BY+ · CONSOLE NETBYUS · MISE EN PLACE

# La console NetByUs : plan de mise en place

Ce que la console doit faire, ce que core-api sait déjà faire, ce qu'on en reprend dans la nouvelle API, comment la console pilote des serveurs séparés (SaaS et dédiés) par des licences signées, et l'ordre de construction.

---

S'appuie sur « L'architecture de BY+ », le « Catalogue des modules », « Le socle BY+ : guide de mise en place », et la relecture du code de core-api.
Code relu : byplus/api (branche new-develop, après la fusion de Feature/Console/Fondation) ; core-api : develop, by-plus-access-develop, Feature/Add/HelperDeskWebhookTicketId, by-plus-shop-mini-staging.
Octobre 2026 · version de référence. Mise à jour du 3 octobre 2026 : serveurs séparés (instances), lien console–instance avec webhooks, licences signées, versions des serveurs dédiés (chapitres 7 et 8, décisions D16 à D23), et avancement des lots 1 à 4. Mise à jour : la console devient un package installé par une commande sur le seul serveur NetByUs (lot 11), et le suivi des tâches (chapitre 18).

# 1 Ce qu'on construit, et quand c'est terminé

> **Objectif.** Construire la console NetByUs dans la nouvelle API ( `platform/` ), pour que l'abonnement d'un client décide des modules qu'il a actifs : NetByUs crée le client, lui attribue un abonnement, et BY+ active automatiquement les bons modules, **sur le serveur qui héberge le client** (SaaS ou dédié). On reprend ce que core-api sait déjà faire, sans recopier ses défauts.

**Terminé quand (phase 1) :** on crée un client, on lui attribue un abonnement avec POS et Entrepôt pour 10 utilisateurs ; Catalogue et Contacts s'activent seuls (modules requis) ; un utilisateur du client se connecte sur son domaine, ne voit que ces modules, et ne peut ni créer un 11e utilisateur ni atteindre un module non souscrit. Avec un serveur séparé, cela passe par une licence signée par la console (lot 3 bis). Ces actions passent par les routes de l'API de la console ( `/api/v1/console/…` ) ; leur authentification (chapitre 17) sera branchée plus tard.

| Phase | Contenu | Terminé quand |
|---|---|---|
| **1. Console minimale** | Clients et déploiement, catalogue, abonnements, réconciliateur, journal, routes de l'API console ; puis instances, lien et licences signées (lot 3 bis) | Le critère ci-dessus, avec un serveur SaaS séparé |
| **2. Vente** | Calculateur de prix, commandes, paiement FedaPay / KkiaPay, factures, codes promo, cycle de vie (essai, grâce, lecture seule), guichets de « Mon abonnement » | Un client paie en ligne et son abonnement s'active seul |
| **3. Exploitation et ouverture** | Migrations et sauvegardes suivies par instance, archivage, support et accès support, applications externes, annonces, reprise des clients actuels | Les clients actuels tournent sur la nouvelle API avec un abonnement |
| **Ensuite** | Accès à la console : comptes des équipes, connexion jumelée LDAP ou `netbyus` , double authentification obligatoire (chapitre 17) | Un agent inscrit se connecte avec double authentification |

**À retenir**

- **La console ne part pas de zéro.** core-api sait déjà vendre : catalogue d'offres, devis, commande, paiement FedaPay et KkiaPay vérifié par webhook, facture PDF, codes promo, clés API, webhooks, tickets. Mais ce code vend des licences d'applications externes (Shop, ECU, Influencer) à des contacts du tenant `netbyus` , pas des abonnements BY+ à des clients. **À terme, tout sera dans la console**, y compris ces ventes (D7).
- **Le lien central n'existait pas.** Dans core-api, rien ne relie ce qu'un client paie et ce qu'il a d'actif : un client s'active lui-même ses modules ( `POST install/modules` ), sans licence ni quota.
- **Plusieurs serveurs.** La console et le client `netbyus` sont sur le serveur NetByUs ; les clients SaaS sur un autre ; un client peut avoir son serveur dédié. La console les pilote tous par des **licences signées** que seule NetByUs peut créer (chapitres 7 et 8).
- **On construit à neuf dans `platform/`** , avec des tables propres en base centrale, en recopiant les algorithmes éprouvés (devis, passerelles de paiement, facture PDF, promotions) et en corrigeant leurs défauts.

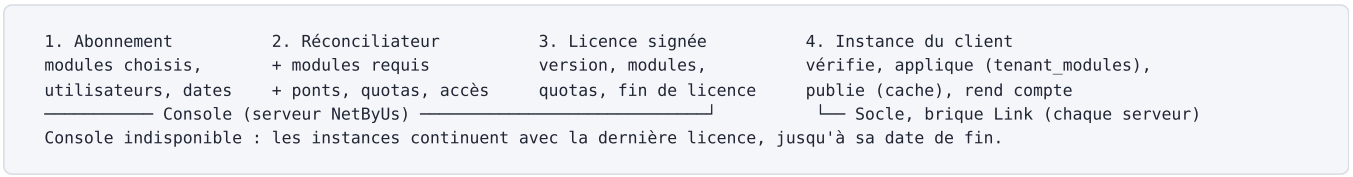
# 2 Ce que fait la console

La console est le back-office de NetByUs : les écrans où les équipes gèrent tous les clients de BY+, comme l'espace d'un opérateur téléphonique. Elle vit sur le serveur NetByUs, avec sa base centrale ; elle ne voit jamais les données métier des clients.

| Rôle | Ce que ça couvre | Phase |
|---|---|---|
| **Déployer** | Créer un client sur son instance (SaaS ou dédiée) : sa base, son domaine, son premier administrateur. Suivre les migrations et les sauvegardes. Archiver. | 1, puis 3 |
| **Vendre** | Le catalogue (modules publiés, prix, options, promotions) et le calcul du prix d'une offre composée par le client. | 1, puis 2 |
| **Suivre** | L'abonnement de chaque client : ses modules, son nombre d'utilisateurs, ses dates, ses commandes, factures et paiements. | 1, puis 2 |
| **Activer** | Mettre chez chaque client exactement les modules que son abonnement prévoit, avec leurs modules requis et leurs ponts, par une licence signée. | 1 |
| **Assister** | Traiter les tickets ; entrer chez un client seulement avec son accord, pour un temps limité. | 3 |
| **Ouvrir** | Enregistrer les applications externes qui appellent l'API (applications NetByUs, partenaires) : leurs clés, leurs droits, leur utilisation. | 3 |

## Les trois règles qui font tenir l'ensemble

1. **L'abonnement est la source de vérité.** Il dit ce que le client doit avoir : ses modules choisis, son nombre d'utilisateurs, ses dates. Personne n'active un module à la main : on modifie l'abonnement, le reste suit.
2. **Un réconciliateur calcule.** Il calcule l'état voulu (modules choisis + modules requis + ponts, quotas, accès) et en fait une licence signée. Il peut être relancé sans risque ; un passage de nuit rattrape les dérives.
3. **Une seule sortie vers les clients : la licence signée.** Chaque instance vient chercher la licence de ses clients, la vérifie et l'applique ; le BY+ du client ne lit que son état local, déduit de cette licence. Il n'interroge jamais les tables de la console.



## 3 Le point de départ : la nouvelle API

| Élément | Ce qu'il apporte à la console | État |
|---|---|---|
| Noyau : `Context`, `Actor` (user, system, api_key, staff), `BricksServiceProvider`, `RouteLoader` | Le cadre des routes centrales et l'acteur de chaque action | en place |
| Tenancy : `tenants`, `domains`, base par client | La création d'une base existe (Stancl), sur chaque instance | en place |
| Registre des modules : manifestes, `requires`, `sellable`, `resolve()` | Le résolveur des exigences ; le réconciliateur l'appelle | en place |
| `tenant_modules` + `origin`, état publié versionné (`tenant_states`, cache) | L'état réel sur l'instance ; écrit uniquement par l'application de la licence | en place |
| Guichets `ModuleActivation`, `ClientProvisioner`, `UserProvisioner`, `ClientState` | La console (ou la brique Link) n'écrit chez un client que par eux | en place |
| Middleware `module:{clé}` sur les routes des modules | Un module non souscrit renvoie 403 | en place |
| Auth : `core_users`, quota d'utilisateurs, connexion LDAP par client | Le premier administrateur est créé là ; le quota s'y applique | en place |
| Horizon (files urgent, courant, lourd), Telescope, Docker de production avec Supervisor | Les traitements longs de la console (passage de nuit, déploiements) partent en file « heavy » | en place |
| `platform/` | Lots 1 à 3 et une grande partie du lot 4 (Chacour) | en place |
| Instances, brique Link, licences signées | Le pilotage de serveurs séparés (chapitres 7 et 8) | lot 3 bis |

## 4 Ce que core-api sait déjà faire

Relu dans le code. « Branche » indique où se trouve la version la plus complète. Sauf mention, ces tables sont dans la base du tenant `netbyus`, pas dans la base centrale.

