, authentification backend, correspondance de commande et résultat de synchronisation.
Le livreur ne voit pas son affectation	Compte actif, fiche Delivery reliée, restaurant, date d'affectation et présence du QR accepté.
Le QR est refusé	Commande ouverte, affectation active, bon livreur, secret, état serveur et ancienne réaffectation éventuelle.
Une action reste en attente	Réseau réel, réponse du serveur, session, délai de reprise et état pending/failed/conflict.
Deux totaux ne correspondent pas	Même période, session, restaurant, canal, annulations, paiements et fraîcheur des données.
Le serveur répond 404 ou renvoie du HTML	Adresse, chemin API, version de contrat et serveur web réellement atteint.

Donner au support un dossier exploitable

Conserver le message d'erreur exact et une capture utile de l'état, avec ses filtres. Identifier si l'opération est encore seulement locale, acceptée au serveur ou déjà visible dans une autre application. Masquer les secrets, les mots de passe et les informations personnelles inutiles.

Attention. Éviter les suppressions de files, les réinstallations et les créations répétées avant ce diagnostic. Ces actions peuvent effacer la preuve du problème ou produire des doublons.

Sources : B01 B05 B06 B09 C03 C04 M01 M02 L02 L03.

Recette complète de la plateforme

Une grille de validation pour une installation d’essai

Préparer des cas identifiables

Choisir un restaurant de recette, des comptes par métier, un petit catalogue et des montants reconnaissables. Pour chaque cas, conserver la référence, le résultat attendu, le résultat constaté, les versions et la date du test. Les contrôles ci-dessous constituent un programme de recette ; ils n’ont pas été exécutés lors de la rédaction.

Cas	Résultat à constater
Authentification et périmètre	Chaque profil accède à son outil ; les données d’un restaurant non autorisé sont refusées.
Catalogue par canal	Le même produit apparaît ou reste masqué selon les choix POS/mobile/web.
Commande mobile	Une référence centrale est retournée et retrouvée par le restaurant.
Reprise de création	Le renvoi de la même demande ne crée pas une deuxième commande dans le contrat concerné.
Prix et avantage	Le détail serveur correspond aux règles actives ; l’usage est lié à la commande créée.
Commande POS et synchronisation	La commande locale est rapprochée d’une commande centrale sans changer d’identité locale.
Affectation et QR	Le bon livreur reçoit l’affectation ; un QR étranger ou ancien est refusé.
Livraison et retour	Les deux chemins mettent à jour les statuts attendus et restent distincts du paiement.
Coupure réseau	Les actions prises en charge par les files restent visibles en attente et reprennent sans doublon.
Réservation	La demande puis la décision et la table concernent le bon restaurant.
Paiement et clôture	Les montants se rapprochent sur le même périmètre, avec les crédits et annulations expliqués.
Sauvegarde et reprise	La restauration isolée permet de relire les données et les correspondances nécessaires.

NB. Un résultat doit être observable des deux côtés du contrat : par exemple, confirmation de livraison sur le téléphone et état correspondant dans Laravel. Un code HTTP réussi sans contrôle de la bonne commande est insuffisant.

Sources : B01 à B10 C01 à C07 M01 à M05 L01 à L04.

Exemple complet du client au rapprochement

Une commande fictive suivie sur les quatre applications

De la demande client à la préparation

Un client choisit le restaurant A et commande deux articles. Le serveur retient un sous-total de 200 MAD, une remise de commande de 20 MAD et des frais de 15 MAD, compensés dans cet exemple par une réduction de livraison de 15 MAD. Le total calculé est de 180 MAD. Les montants de l’exemple ne décrivent pas une offre commerciale réelle.

Le mobile envoie sa demande et conserve la clé de cette tentative. Laravel crée la commande centrale 845 puis renvoie sa référence. Le restaurant la retrouve dans son périmètre. La caisse peut la représenter par un identifiant local différent, par exemple 42, tout en conservant sa correspondance avec 845.

Du comptoir à la livraison

L’équipe prépare la commande et choisit le livreur. La fiche centrale de ce livreur est reliée à son compte. L’affectation est acceptée avec sa clé et son QR. Le livreur ouvre la commande sur son application et scanne le support de cette affectation.

Si le réseau est coupé au moment de la réception, l’application conserve l’événement en attente. À la reprise, elle renvoie le même événement. Après acceptation, l’affectation passe à picked_up et la commande à out_for_delivery. Le livreur peut ensuite déclarer la livraison ; le serveur contrôle l’étape avant de la confirmer.