| Domaine | Où (branche · module) | Tables | Services et comportement |
|---|---|---|---|
| Gestion des applications BY+ | develop · Module (base centrale) | `suit_apps`, `modules`, `module_installs` | `SuitAppsEnum` / `SuitModulesEnum` codés en dur ; `installModules()` bascule installer/supprimer, sans licence ; `getSeeders()` jamais appelé |
| Applications externes | by-plus-access-develop · Application | `applications`, `application_modules`, `application_packs`, dépendances, permissions, `storage_offers`, `storage_subscriptions` | Catalogue des applications vendues (Shop, ECU, Influencer), packs, `ModuleDependencyResolver`, options de stockage |
| Licences | id. · License | `licenses` (propriétaire : un contact), types, durées, activations, modules, `license_offer`, usages, `contact_applications` | Statuts pending, active, expired, revoked, suspended ; validate, activate, check-option ; jeton signé en HMAC-SHA256 |
| Offres et prix | Feature/Add/HelperDesk... · License | `offer_types`, `service_offers`, `service_offer_prices` | `CompositeOfferService` ; `LicenseCheckoutService::quote()` |
| Commande et paiement | id. · License | `checkout_orders` (montants `decimal(10,2)`), `payment_aggregator_configs` | FedaPay, KkiaPay ; webhooks signés ; contrôle du montant payé ; facture PDF ; envoi par mail et WhatsApp |
| Promotions | id. · PromoCode | `promo_codes`, `promo_code_usages` | `PromoCodeService::resolve()`, rattaché aux influenceurs |
| Déploiement des clients | develop · Central | `tenants`, `domains`, `tenant_information`, `legal_pages` | `CreateANewTenantJob` : un seul job, cache:clear, une quinzaine de seeders ; commandes Migrate, Rollback, SaveDataBase, DropTenant ; inscription publique |
| Sauvegardes et stockage | by-plus-shop-mini-staging · TenantBackup | `backups`, `storages` | Sauvegardes par client, place utilisée, demandes payantes |
| Support | by-plus-access-develop · Helpdesk | `helpdesk_tickets`, réponses, catégories, affectations, appels (LiveKit) | Source polymorphe du ticket ; notifications Mattermost ; temps de première réponse |
| Applications externes et intégrations | id. · ExternalApplications, Webhook | `contact_applications` (type contact ou external), clés pk/sk, scopes, usages, `external_api_endpoint_configurations`, `webhook_subscriptions` | `AuthenticateExternalApplication`, `EndpointScopeManager`, statistiques, `ApiKeyService` ; `WebhookDispatcherService` envoie sans garder de trace |
| Distributeurs | id. · Influencer, License | `application_influencers` | Vente par des influenceurs ; repris dans la console à terme (D7) |

## 5 Les constats qui comptent pour la console

| Constat | Où | Conséquence pour la console |
|---|---|---|
| Montants en décimal et calculs en float | `checkout_orders.amount`, `quote()`, `PromoCodeService`, facture PDF | Montants en entiers (F CFA) ; prix figés à la commande |
| Jeton de licence signé avec une clé partagée (HMAC) | `LicenseTokenService` | Signature asymétrique Ed25519 : un serveur vérifie la licence sans pouvoir en créer (chapitre 8) |
| Pas d'état « essai » ni « grâce » ; aucune tâche d'expiration | `LicenseStatus` | Cycle de vie complet et tâche quotidienne |
| Le titulaire d'une licence est un contact, pas un client | `License::contact()` | L'abonnement est rattaché au client |
| Webhooks de paiement vérifiés mais non dédoublonnés | `LicenseCheckoutController` | Journal des événements reçus, unique par référence |

| Constat | Où | Conséquence pour la console |
|---|---|---|
| Création d'un client en un seul job, non rejouable, avec cache:clear global | `CreateANewTenantJob` | Déploiement en étapes rejouables et suivies |
| Clés de passerelle de paiement en clair | `payment_aggregator_configs` | Colonnes chiffrées |
| Envoi de webhook sans trace ni nouvelle tentative | `WebhookDispatcherService` | Journal des livraisons, nouvelles tentatives (aussi pour le lien avec les instances) |
| Secrets versionnés dans Git | Historique des dépôts | Aucun secret dans le code ; clé privée de licence uniquement sur le serveur NetByUs |
| Routes centrales ouvertes | `Central/routes/api.php` | Aucune route d'écriture de la console sans protection |

# 6 Où va la console dans la nouvelle API

La console a la même forme que le socle : un dossier par domaine, chacun avec sa fiche ( `manifest.php` ), ses routes, son guichet ( `Public/` ), ses `Services/` , son dossier `lang/` et ses tables.

> **Un package, installé sur le seul serveur NetByUs.** À la fin de la construction (lot 11), la console sort du dépôt de l'API et devient un package Composer privé, `netbyus/byplus-console` . Il s'installe par une commande, uniquement sur le serveur NetByUs :
>
> ```
> composer require netbyus/byplus-console
> php artisan byplus:console:install
> ```
>
> Le serveur SaaS et les serveurs dédiés n'ont jamais le code de la console : un client qui a la main sur son serveur ne voit ni la console, ni la façon dont les licences sont signées. **C'est fait (T30) :** le code est dans `packages/byplus-console/src/` (espace de noms `Platform\` , fournisseur découvert automatiquement). Dans le dépôt de l'API, il est installé en développement ( `require-dev` , dépôt « path ») pour les tests ; un serveur installé avec `composer install --no-dev` ne l'a pas. Les écrans de la console sont un package du front, `@netbyus/byplus-console-app` : seul le build du front du serveur NetByUs l'inclut ( `VITE_WITH_CONSOLE=1` ), et il ne s'affiche que dans le shell de BY+ du client `netbyus` , pour un agent inscrit (entrée « Console NetByUs » du menu).

```
packages/byplus-console/src/    LA CONSOLE · namespace Platform\ · base centrale du serveur NetByUs
├── Kernel/         ● fait       chargement des domaines, garde console, contrôleur et tableaux communs
├── Tenants/        ● fait       clients, profil, contacts, étapes de déploiement
├── Catalog/        ● fait       modules publiés, grilles de prix, options, promotions, calculateur
├── Subscriptions/  ● fait       abonnement de chaque client, historique
├── Activation/     ● fait       réconciliateur (passage de nuit en file « heavy »)
├── Audit/          ● fait       journal des actions de la console
├── References/     ● fait       pays, devises, fuseaux pour les écrans
├── Instances/      ● fait       registre des serveurs, enrôlement, lien, webhooks, comptes rendus
├── Licensing/      ● fait       clé de signature, licences signées par instance et par client
├── Orders/         ● fait       commandes, devis figés
├── Billing/        ● fait       factures, paiements, passerelles, webhooks
├── (Instances)     ● fait       migrations, sauvegardes, archivage (par instance)
├── Support/        ● fait (API) tickets de toutes les applications, widget, accès support, appels vidéo
├── Applications/   ● fait       applications externes : registre, clés pk/sk, scopes, utilisation
├── Announcements/  ● fait       annonces aux clients
└── Staff/          ● fait       agents NetByUs, rôles, connexion jumelée, double authentification

core/Licensing/     ● fait       (socle, sur chaque serveur) vérifier, garder et appliquer la licence ; « Ma licence » du client
core/Link/          ● fait       (socle, sur chaque serveur) parler à la console : enrôlement, interrogation, webhook, comptes rendus
```

| Règle | Pourquoi |
|---|---|
| Chaque domaine a une fiche `manifest.php` ; ses routes sont servies sous `/api/v1/console/{clé}` | Même chargeur que les briques ; pas de route client |
| Toutes les tables de la console sont en base centrale du serveur NetByUs | La console ne lit jamais une base client, ni la base d'une autre instance |
| La console n'agit sur un client que par sa licence et ses demandes de déploiement | Le socle de chaque instance reste maître de ses tables ; Deptrac le vérifie |
| Le socle n'importe jamais `Platform\` | Un client lit sa licence, pas la console |
| Chaque action est une Action appelée par une route de l'API console, derrière le garde console | Seul le garde change quand l'accès sera choisi (chapitre 17) |
| Montants en entiers, dates en UTC, identifiant du client = chaîne lisible (ex. `sdp` ) | Pas de calcul flottant |
| Traductions par domaine ( `lang/fr` , `lang/en` ), appelées par la clé du domaine ( `tenants::messages.…` ) | Comme le socle |
| La console n'utilise du socle que ses guichets ( `Public/` ) et le noyau ; elle a ses propres migrations, routes, traductions et configuration | Elle se sort dans un package sans rien changer au socle (lot 11) |

> **Noms des tables.** Chaque domaine de la console a son préfixe : `tenant_`, `catalog_`, `sub_`, `act_`, `inst_`, `lic_`, `ord_`, `bill_`, `ops_`, `sup_`, `app_`, `ann_`, `aud_`, `staff_`. Exceptions : `tenants`, `domains` (Stancl), `tenant_modules` et `tenant_states` (socle), qui vivent sur chaque instance.

## Les applications externes : deux niveaux

Une application externe est un logiciel qui appelle l'API de BY+ avec des clés (pk / sk) : applications de NetByUs (passerelle Access, application mobile de caisse, Influencer, ECU) ou logiciel d'un partenaire.

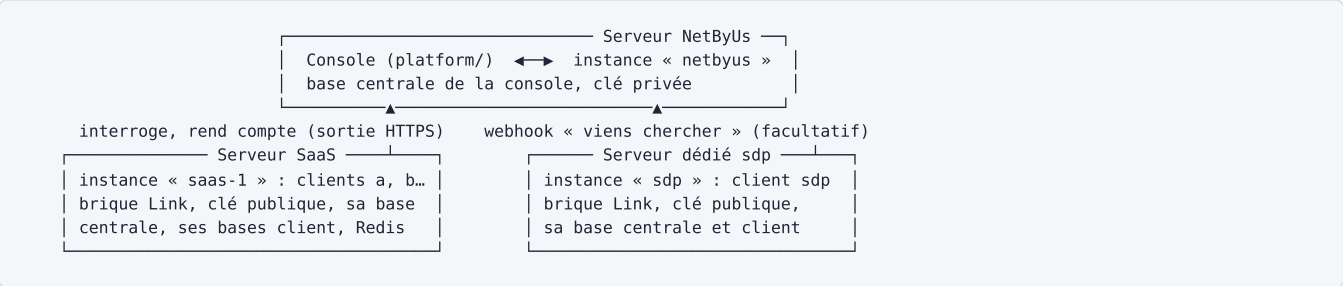
| Niveau | Ce qu'il porte | Exemple |
|---|---|---|
| Console · `platform/Applications` | Le registre : identité, organisation, type, clés, scopes maximum, secret de webhook, journal d'utilisation, suspension | NetByUs enregistre ECU avec les scopes `contacts.*` et `facturation.factures.*` |
| Socle · brique Integrations (côté client) | Le consentement : l'administrateur d'un client autorise une application enregistrée, avec tout ou partie de ses scopes ; il y crée aussi ses propres clés API et webhooks | Le client `sdp` autorise ECU en lecture sur ses contacts |

- Une application n'atteint un client que s'il l'a autorisée, et seulement dans les scopes accordés.
- Les scopes sont les noms des routes ( `{clé}.{action}` ) : plus de table de points d'accès à importer.
- Les applications autorisées d'un client arrivent à son instance avec sa licence ; le middleware du socle vérifie la clé et le scope sur place.
- La passerelle Access devient une application externe enregistrée.

# 7 Les instances : la console et les serveurs séparés

La console NetByUs pilote des **serveurs BY+ séparés**, appelés **instances**. Dans tous les cas, c'est elle qui décide : modules actifs, quotas, accès (actif, grâce, lecture seule), déploiement d'un client.

| Serveur | Ce qu'il héberge | Domaine |
|---|---|---|
| **Serveur NetByUs** | La console NetByUs et le client `netbyus` | `byplus.io` , `netbyus.byplus.io` |
| **Serveur SaaS** | Tous les clients en SaaS, une base chacun | `{client}.byplus.io` |
| **Serveur dédié** (0 à n) | Un client qui veut son propre serveur, chez NetByUs ou chez lui, installé et mis à jour par NetByUs ou par le client (D19) | Son propre nom de domaine, ex. `erp.sdp.bj` |




> **Le même BY+ partout, la console en plus sur un seul serveur.** Toutes les instances ont le même code BY+ (socle et modules). La console est un package installé uniquement sur le serveur NetByUs (lot 11). Un serveur dédié est une instance BY+ normale, avec un seul client et son nom de domaine. Le client `netbyus` suit le même protocole, avec un transport « sur place ».

## Le principe : l'instance vient chercher sa licence (D16)

La console ne se connecte jamais aux bases ni au Redis d'une instance. Chaque instance **interroge la console**, reçoit la licence signée de ses clients et l'applique avec les guichets du socle.

- **Réseau :** un serveur dédié chez un client n'a besoin que d'une sortie HTTPS vers la console, aucun port entrant.
- **Robustesse :** un serveur éteint récupère la dernière licence à son retour ; rien n'est perdu, rien n'est à relancer, pas de problème d'ordre (on reçoit toujours l'état complet).
- **Sécurité :** les instances n'exposent aucune adresse de pilotage ; seule la console est ouverte.
- **Console en panne :** les instances continuent avec la dernière licence, jusqu'à sa date de fin.

## Les webhooks : pour aller vite, et pour rendre compte (D17)

Un webhook est une requête HTTPS signée, envoyée à une adresse enregistrée quand un événement se produit, avec un journal des livraisons et de nouvelles tentatives.

| Sens | Événement | Contenu et règle |
|---|---|---|
| **Console → instance** | `state.changed` , `provisioning.requested` | Un simple signal : type, client, numéro de version, horodatage. **Aucune donnée** : l'instance vient chercher la licence. Si l'instance n'est pas joignable (dédié derrière un pare-feu), l'interrogation régulière rattrape. Facultatif par instance. |
| **Instance → console** | `state.applied` , `provisioning.step` , `instance.report` , `quota.reached` , `error` | Le compte rendu : licence appliquée et résultat, étape de déploiement, version du code et modules présents, utilisateurs actifs, place utilisée, horloge. Vérifié (signature de l'instance) et enregistré. |

| Mécanisme | Délai | Rôle |
|---|---|---|
| Webhook `state.changed` | quelques secondes | Chemin rapide, quand l'instance est joignable |
| Interrogation régulière | 5 minutes (réglable) | Filet de sécurité, toujours actif |
| Au démarrage de l'instance | immédiat | Rattrape une coupure |

| Mécanisme | Délai | Rôle |
|---|---|---|
| Fichier de licence importé | à la main | Serveur coupé de la console (chapitre 8) |

## Les échanges

| Requête | Rôle |
|---|---|
| `POST /api/v1/link/enroll` `{ token, url, public_key, version }` | Instance → console : s'enregistre avec un jeton à usage unique (24 h) créé par un agent |
| `GET /api/v1/link/state` | Instance → console : les licences de ses clients et les demandes de déploiement en attente |
| `POST /api/v1/link/events` | Instance → console : les comptes rendus |
| `POST {url de l'instance}/api/v1/link/webhook` | Console → instance (facultatif) : le signal « viens chercher » |
| `… /api/v1/link/desk/{client}/subscription \| orders \| licenses` | Instance → console : « Mon abonnement » d'un client de l'instance (brique `core/Subscription`), au nom de l'utilisateur ; ses droits sont vérifiés par l'instance. Sur le serveur NetByUs, la même route est appelée sur place, sans réseau |

## Enrôler un serveur dédié ou SaaS

1. Dans la console, un agent crée l'instance ( `POST /api/v1/console/instances` ) : il reçoit un jeton, valable 24 h, une fois.
2. Sur le serveur : `php artisan byplus:link:enroll https://byplus.io <jeton> --url=https://serveur-du-client`. Le serveur crée sa paire de clés, s'enregistre, reçoit les clés publiques de NetByUs et applique les licences de ses clients.
3. Le planificateur ( `schedule:run` ) lance `byplus:link:sync` toutes les 5 minutes ( `BYPLUS_LINK_SYNC_MINUTES` ) : rien d'autre à configurer. Avec `--url`, le webhook « viens chercher » accélère tout ; sans, l'interrogation suffit.
4. Un client se crée sur ce serveur depuis la console ( `instance_id` à la création) : le serveur exécute les étapes du déploiement et les rapporte.

## Sécurité du lien

- **Enrôlement :** jeton à usage unique ; l'instance donne sa clé publique.
- **Deux paires de clés Ed25519 :** celle de la console signe les licences et ses webhooks ; celle de chaque instance signe ses requêtes et comptes rendus. Les clés privées ne quittent jamais leur serveur.
- **Requêtes signées :** méthode, adresse, horodatage, empreinte du corps ; refus au-delà de 5 minutes d'écart et de tout identifiant déjà vu. HTTPS obligatoire.
- **Révocation :** une instance révoquée ne reçoit plus rien ; ses clients gardent leur licence jusqu'à sa date de fin.
- **Cloisonnement :** une instance ne lit et ne rapporte que pour ses propres clients ; une licence ne s'applique que sur le serveur à qui elle est destinée.

# 8 Les licences : seule NetByUs peut en créer

Un client ne doit jamais pouvoir s'activer un module, augmenter son nombre d'utilisateurs ou prolonger son abonnement sans NetByUs, même sur un serveur dédié dont il a l'accès complet.

## Qui fait quoi : la console émet, le socle vérifie

| Où | Rôle |
|---|---|
| **Console** · `platform/Licensing` (serveur NetByUs) | Émettre : licence commerciale (vente, clé, durée), puis licence signée par la clé privée, versionnée, par instance et par client |
| **Socle** · brique `core/Licensing` (chaque serveur, chaque client) | Vérifier et appliquer : clé publique, vérification de la signature, licence en cours gardée, modules, quotas et accès déduits, fin de licence, grâce, lecture seule, horloge, import d'un fichier de licence ; « Ma licence » pour l'administrateur du client (modules, utilisateurs, dates, état, dernier contact) |
| **Socle** · brique `core/Link` (chaque serveur) | Transporter : enrôlement, interrogation de la console, webhook, comptes rendus. Un serveur coupé de la console fonctionne sans elle, avec un fichier de licence |

Chaque client a donc, sur son propre serveur, de quoi contrôler sa licence et ses modules, sans dépendre de la console pour fonctionner ; seule la création d'une licence reste chez NetByUs.

## Le principe : une signature (D22)

| Clé | Où elle est | Ce qu'elle permet |
|---|---|---|
| **Clé privée** de la console | Uniquement sur le serveur NetByUs, chiffrée, sauvegardée hors ligne | **Créer** une licence. Personne d'autre ne peut le faire |
| **Clé publique** de la console | Dans chaque instance (livrée avec le code, confirmée à l'enrôlement) | **Vérifier** une licence, jamais en créer ni en modifier une |

C'est la signature qui protège : un client peut lire sa licence, mais s'il y change un module, un nombre ou une date, la vérification échoue et l'instance la refuse. Rotation de la clé possible sans réinstaller les instances.

## Ce que contient une licence

- L'instance (identifiant, empreinte de sa clé publique) et ses domaines : une licence copiée sur un autre serveur ne vaut rien.
- Pour chaque client : modules, quotas (utilisateurs, stockage), accès (actif, grâce, lecture seule), applications externes autorisées.
- Un numéro de version, la **date de fin de licence** (fin de la période payée) et la date de fin de grâce.

## Sans contact : la licence suffit (D18)

Le serveur fonctionne jusqu'à la **date de fin de sa licence**, même sans contact avec la console ; ensuite grâce (bandeau « renouvelez »), puis lecture seule, comme un client SaaS.