Du suivi de livraison au suivi d’argent

La commande est maintenant livrée, mais son paiement doit encore être lu séparément. Si le client a payé à réception, le restaurant rapproche ce règlement réel dans son parcours autorisé. Si le paiement était déjà enregistré, il ne faut pas le compter une seconde fois lors du retour du livreur.

Le gérant compare enfin les rapports sur le restaurant A et la bonne période. Il vérifie la commande 845, ses états, son règlement et les éventuelles actions en attente. Il n’ajoute pas les 180 MAD affichés en caisse, dans le mobile et chez le livreur comme trois ventes : ces écrans décrivent la même commande.

À retenir. Une seule opération commerciale peut avoir plusieurs représentations. Les références, les statuts et le périmètre permettent de les rapprocher sans les compter plusieurs fois.

Sources : B03 B04 B05 B09 C03 M02 L03.

Vocabulaire de la plateforme

Les termes utiles dans les écrans et les échanges techniques

Terme	Signification dans ce manuel
Backend	Application centrale Laravel qui expose les interfaces, applique les contrôles et conserve les données partagées.
Back-office	Interface web de gestion destinée aux profils autorisés.
API	Contrat HTTP entre applications ; la réponse exprime un résultat ou une erreur.
Restaurant actif	Établissement sélectionné ou autorisé pour les données consultées.
Canal	Origine d'une commande : POS, web, API/mobile ou autre valeur prise en charge.
Fulfillment	Mode de service : sur place, retrait ou livraison selon le contrat.
Base locale	Données conservées sur le terminal, distinctes de la base centrale.
Synchronisation	Échange qui transmet, récupère ou rapproche des données selon un contrat.
File persistante	Événements conservés pour une reprise au-delà de l'action immédiate.
Idempotence	Reconnaissance d'une même opération lors de ses reprises.
Affectation	Lien opérationnel entre une commande et une fiche de livreur.
QR de livraison	Preuve d'une affectation utilisée pour demander la réception au backend.
QR promotionnel	Support d'accès à une promotion ; il ne valide pas une prise en charge.
Pending	Selon le contexte, opération en attente ou état métier encore en attente.
Conflict	Une demande locale n'est pas cohérente avec l'état accepté ou les règles du serveur.
Crédit	Créance à recouvrer, distincte d'une somme déjà encaissée.
Recette	Vérification organisée de scénarios avec résultats observables.

NB. Le même mot peut désigner des objets différents selon son écran. Toujours identifier s'il s'agit du statut de la commande, du paiement, de la livraison ou de sa synchronisation.

Sources : B01 B02 B05 B06 C05 L03.

Repères dans les sources du backend

Index technique pour retrouver les règles décrites

Les chemins ci-dessous sont relatifs à **backend_platform/**, sauf la référence d'organisation à la racine du dossier de travail. Les identifiants B sont utilisés à la fin des chapitres pour relier l'explication au composant consulté.

ID	Emplacements et domaine vérifié
B01	routes/api.php et routes/web.php — Contrats historiques et v1, surfaces web, groupes d'authentification et permissions.
B02	app/Models/Restaurant.php, User.php, Order.php et modèles associés — Relations, rôles normalisés et accès restaurant.
B03	app/Services/Orders/OrderService.php, OrderPricingService.php ; app/Services/Catalog/ProductPriceResolver.php — Création par origine, calculs, avantages et prix par type.
B04	app/Services/Orders/OrderStatusTransitionService.php ; app/Services/Payments/PaymentService.php — Transitions métier, paiements et événements enregistrés.
B05	app/Services/Delivery/ ; app/Http/Controllers/Api/OrderDeliveryController.php ; app/Models/OrderDelivery.php — Affectation, QR, livreur, réception, fin de livraison et idempotence.
B06	app/Services/Sync/SyncService.php ; app/Http/Controllers/Api/SyncController.php — Appareils, opérations, doublons, versions et échanges de synchronisation.
B07	app/Http/Controllers/Api/ReservationController.php ; app/Models/Reservation.php — Création, décisions et rattachement de table.
B08	app/Http/Controllers/Api/CatalogController.php ; app/Models/Product.php — Visibilité POS/mobile/web, disponibilité et échanges catalogue.
B09	app/Http/Controllers/Admin/InfrastructureController.php et OperationsController.php — Rapports, filtres, opérations, sessions et indicateurs.
B10	app/Http/Controllers/StorefrontController.php et HomepageController.php — Site public, panier, réservation et QR promotionnel.

Référence d'organisation. ARCHITECTURE_REFERENCE.md décrit l'orientation vers une installation indépendante par entreprise. Les guides d'architecture existants ont servi à orienter la lecture ; les explications opérationnelles de ce manuel ont été confrontées au code présent.