- **Le renouvellement doit arriver jusqu'au serveur :** par le lien, ou par un **fichier de licence** téléchargé dans la console et importé par l'administrateur du client (« Mon abonnement »). Le fichier est signé : aucun risque à le confier au client.
- **Une baisse d'abonnement en cours de période** ne s'applique qu'au contact suivant ou à la licence suivante, sans perte : la licence en cours couvre ce qui a été payé.
- **La durée d'une licence suit la période payée :** un mois pour un client mensuel, un an pour un client annuel.

## Où la licence est vérifiée

La licence est la seule source de vérité sur l'instance : `tenant_modules` et l'état publié en sont **déduits**. Modifier la base à la main ne sert à rien : la vérification se fait en plusieurs endroits du socle (registre des modules, garde `module:{clé}`, menus et `/me/bootstrap`, tâches planifiées, création d'utilisateur), et le passage suivant remet tout en ordre.

## La limite, et comment la réduire (D21)

Sur un serveur qui lui appartient, un client qui modifie le code PHP pourrait sauter la vérification. On ne peut pas l'empêcher totalement ; on le rend difficile et on le détecte, **sans encoder le code** (pas d'ionCube) :

| Mesure | Effet |
|---|---|
| Vérification en plusieurs endroits | Pas un seul « interrupteur » à retirer |
| Intégrité du code | Chaque version livrée a une empreinte signée par NetByUs ; l'instance la contrôle et la rapporte |
| Comptes rendus réguliers | Modules réellement actifs et utilisateurs déclarés : un écart avec la licence est vu dans la console |
| Fin de licence | Prolonger exige une nouvelle licence signée par NetByUs |
| Horloge du serveur | L'instance retient la date la plus récente vue ; si l'horloge recule pour prolonger une licence, elle s'en tient à cette date et le signale |
| Contrat | Le contrat d'un serveur dédié interdit de modifier le code et autorise le contrôle de conformité |

## Les versions des serveurs dédiés (D23)

Un serveur dédié mis à jour par le client peut garder une ancienne version. **Toutes les versions restent supportées**, même sans mise à jour.

| Règle | Ce que ça impose |
|---|---|
| Le lien ne casse jamais | Le protocole du lien porte un numéro de version et n'évolue que par ajouts : on ne retire jamais un champ ni un échange |
| Un module absent du code | Si la licence contient un module que la version du serveur n'a pas, l'instance l'ignore sans erreur et le signale ; il s'active dès la mise à jour, sans nouvelle licence |
| Visibilité | La console affiche la version et les modules présents de chaque instance |
| Support | Normal pour toutes les versions |

# 9 Le cœur : de l'abonnement aux modules actifs

## Ce que fait le réconciliateur, pour un client (console)

1. Lire l'abonnement : modules choisis, état, quotas.
2. Calculer l'état voulu : modules choisis + exigences (registre) + ponts + extensions ; en lecture seule, les modules restent (le client consulte) ; un client archivé n'a plus aucun module.
3. Produire la licence signée, versionnée, pour l'instance du client ; envoyer le webhook `state.changed`.
4. Tracer l'exécution dans `act_runs` (déclencheur, état voulu, résultat rapporté par l'instance).

## Ce que fait l'instance, à la réception (brique Link)

1. Vérifier la signature, l'instance, la version (refus d'une licence plus ancienne) et la date.
2. Calculer l'écart avec `tenant_modules` ; préparer la base du client pour chaque module à activer (droits, rôles, données de départ, rejouables).
3. Écrire l'état réel en dernier ( `tenant_modules` avec son origine, `tenant_states` ) : un module n'est jamais marqué actif sans ses données.
4. Publier : incrémenter la version locale, mettre l'état et les quotas dans le cache, émettre l'événement « état modifié » (menus rechargés).
5. Rendre compte ( `state.applied` ).

| Déclencheur | Exemple |
|---|---|
| Changement d'abonnement | Module ajouté, retiré, quota modifié, abonnement payé, passé en lecture seule ou rétabli |
| Nouvelle version du code | Un module gagne une exigence ou un pont |
| Passage de nuit | À 03:00, un job par client dans la file « heavy » ( `byplus:reconcile --queue` ) |
| Demande d'un agent | `POST /api/v1/console/activation/{client}/reconcile` |

## Ce que lit le BY+ du client (sur son instance)

| État publié | Lu par |
|---|---|
| Version des modules | Le front (rechargement des modules si la version change) |
| Modules actifs | Middleware `module:{clé}` , écouteurs, tâches planifiées, `/me/bootstrap` |
| Quotas | Auth (nombre d'utilisateurs), Files (stockage) |
| Applications autorisées | Middleware des applications externes |
| Accès | Tenancy : actif, grâce (bandeau), lecture seule |

Si le cache est vidé, l'instance reconstruit l'état depuis sa dernière licence : le cache n'est jamais la seule copie.

# 10 L'offre d'un client et son prix

L'offre d'un client est l'ensemble des modules qu'il choisit, enregistré dans son abonnement. Le calcul du prix est fait côté serveur, par un seul calculateur, le même pour la page de tarifs, « Mon abonnement », la console et le paiement.

| Élément | Proposition | Statut |
|---|---|---|
| Prix d'un module | Montant par mois (et par an, avec remise) pour chaque module vendable ; les modules de base sont gratuits | à confirmer |
| Utilisateurs | Prix par utilisateur et par mois, au-delà d'un nombre inclus | à confirmer |
| Formules | Facultatives : une formule peut fixer un plafond de modules, un nombre d'utilisateurs inclus et des remises. Ce sont des données | à trancher |
| Options | Stockage supplémentaire, porte connectée, installation… quantité × prix unitaire | fait |
| Promotions | Codes promo : pourcentage ou montant, limites d'usage par client | fait |
| Grille de prix | Versionnée (date d'effet) : une nouvelle grille ne change pas un abonnement en cours | fait |
| Prorata | Un module ajouté en cours de période est facturé au prorata | Proposé |

## Le cycle de vie d'un abonnement

| État | Ce que voit le client | Comment on en sort |
|---|---|---|
| Essai | Accès complet | Paiement : Actif. Fin d'essai sans paiement : Lecture seule |
| Actif | Accès complet | Fin de licence sans renouvellement : En grâce |
| En grâce | Accès complet, bandeau « renouvelez » | Paiement : Actif. Fin de la grâce : Lecture seule |
| Lecture seule | Le client se connecte, consulte et exporte toutes ses données, mais ne peut plus rien créer ni modifier ; bandeau permanent vers « Mon abonnement » | Paiement : Actif, tout est rétabli. Fin de la conservation : Archivé |
| Archivé | Base sauvegardée puis supprimée, après plusieurs avertissements | État final, sous double validation |

Ces dates sont dans la licence : un serveur coupé de la console passe en grâce puis en lecture seule à l'heure dite. Durées de l'essai, de la grâce et de la conservation : à fixer en atelier.

## La lecture seule, concrètement

| Ce qui reste possible | Ce qui est refusé |
|---|---|
| Se connecter, consulter, rechercher, télécharger ses fichiers, exporter | Toute création, modification ou suppression (POST, PUT, PATCH, DELETE des routes client) : 403 `subscription_read_only` |
| Ouvrir « Mon abonnement », composer et payer, importer un fichier de licence | Créer un utilisateur, synchronisation mobile qui écrit |
| Se déconnecter, changer son mot de passe, ouvrir un ticket | Les tâches planifiées et traitements des modules qui créent des données |

C'est le socle qui applique la règle, une seule fois pour tous les modules, à partir de l'accès donné par la licence.

# 11 Le déploiement d'un client

Remplace `CreateANewTenantJob`. Chaque étape est rejouable et son résultat est noté. La console demande et suit ; **l'instance exécute**.

| # | Qui | Étape | Ce qui se passe |
|---|---|---|---|
| 1 | Console | Enregistrer le client | Profil, contacts, abonnement ; choix de l'instance (SaaS par défaut, ou son serveur dédié) ; demande de déploiement et webhook `provisioning.requested` |
| 2 | Instance | Créer le client et la base | `tenants`, base PostgreSQL du client (Stancl) |
| 3 | Instance | Jouer les migrations | Socle et tous les modules |
| 4 | Instance | Préparer le socle | Première société (pays, devises, unités sont livrés avec le code) |
| 5 | Instance | Premier administrateur | Compte dans `core_users` ; invitation par mail |
| 6 | Instance | Rattacher le domaine | `domains` (principal) |
| 7 | Console → instance | Licence | La licence (modules de l'abonnement) suit le chemin normal |
| 8 | Instance | Ouvrir | État `ready` ; chaque étape a été rapportée ( `provisioning.step` ) dans `tenant_provisioning_steps` |

En cas d'échec, l'étape est rejouée à la demande suivante, sans refaire les précédentes. **Un serveur dédié** est d'abord installé (« Installer l'API BY+ en production avec Docker »), puis enrôlé avec son jeton ; on y déploie ensuite son client comme ci-dessus.

## Exploitation des bases (phase 3)

| Besoin | Repris de | Ajout |
|---|---|---|
| Migrer | `tenants:migrate` (Stancl) | Suivi par instance et par client ; reprise des clients en échec |
| Sauvegarder | SaveDataBase, TenantBackup | Sauvegardes chiffrées, suivies, restauration testée ; rapportées par l'instance |
| Place utilisée | RecalculateTenantStorageUsageJob | Rapportée par l'instance, comparée au quota de la licence |
| Archiver | DropTenant | Sauvegarde vérifiée avant suppression ; double validation |

# 12 Les tables de la console

Base centrale du serveur NetByUs, créées par les migrations du package de la console. Montants en entiers (F CFA), `tenant_id` = chaîne. **fait** : déjà en place (lots 1 à 4).

| Domaine | Table | Colonnes principales |
|---|---|---|
| Tenants | `tenant_profiles` **fait** | tenant_id, raison sociale, IFU, RCCM, pays, devise, adresse, logo ; + **instance_id** et domaine principal (lot 3 bis) |
| | `tenant_contacts` **fait** | tenant_id, nom, email, téléphone, rôle (facturation, technique, décideur) |
| | `tenant_provisioning_steps` **fait** | tenant_id, étape, statut, erreur, tentatives, fin (alimentée par les comptes rendus au lot 3 bis) |
| | `tenants`, `domains` | Sur chaque instance (Stancl) ; + `status`, `deployment`, `is_primary` **fait** |
| Catalog | `catalog_modules` **fait** | module_key, nom affiché, catégorie, description, publié, ordre, essai |
| | `catalog_price_lists` **fait** | version, devise, utilisateurs inclus, effective_from, effective_to, publiée |
| | `catalog_prices` **fait** | price_list_id, cible (module, utilisateur, option), clé, fréquence, montant |
| | `catalog_options` **fait** | clé, nom, unité, récurrente, module lié |
| | `catalog_promotions` · `_usages` **fait** | code, type, valeur, limites, dates · tenant_id, remise appliquée |
| Subscriptions | `sub_subscriptions` **fait** | tenant_id (unique), statut, fréquence, grille, dates, essai, grâce, utilisateurs, renouvellement |
| | `sub_subscription_modules` **fait** | subscription_id, module_key, prix figé, ajouté, retiré |
| | `sub_subscription_options` **fait** | subscription_id, option_key, quantité, prix figé |
| | `sub_events` **fait** | subscription_id, type, avant, après, acteur |
| Activation | `act_runs` **fait** | tenant_id, déclencheur, état voulu, écart, statut, erreur, début, fin |
| Instances | `inst_instances` | clé, nom, type (netbyus, saas, dédiée), URL, clé publique, version du code, version du protocole, modules présents, dernier contact, santé, statut (active, révoquée) |
| | `inst_enrollment_tokens` | instance_id, empreinte du jeton, expire le, utilisé le |
| | `inst_reports` | instance_id, tenant_id, type d'événement, contenu, reçu le (purgé après une durée fixée) |
| | `inst_webhook_deliveries` | instance_id, événement, tentatives, dernier code, statut, prochaine tentative |
| Licensing | `lic_signing_keys` | identifiant, clé publique, clé privée (chiffrée), active, rotation |
| | `lic_licenses` | instance_id, tenant_id, version, contenu signé, fin de licence, fin de grâce, émise le, appliquée le |
| Audit | `aud_logs` **fait** | acteur, action, cible, tenant_id, avant, après, ip |
| Orders | `ord_orders` · `ord_order_lines` | tenant_id, statut, devis figé, totaux, source, expiration · lignes |
| Billing | `bill_invoices` · `_lines` | numéro, tenant_id, order_id, période, totaux, état, PDF |
| | `bill_payments` · `bill_gateway_events` | passerelle, référence (unique), montant, état · événements reçus (unique) |
| | `bill_gateway_configs` | passerelle, clés chiffrées, secret de webhook, sandbox |
| Operations | `ops_migration_runs` · `ops_backups` | par instance et par client : statut, erreur, emplacement, vérification |
| Support **fait** | `helpdesk_*` (noms de core-api gardés) | tickets, messages, catégories, déclarants, widget, accès accordés, appels : voir `support-et-widgets.pdf` |

| Domaine | Table | Colonnes principales |
|---|---|---|
| Applications **fait** | `app_applications` · `app_instances` | toutes les applications NetByUs, BY+ compris ( `kind` platform ou product), droits demandables ; les événements de webhook sont ceux de la plateforme, dans le code ( `WebhookEventEnum` , repris de core-api) · une instance par application et par client (hébergement, adresse, état) |
| | `app_instance_keys` · `app_instance_webhooks` | clés Ed25519 de l'instance (clé privée montrée une fois ; les clés publiques voyagent dans la licence) · webhooks de l'instance (secret chiffré, signature HMAC-SHA256, dernier envoi) |
| Catalog | `catalog_modules` · `catalog_options` · `catalog_price_lists` · `catalog_storage_offers` | chaque ligne porte `application_key` : chaque application a son catalogue (modules et leurs permissions, options chiffrées), ses prix à la carte (grille versionnée) et ses offres de stockage |
| | `catalog_offers` · `catalog_offer_prices` | les offres fixes de NetByUs : modules avec toutes ou une partie de leurs permissions, utilisateurs, options chiffrées (caisses, carnets…), état (brouillon, en vente, archivée) · leurs tarifs (durée, hébergement, prix pour toute la durée). L'offre fait la licence : `lic_license_offers` et `sub_subscriptions` gardent `offer_id` et `permissions` , la licence signée porte `offer` , `permissions` et les quotas (utilisateurs et options) |
| Announcements | `ann_announcements` · `ann_reads` | titre, corps, cible, dates · lectures |
| Staff | `staff_users` · `staff_roles` · `staff_challenges` · `staff_tokens` | email, fournisseur (ldap, netbyus), identifiant externe, actif, rôle, droits ajoutés et retirés ( `rights_granted` , `rights_revoked` ) ; rôles modifiables avec leurs droits ; défis de connexion avec le code envoyé par e-mail (haché, 60 s) |

# 13 Ce qu'on reprend de core-api, et comment

« Recopier » : reprendre l'algorithme en l'adaptant. « Concept » : reprendre l'idée, réécrire. « Non » : ne pas reprendre.

| Existant | Cible | Mode | Changement |
|---|---|---|---|
| `SuitAppsEnum`, `SuitModulesEnum` | manifestes des modules | non | Le catalogue vient des manifestes (déclarés par les instances) |
| `installModules()` | — | non | Plus d'auto-activation : abonnement puis licence |
| `ModuleDependencyResolver` | `ModuleRegistry::resolve()` | fait | Le registre résout les exigences et détecte les cycles |
| `LicenseStatus`, `LicenseService` | Subscriptions | concept | Essai et grâce ; titulaire = client ; quotas dans la licence |
| `license_offer` | prix figés | concept | Prix figés par ligne |
| `quote()`, `CompositeOfferService` | calculateur (Catalog) | fait | Entiers, prix par module et par utilisateur, grille versionnée |
| `checkout_orders`, `initiateSubscription()` | Orders | recopier | Lignes de commande, devis figé, client déduit du domaine |
| FedaPay, KkiaPay, webhooks | Billing | recopier | Dédoublonnage, montants entiers |
| `payment_aggregator_configs` | `bill_gateway_configs` | recopier | Clés chiffrées |
| Facture PDF, mails et WhatsApp | Billing | recopier | Numérotation continue, PDF dans MinIO, brique Notifications |
| `PromoCodeService` | Catalog | fait | Sans lien influenceur ; limite par client |
| `CreateANewTenantJob` | Tenants + instance | concept | Étapes rejouables, exécutées par l'instance (chapitre 11) |
| `tenant_information` | `tenant_profiles` | fait | Responsable → `tenant_contacts` |
| Migrate, SaveDataBase, DropTenant, TenantBackup | Operations | concept | Suivi par instance et par client, double validation |
| Helpdesk | Support | concept | Source = client et module ; accès support limité dans le temps |
| `contact_applications` (external) et ses services | console · Applications | recopier | Secrets hachés, scopes = noms de routes, consentement du client |
| `ApiKeyService`, `WebhookDispatcherService` | brique Integrations | concept | Clés d'un client, journal des livraisons |
| `LicenseTokenService` (HMAC) | Licensing | concept | Signature Ed25519, clé privée chez NetByUs seulement, rotation |
| Vente de Shop, ECU, Influencer ; distributeurs ( `DistributorLicenseController` ) | console · Applications et Licensing | concept | Repris dans la console (D7) : licences des applications signées comme celles des clients, commandes et paiements par Orders et Billing ; core-api les garde jusqu'à la reprise |

## 14 Ce que le travail déjà fait devient

La console passe déjà par des **guichets du socle** au lieu d'écrire dans les tables des clients. Ces guichets restent ; seul celui qui les appelle change : la brique `Link` de l'instance, à la réception de la licence.

| Élément (lots 1 à 4) | Aujourd'hui | Lot 3 bis | Effort |
|---|---|---|---|
| Catalogue, grilles, calculateur, promotions, options | Console | Inchangé | aucun |
| Abonnements, historique, journal | Console | Inchangé | aucun |
| Guichets du socle | Appelés par la console, sur place | Appelés par la brique Link, sur l'instance | aucun |
| Réconciliateur | Calcule et applique sur place | Calcule, signe et publie la licence ; l'application revient dans le compte rendu | moyen |
| Déploiement | Exécuté par la console, dans la base du client | Exécuté par l'instance ; la console suit les étapes | moyen |
| Clients (création, liste) | Lisent et écrivent `tenants` / `domains` | Registre propre de la console (profil + instance + domaine) ; `tenants` / `domains` restent sur l'instance | moyen |
| Synchronisation du catalogue | Modules du code de la console | Modules déclarés par les instances, avec leur version | petit |
| Test de bout en bout (phase 1) | Un seul processus | Console et instance simulées, reliées par un faux transport | moyen |