Repères dans les sources des applications

Index technique de la caisse du mobile et de la livraison

Les chemins de la deuxième colonne sont relatifs au dossier de l'application indiqué. Les regroupements évitent de reproduire toutes les classes ; les éléments détaillés sont conservés dans les sources de fabrication du manuel.

ID	Emplacements et domaine vérifié
C01	caisse_soya : lib/main.dart et lib/services/database_service.dart — Démarrage, persistance locale Isar.
C02	caisse_soya : lib/controllers/pos_controller.dart et lib/views/ — Saisie, tables, parcours personnel.
C03	caisse_soya : lib/services/sync_queue_service.dart et api_order_pull_service.dart — File d'envoi, récupération, correspondances.
C04	caisse_soya : services delivery_qr_assignment_service.dart, delivery_qr_secret_store.dart et order_delivery_api_service.dart — Création QR, secret et affectation.
C05	caisse_soya : lib/models/pos_order.dart, order_delivery.dart, user.dart et delivery.dart — Identités locales et références distantes.
C06	caisse_soya : services de rapport, session, crédit et synchronisation quotidienne — Argent, clôture et historiques.
C07	caisse_soya : services de paramètres, session et réservations — Configuration et contexte connecté.
M01	soyabox_mobile : lib/main.dart, helpers/dependencies.dart, api/api_client.dart, utils/app_constant.dart — Organisation, serveur, session et réseau.
M02	soyabox_mobile : controllers/order_controller.dart, data/order_repo.dart et screen/cart/ — Création client et suivi de commande.
M03	soyabox_mobile : services/restaurant_context.dart, contrôleurs catalogue et dépôts associés — Restaurant et catalogue.
M04	soyabox_mobile : contrôleurs et dépôts réservation et reward — Réservations et avantages.
M05	soyabox_mobile : services/notification_service.dart et deep_link_service.dart — Notifications et liens entrants.
L01	livraison_app : lib/core/config/, lib/features/auth/ et lib/app/ — Configuration, compte et session.
L02	livraison_app : lib/features/home/data/delivery_orders_service.dart — Liste et API de livraison.
L03	livraison_app : lib/features/home/data/receive_action.dart, receive_action_store.dart et receive_sync_queue.dart — Événements persistants et reprises.
L04	livraison_app : lib/features/home/presentation/ et models/ — Écrans, regroupements et lecture des statuts.
T01	theme_platform : routes/web.php — Projet distinct d'édition d'accueil.

Périmètre vérifié et utilisation du manuel

Ce que ce document permet d’affirmer

Un manuel fondé sur l’état des sources

La rédaction s’appuie sur la lecture des routes, des modèles, des contrôleurs, des services et des interfaces des projets présents le 19 octobre 2026. Les explications couvrent le backend opérationnel, le mobile client, la livraison et la caisse. Le site public et le projet de présentation distinct sont situés dans cet ensemble pour éviter les confusions.

Les applications et leurs données n’ont pas été modifiées pour produire ce document. Le guide Caisse Pro existant sert de référence de couverture, de typographie, de couleurs et de pagination. Ce deuxième manuel reste un document séparé.

Distinguer architecture et validation d’exploitation

La présence d’un endpoint, d’un service ou d’une file établit le mécanisme prévu par le code. Elle ne prouve pas que toutes les versions distribuées, tous les téléphones et tous les serveurs ont été testés ensemble. Les chapitres de mise en service et de recette donnent les contrôles à réaliser sur l’installation concernée.

Les adresses de production, les certificats, les sauvegardes effectives, la configuration Firebase, les imprimantes, les stores mobiles et les performances en charge ne sont pas attestés par ce travail documentaire. Les procédures proposées doivent être adaptées et validées par le responsable de l’installation.

Garder un périmètre de présentation exact

La couverture fournie présente l’écosystème et ses domaines d’évolution. Les mentions de stock, comptabilité ou ressources humaines figurant sur ce visuel ne sont pas présentées ici comme des modules intégrés et validés des quatre applications examinées. Les démonstrations web et les anciens documents SaaS ne constituent pas non plus une preuve de parité avec la caisse desktop ou d’un SaaS en exploitation.

NB. Lors d’une mise à jour, revoir les contrats, les statuts, les écrans et les références de code concernés avant de réutiliser une procédure. Un exemple de ce manuel sert à expliquer un fonctionnement ; il ne doit pas être appliqué comme une commande de réparation de données.

Édition. Plateforme Flaravel — Architecture et relations entre backend, mobile client, livraison et caisse. Version documentaire du 19 octobre 2026.