# 15 Les décisions

| # | Sujet | Décision |
|---|---|---|
| D1 | Où construire | Dans la nouvelle API, `platform/`, base centrale du serveur NetByUs. Rien n'est construit dans core-api |
| D2 | Réutilisation | Recopier les algorithmes (devis, passerelles, PDF, promotions), pas le code ni les tables |
| D3 | Noms des tables | Préfixe par domaine ; seules `tenants`, `domains`, `tenant_modules`, `tenant_states` gardent leur nom |
| D4 | Identifiant du client | Chaîne lisible (ex. `sdp`), unique sur toute la plateforme |
| D5 | Montants | Entiers en F CFA ; prix figés à la commande ; grille versionnée |
| D6 | Avant l'accès à la console | Routes derrière un garde console ; en local et en test, un agent de développement ; fermé sur un serveur |
| D7 | Applications externes | **décidé** à terme, tout est dans la console : la vente des applications externes (Shop, ECU, Influencer), leurs licences et les distributeurs y sont repris ; core-api les garde seulement jusqu'à leur reprise (lot 9). Comme logiciels qui appellent l'API, ils sont enregistrés dans Applications |
| D8 | Modèle de prix | **à confirmer** prix par module et par utilisateur ; formules facultatives |
| D9 | Durées du cycle de vie | **à confirmer** essai, grâce, conservation en lecture seule |
| D10 | Clients actuels | **à confirmer** un abonnement avec les modules qu'ils utilisent, au prix et à la durée décidés |
| D11 | Inscription publique | **à confirmer** une page publique peut-elle créer un client (essai) ? |
| D12 | Clé de signature | **décidé** une seule paire pour la plateforme, avec rotation (voir D22) |
| D13 | Identité des agents | **décidé** méthode jumelée : LDAP ou compte du client `netbyus` |
| D14 | Droits sur la console | **décidé** inscription explicite dans `staff_users`, jamais par un administrateur de `netbyus` |
| D15 | Double authentification | **décidé** obligatoire pour tous les agents |
| D16 | Sens de la communication | **décidé** l'instance vient chercher sa licence ; la console ne se connecte jamais aux bases d'une instance |
| D17 | Webhooks | **décidé** console → instance : signal sans données ; instance → console : comptes rendus. Signés, journalisés, rejoués ; l'interrogation régulière reste le filet de sécurité |
| D18 | Fonctionnement sans contact | **décidé** jusqu'à la date de fin de licence (fin de la période payée), puis grâce, puis lecture seule ; renouvellement par le lien ou par fichier de licence importé |
| D19 | Serveurs dédiés | **décidé** installés et mis à jour par NetByUs **ou** par le client ; seule une licence signée par NetByUs active des modules |
| D20 | Nombre de consoles | **décidé** une seule console, sur le serveur NetByUs |
| D21 | Protection du code | **décidé** pas d'encodage (pas d'ionCube) : signature, vérification en plusieurs endroits, intégrité du code, comptes rendus, contrat |
| D22 | Clés de licence | **décidé** Ed25519 ; clé privée uniquement sur le serveur NetByUs (chiffrée, sauvegardée hors ligne) ; clé publique dans chaque instance ; rotation sans réinstaller |
| D23 | Versions des serveurs dédiés | **décidé** toutes les versions restent supportées, même sans mise à jour ; le protocole du lien n'évolue que par ajouts ; un module de la licence absent du code est signalé et s'active à la mise à jour |

# 16 Le plan de construction

| Lot | Contenu | Terminé quand |
|---|---|---|
| **1 · Fondations** fait | Chargement de `platform/*/manifest.php` sur le serveur NetByUs ; Deptrac : le socle n'importe pas `Platform\` ; garde console et agent de développement ; acteur « agent NetByUs » ; journal `aud_logs` | Un domaine vide est chargé et testé ; une action est journalisée avec son acteur |
| **2 · Clients** fait | Profils, contacts, étapes de déploiement rejouables ; premier administrateur ; routes `tenants` | Un client est créé de bout en bout ; une étape en échec se relance seule |
| **3 · Catalogue, abonnements, activation** fait | Catalogue synchronisé depuis les manifestes ; abonnements et historique ; réconciliateur, `act_runs`, `origin`, état publié versionné ; passage de nuit (file « heavy ») ; quota d'utilisateurs appliqué par Auth | Le critère du chapitre 1, dans un seul processus |
| **3 bis · Instances, lien et licences** à faire | Console : domaines `Instances` (registre, jeton d'enrôlement, santé, version, modules présents, webhooks sortants, comptes rendus) et `Licensing` (clé Ed25519, licences signées et versionnées, fichier de licence).<br>Socle : brique `Link` (enrôlement, interrogation, vérification, application par les guichets, comptes rendus, webhook, fin de licence, horloge, intégrité du code).<br>Reprise des lots 2 et 3 (chapitre 14). | Avec deux applications (console et instance SaaS) : un client est déployé sur l'instance ; un module ajouté y est actif en quelques secondes par webhook, en 5 minutes sans ; une instance éteinte rattrape ; une licence modifiée ou non signée est refusée ; un fichier de licence active un serveur coupé de la console |
| **4 · Calculateur, grilles, promotions** en grande partie fait | Grilles, prix, options, promotions, calculateur en entiers ; reste le prorata | Le même devis sort pour la console, « Mon abonnement » et le paiement |
| **5 · Commandes et paiement** en partie fait | Commandes, FedaPay et KkiaPay, webhooks dédoublonnés, factures et PDF ; guichets de « Mon abonnement » | Un client ajoute un module, paie, et le module s'active seul, sur son instance |
| **6 · Cycle de vie** | Tâche quotidienne (essai, grâce, lecture seule, archivage) ; middleware de lecture seule du socle, piloté par la licence | Un abonnement non renouvelé passe en grâce puis en lecture seule, même sur un serveur coupé de la console |
| **7 · Exploitation** | Migrations, sauvegardes, place utilisée et archivage, par instance | Une migration générale montre les instances et les clients en échec |
| **8 · Support, applications, annonces** | Tickets et accès support ; applications externes (registre, consentement, autorisations dans la licence) ; annonces | Une application n'atteint que les clients qui l'ont autorisée |
| **9 · Reprise des clients actuels** | Profil, abonnement et données de chaque client de core-api ; reprise de la vente des applications externes (Shop, ECU, Influencer), de leurs licences et des distributeurs (D7) | Le dernier client tourne sur la nouvelle API ; `netbyus` est un client ordinaire ; core-api ne vend plus rien |
| **10 · Staff et accès à la console** | Comptes, rôles, connexion jumelée, double authentification (chapitre 17) | Un agent inscrit se connecte ; un compte non inscrit est refusé |
| **11 · La console en package** | Sortir `platform/` dans le package privé `netbyus/byplus-console` (son `composer.json`, son ServiceProvider découvert automatiquement) ; commande `byplus:console:install` : vérifier qu'on est sur le serveur NetByUs, publier la configuration, jouer les migrations de la console, créer la clé de signature si elle manque, inscrire le premier super-administrateur, enregistrer l'instance `netbyus`, planifier les tâches ; mise à jour par `composer update` puis `byplus:console:install` (rejouable) ; documentation d'installation | Sur un serveur BY+ vierge, deux commandes installent la console ; le serveur SaaS et un serveur dédié n'en ont pas le code ; tous les tests de la console passent depuis le package |

**Prochaine étape :** le lot 3 bis, avant la vente : sans lui, rien de ce qui est vendu ne peut atteindre un client qui n'est pas sur le serveur de la console.

# 17 L'accès à la console

L'authentification est branchée après les lots 1 à 3, mais elle conditionne la sécurité de toute la plateforme : la console agit sur tous les clients et signe leurs licences. Deux approches ont été étudiées ; elles sont retenues ensemble, encadrées par trois conditions.

| | A · LDAP (annuaire de NetByUs) | B · Utilisateurs du client netbyus |
|---|---|---|
| **Principe** | L'agent saisit son identifiant d'annuaire ; l'API vérifie auprès du serveur LDAP, puis délivre un jeton | L'agent se connecte avec son compte BY+ du client `netbyus` ; certains comptes sont autorisés dans la console |
| **Avantages** | Une seule identité pour tout NetByUs ; un départ est coupé partout en un geste | Rien de nouveau à installer ; mêmes écrans que BY+ |
| **Risques** | Le serveur LDAP doit être joignable et disponible ; pas de double authentification native | Qui prend la main sur `netbyus` prendrait la main sur tous les clients, sans les conditions ci-dessous |
| **Retenu** | oui · pilote `ldap` | oui · pilote `netbyus`, comme preuve d'identité seulement |

## Les trois conditions (toutes obligatoires)

1. L'agent est inscrit explicitement dans `staff_users`, actif, avec son rôle. Un compte LDAP ou `netbyus` valide ne suffit pas.
2. Un administrateur du client `netbyus` ne peut pas s'y ajouter lui-même : l'inscription se fait uniquement depuis la console, par un super-administrateur, et elle est journalisée.
3. La double authentification est obligatoire pour tous les agents, même si l'identité vient de LDAP ou de `netbyus`.

## Le parcours de connexion

> **Depuis BY+ netbyus (le parcours ordinaire).** L'agent est déjà connecté à BY+ du client netbyus : sa session prouve son identité (étape 2, pilote netbyus). En ouvrant « Console NetByUs », le front envoie son jeton netbyus à `POST /api/v1/console-auth/netbyus` ; l'API vérifie l'inscription (étape 3), puis envoie un code à l'e-mail de l'agent inscrit et le demande (étape 4) et délivre le jeton de la console (étape 5). Le parcours par identifiant et mot de passe ( `/console-auth/login` , LDAP ou netbyus) reste disponible.

| # | Étape | Ce qui se passe |
|---|---|---|
| 1 | Identifiant et mot de passe | Sur le domaine de la console |
| 2 | Preuve d'identité | Pilote `ldap`, ou pilote `netbyus` (guichet de la brique Auth, qui peut elle-même s'appuyer sur LDAP) |
| 3 | Condition 1 | Ligne active de `staff_users` (même fournisseur, même identifiant externe) ; sinon refus |
| 4 | Condition 3 | Code de double authentification vérifié |
| 5 | Jeton | Jeton à durée courte, valable uniquement pour `/api/v1/console/…` ; acteur « agent NetByUs » |
| 6 | Trace | Connexion réussie ou refusée journalisée ( `aud_logs` ) |

| Élément | Contenu |
|---|---|
| `staff_users` , `staff_roles` | email, nom, fournisseur, identifiant externe, actif, rôle (le code de double authentification part à l'e-mail inscrit, haché dans `staff_challenges` ). Les droits : ceux du rôle ( `staff_roles` : nom fr/en et liste de droits, modifiables dans « Rôles » ; super-administrateur verrouillé, il a tout), plus ceux ajoutés à l'agent, moins ceux retirés (« Droits de l'agent »). Catalogue `ConsoleRights` : « {domaine}.view » pour chaque domaine qui a des écrans, et les droits fins déclarés par les domaines ( `support.tickets.all` …). Rôles livrés : super-administrateur, commercial, support, finance, exploitation |
| Pilotes d'identité | `ConsoleIdentityProvider` : LDAP, et `netbyus` par le guichet de la brique Auth (jamais une lecture directe de la base du client) |
| Compte de secours | Un compte local (mot de passe et code par e-mail), à usage exceptionnel et tracé |
| Révocation | Compte désactivé ou ligne `staff_users` inactive : la connexion échoue et les jetons sont révoqués |

# 18 Le suivi des tâches

La liste de référence de ce qui est fait et de ce qui reste, **dans l'ordre d'exécution**. Règle : quand une tâche est terminée, on met à jour cette page dans la même livraison (statut, date, qui) ; une nouvelle tâche s'insère à sa place dans l'ordre, jamais en fin de liste par défaut.

## Faites

| # | Tâche | Lot | Date | Qui |
|---|---|---|---|---|
| T01 | Chargement des domaines `platform/*/manifest.php`, routes `/api/v1/console/{clé}`, Deptrac (le socle n'importe pas `Platform\`) | 1 | 2 oct. 2026 | Chacour |
| T02 | Garde console, agent de développement (local et tests seulement), acteur « agent NetByUs » | 1 | 2 oct. 2026 | Chacour |
| T03 | Journal de la console (`aud_logs`), acteur obligatoire | 1 | 2 oct. 2026 | Chacour |
| T04 | Clients : profils, contacts, étapes de déploiement rejouables, premier administrateur ; colonnes `status`, `deployment`, `is_primary` | 2 | 2 oct. 2026 | Chacour |
| T05 | Catalogue synchronisé depuis les manifestes, fiches publiées | 3 | 2 oct. 2026 | Chacour |
| T06 | Abonnements, modules et options souscrits, historique | 3 | 2 oct. 2026 | Chacour |
| T07 | Réconciliateur, `act_runs`, colonne `origin`, état publié versionné (`tenant_states`), passage de nuit | 3 | 2 oct. 2026 | Chacour |
| T08 | Quota d'utilisateurs appliqué par Auth | 3 | 2 oct. 2026 | Chacour |
| T09 | Grilles de prix versionnées, prix, options, promotions, calculateur en entiers, devis | 4 | 3 oct. 2026 | Chacour |
| T10 | Fusion dans `new-develop` : chemins et espaces de noms, classes rangées dans `Services/`, configuration phpstan et phpunit, planificateur des modules | — | 3 oct. 2026 | Claude |
| T11 | Console alignée sur la structure commune ; traductions par domaine ; libellés des énumérations dans les traductions | — | 3 oct. 2026 | Claude |
| T12 | Passage de nuit en file « heavy » (un job par client, 03:00) | 3 | 3 oct. 2026 | Claude |
| T13 | Décisions D7 et D16 à D23 (instances, webhooks, licences signées, versions) ; document unique de la console | — | 3 oct. 2026 | Claude |
| T31 | Commandes (`ord_orders`, lignes, devis figé, expiration), traitement d'une commande, mails de commande | 5 | 3 oct. 2026 | Chacour |
| T32 | Paiement en ligne FedaPay et KkiaPay : montants en entiers, clés chiffrées, webhooks signés vérifiés, transactions uniques (`bill_payments`, `bill_gateway_transactions`, `bill_gateway_configs`) | 5 | 3 oct. 2026 | Chacour |
| T33 | Licences commerciales sur le modèle de core-api (émission, clé, démarrage, expiration, renouvellement) ; réglages de la console (`Settings`) ; page de commande publique (`/api/v1/store`) | 5 | 3 oct. 2026 | Chacour |
| T34 | Relecture du lot 5 ; tests indépendants du `.env` local (adresses d'invitation et de réinitialisation fixées dans `phpunit.xml`) | — | 3 oct. 2026 | Claude |
| T36 | Une seule mise en page des mails (constructeur `MailContent` de Notifications) ; mails de commande chiffrés, en file « notifications » ; NetByUs seulement dans « Powered by NetByUs » | 5 | 3 oct. 2026 | Claude |
| T14 | Domaine `platform/Licensing` : clés Ed25519 (privée chiffrée, `lic_signing_keys`), `byplus:license:keys --rotate`, licences signées et versionnées (`lic_signed_licenses`) | 3 bis | 3 oct. 2026 | Claude |
| T16 | Brique `core/Licensing` (vérifier, garder, appliquer ; accès selon les dates ; horloge reculée ; « Ma licence » et import de fichier) et brique `core/Link` (`byplus:link:enroll`, `byplus:link:sync` toutes les 5 min, identité `link_identity`) | 3 bis | 3 oct. 2026 | Claude |

| # | Tâche | Lot | Date | Qui |
|---|---|---|---|---|
| T15 | Domaine `platform/Instances` : registre, jeton d'enrôlement à usage unique, `link/enroll`, `link/state`, `link/events`, requêtes signées sans rejeu ; une instance ne rend compte que pour ses clients | 3 bis | 3 oct. 2026 | Claude |
| T17 | Webhook `state.changed` signé par NetByUs (journal `inst_webhook_deliveries`, 5 essais) reçu sur `/api/v1/link/webhook` ; comptes rendus `state.applied`, `instance.report`, `error` ; test de bout en bout console ↔ instance ( `provisioning.*` passe en T19) | 3 bis | 3 oct. 2026 | Claude |
| T18 | Le réconciliateur signe et publie la licence ; livraison sur place pour le serveur NetByUs, par le lien sinon ; statut `pending` dans `act_runs` jusqu'au compte rendu | 3 bis | 3 oct. 2026 | Claude |
| T37 | Structure commune : `OrderRules`, `OrderText`, `TenantRules` dans `Services/` ; énumérations dans `Public/Data/` ; libellés des droits en traductions (avec T40) | 5 | 3-4 oct. 2026 | Claude |
| T35 | La console ne sert plus rien sur le domaine d'un client : « Mon abonnement » est la brique `core/Subscription` ( `/api/v1/subscription/…` ), qui transmet à la console par le guichet `/api/v1/link/desk/{client}/…` (signé par l'instance ; sur place sur le serveur NetByUs) | 5 | 4 oct. 2026 | Claude |
| T40 | Droits `subscription.view` et `subscription.purchase` déclarés par la brique du socle (libellés traduits) : ils existent sur tous les serveurs | 5 | 4 oct. 2026 | Claude |
| T39 | Page de commande publique fermée tant que D11 n'est pas tranchée ( `BYPLUS_PUBLIC_SIGNUP=false` ) ; passerelles et webhooks de paiement restent ouverts pour les compléments des clients | 5 | 4 oct. 2026 | Claude |
| T19 | Registre des clients de la console ( `tenant_profiles` : état, domaine, déploiement d'un client distant) ; création d'un client sur une instance ( `instance_id` ) : l'abonnement par la console, les autres étapes exécutées par l'instance ( `core/Link` `ClientSetup` ) et rapportées ( `provisioning.step` ), relance depuis la console | 3 bis | 4 oct. 2026 | Claude |
| T20 | Catalogue des modules connus : code de la console + manifestes déclarés par les instances (enrôlement, comptes rendus) ; fiche créée (non publiée) dès qu'une instance déclare un nouveau module ; le réconciliateur résout les requis sur cet ensemble | 3 bis | 4 oct. 2026 | Claude |
| T22 | Tests de bout en bout console + instance (licence par webhook et par interrogation, « Mon abonnement » par le lien, déploiement distant, signatures, rejeu, cloisonnement) ; enrôlement documenté (chapitre 7) | 3 bis | 4 oct. 2026 | Claude |
| T21 | Empreinte du code de chaque serveur ( `byplus:code:fingerprint` ), rapportée à la console et comparée à la version publiée ( `byplus:release:register` ) ; un code modifié est journalisé et signalé | 3 bis | 4 oct. 2026 | Claude |
| T41 | Front : « Mon abonnement BY+ » dans les Paramètres (abonnement, ajouts et panier avec devis, paiement KkiaPay / FedaPay / virement, demandes, factures, clés de licence, licence du serveur et import de fichier) ; vérifié à 375 px | 5 | 4 oct. 2026 | Claude |
| T23 | Prorata au calculateur (déjà fait avec le lot 4 : `PricingTest` ) ; vérifié | 4 | 3 oct. 2026 | Chacour |
| T24 | Factures : une par demande payée, numérotée sans trou ( `FAC-2026-00001` ), vendeur et acheteur figés, taxe comprise réglable ; PDF (dompdf) ; console, guichet du client et « Mes factures » ; client d'une commande créé sur une instance ( `instance_id` ) | 5 | 4 oct. 2026 | Claude |
| T25 | Cycle de vie chaque nuit ( `byplus:subscriptions:lifecycle` ) : rappels avant l'échéance, grâce, lecture seule, archivage ; lecture seule appliquée par chaque serveur d'après sa licence (réponse 423, sauf connexion, abonnement et licence) ; bandeaux de grâce et de lecture seule | 6 | 4 oct. 2026 | Claude |
| T26 | Exploitation par instance : migration générale, sauvegarde (pg_dump), archivage, demandés par lot à tous les serveurs ; résultat par serveur ; place utilisée par client rapportée chaque heure | 7 | 4 oct. 2026 | Claude |
| T27 | Applications externes : registre ( `platform/Applications` ), consentement du client (brique `core/Integrations`, écran « Intégrations »), droits accordés et clé publique dans la licence signée, requêtes signées de l'application ; annonces ( `platform/Announcements` ) diffusées à tous les serveurs et affichées dans BY+. Le support est laissé de côté | 8 | 4 oct. 2026 | Claude |

| # | Tâche | Lot | Date | Qui |
|---|---|---|---|---|
| T29 | Agents de la console ( `platform/Staff` ) : inscription par un super-administrateur, preuve d'identité LDAP NetByUs ou compte netbyus, double authentification obligatoire (code par e-mail, voir T43), jeton court réservé à la console, droits par rôle et par domaine, tout journalisé ; `byplus:staff:super-admin` | 10 | 4 oct. 2026 | Claude |
| T38 | Écrans de la console (front) : connexion avec double authentification, tableau de bord, clients (liste, création, fiche : déploiement, abonnement, activation, serveur, fichier de licence), demandes et paiements, factures, licences, catalogue, grilles de prix, applications, serveurs, exploitation, annonces, journal, agents, paiement en ligne, réglages ; menu selon le rôle ; vérifiés à 375 px | — | 4 oct. 2026 | Claude |
| T28 | Reprise des clients de core-api : `byplus:import:core-api --dry-run` puis réelle, rejouable (profil, sociétés, utilisateurs avec leur mot de passe et administrateurs, modules par correspondance dans `config/byplus-import.php` ) ; la vente des applications externes (D7) est reprise par `byplus:import:console` : applications, catalogue, un client par contact titulaire, instances, commandes, paiements et factures, licences avec leur clé. Reste à faire en exploitation : lancer la reprise sur les vraies données, remplir la correspondance des modules, puis arrêter la vente dans core-api | 9 | 4 oct. 2026 | Claude |
| T30 | Console en package `netbyus/byplus-console` ( `packages/byplus-console` ) et `byplus:console:install` : refuse une instance enrôlée, migrations, clé de signature, catalogue, premier super-administrateur ; rejouable ; un serveur sans le package n'a aucune route de console | 11 | 4 oct. 2026 | Claude |
| T43 | Console dans le shell de BY+ netbyus : package front `@netbyus/byplus-console-app` (inclus seulement si `VITE_WITH_CONSOLE=1` , chargé seulement pour le client netbyus et un agent inscrit) ; entrée par la session netbyus et un code envoyé à l'e-mail de l'agent inscrit ( `/console-auth/netbyus` , `/verify` , `/resend` ) ; `byplus:console:install` crée le client netbyus (identifiant réservé) | 11 | 4 oct. 2026 | Claude |
| T44 | Toutes les applications gérées de la même façon, BY+ compris : une fiche par application (vue d'ensemble, catalogue, offres standard et de stockage, événements de webhook, instances, licences) ; une instance par client avec ses clés (signature des appels, clés publiques dans la licence) et ses webhooks signés et testables ; fiche d'instance ; applications du client sur sa fiche ; licences filtrées par application ; menu revu (Produits, Ventes, Plateforme, Administration) ; anciens écrans Catalogue et Grilles de prix repris dans la fiche BY+ | 11 | 4 oct. 2026 | Claude |
| T45 | Les offres font la licence : offres fixes de NetByUs par application (durée et prix par tarif, modules avec toutes ou une partie de leurs permissions, utilisateurs, options chiffrées) à côté des prix à la carte avec lesquels le client compose la sienne ; une demande (nouveau client, renouvellement) ou une licence s'émet depuis une offre, pour toute application ; la licence et l'abonnement gardent l'offre et ses permissions, la licence signée porte permissions et quotas ( `LicenseState::permissions` ) ; une licence d'une autre application ne touche pas l'abonnement BY+ et prévient son instance ; événements de webhook dans le code, `licence_updated` envoyé aux instances du client. Reste : appliquer les permissions et les quotas d'options dans BY+, et proposer les offres fixes dans l'écran d'abonnement du client | 11 | 4 oct. 2026 | Claude |
| T46 | Licence composée et modifiable : un agent crée une licence directement pour un client (application, offre de départ facultative, puis modules, permissions, utilisateurs, options et durée ajustés) et modifie les caractéristiques d'une licence existante (contenu et fin) ; une licence en cours applique le changement tout de suite (abonnement BY+ et licence signée, ou webhook de l'instance). Onglet « Clés API » sur chaque application (clés de tous ses clients). Les prix des modules et options rejoignent l'onglet Offres. Console redessinée : menu, en-tête, tableau de bord, fiche client | 11 | 4 oct. 2026 | Claude |
| T42 | Support, API (lots S1 à S7 de `support-et-widgets.pdf` ) : domaine `platform/Support` (tickets de toutes les applications rattachés à leur instance, réglages par application, temps réel, Mattermost, webhooks, statistiques, widget à identité signée, requêtes signées des applications et délégués, appels vidéo, reprise de core-api) ; brique `core/Helpdesk` (guichet par le lien, accès support en lecture seule accordé par le client) ; brique `core/LiveKit` | 8 | 17-19 oct. 2026 | Chacour, Claude |
| T50 | Catalogue aligné sur le code : la fiche d'un module de BY+ disparu du code (ou devenu non vendable) est supprimée si rien ne l'utilise, sinon seulement retirée de la vente. Chaque domaine dit ce qu'il utilise par le guichet `Platform\Catalog\Public\Contracts\ModuleUsage` (étiquette) : prix, options et offres (Catalog), abonnements (Subscriptions), licences commerciales (Licensing), demandes de démo et essais (Leads) | 3 | 19 oct. 2026 | Claude |

| # | Tâche | Lot | Date | Qui |
|---|---|---|---|---|
| T51 | Support, écrans des agents (partie de T47) : menu de l'icône « Support » (tableau de bord avec les statistiques filtrées par période et application, « Les tickets », « Mes assignés », « Favoris » des tickets épinglés depuis leur fiche, « Paramètres › Catégories » avec création, modification et suppression) ; table `helpdesk_ticket_pins` et adresses `/support/pins`, `/support/tickets/{ticket}/pin`, filtre `assigned_to_user_id=me` ; une entrée de menu peut porter des paramètres de route (socle) | 8 | 23 oct. 2026 | Claude |
| T52 | Droits de la console : catalogue `ConsoleRights` (accès « {domaine}.view » à chaque domaine qui a des écrans, droits fins déclarés par les domaines), rôles modifiables ( `staff_roles` , les cinq rôles repris tels quels ; super-administrateur verrouillé) et droits ajoutés ou retirés par agent ( `staff_users.rights_granted` , `rights_revoked` ) ; contrôle par route ( `console.right:…` ) ; écrans « Rôles » et « Droits de l'agent » ; support : `support.view` (ses tickets), `support.tickets.all` , `support.categories.manage` , `support.settings.manage` (partie de T48) | 10 | 23 oct. 2026 | Claude |
| T53 | Statistiques du support en widgets du tableau de bord de BY+ ( `support.*` , pour les agents qui voient tous les tickets, chez le client netbyus), ajoutés depuis la console (« Ajouter au tableau de bord », « Tout ajouter ») ; images et PDF lus dans la discussion (album, visionneuse, aperçu) ; lecteur vocal (onde réelle, curseur, vitesse) ; enregistrement vocal retiré ; épingles depuis la file ; LiveKit local | 8 | 24 oct. 2026 | Claude |

## À faire, dans l'ordre

| # | Tâche | Lot | Terminé quand |
|---|---|---|---|
| T47 | Support, écrans : agents dans la console (reste : onglets « Support » des fiches application et client, réglages par application, clés du widget, entrée chez le client ; fait : voir T51), écran « Support » de BY+, `support-widget.js` , page d'appel `/support/call/{call}` | 8 | Un ticket est suivi de bout en bout depuis les écrans |
| T48 | Support : agent limité à certaines applications (décision S6) ; les droits fins sont faits (T52) | 8 | Un agent limité à Shop ne voit pas les tickets d'ECU |
| T49 | Support : pièces jointes par la brique Files (stockage S3, liens temporaires) | 8 | Une pièce jointe sort par un lien temporaire |

## Décisions qui attendent une réponse

| # | Sujet | Nécessaire avant |
|---|---|---|
| D8 | Modèle de prix (par module, par utilisateur, formules) | T24 (vente) |
| D9 | Durées de l'essai, de la grâce et de la conservation en lecture seule | T21 et T25 |
| D10 | Reprise des clients actuels (prix, durée) | T28 |
| D11 | Inscription publique (essai sans NetByUs) | T24 |