BY+ · DOCUMENTATION MODULE

Module Contacts — document de définition

Version de travail du 5 octobre 2026.

Statut : version de travail, à valider en atelier, puis à suivre pour `php artisan byplus:make-module contacts`. Dans le § 8, les décisions marquées « Décidé » sont prises ; celles marquées « Proposé » attendent l'atelier.

Références : `docs/catalogue-modules.pdf` (§ 4 : module Contacts, famille Relation client ; § 5 : « Contacts : un module, pas le socle » ; § 6 : ponts vers Contacts), `docs/architecture.pdf` (règles des tables, 10 règles d'or), `docs/socle-mise-en-place.pdf`, document de définition du module POS (modèle de ce document).

Ancien code de référence : `app/v1/Contact/` (namespace `Core\v1\Contact`), sans classe Service : la logique est dans les contrôleurs et le modèle `Contact`. Il sert de référence métier, pas de base à copier.

En bref

- Un module, clé `contacts`, vendable seul (carnet d'adresses), qui ne requiert aucun module.
- Activé automatiquement par Facturation et Pipeline (et, dans le catalogue, par Rendez-vous, Appels d'offres, NPS, Portail client, Event, Fidélité, Entrepôt, Dossiers notariaux).
- Six ensembles fonctionnels : fiches, adresses, interlocuteurs d'une société, professions, import et export, statistiques.
- Trois tables préfixées `contacts_`, en UUID, dans la base de chaque client.
- Sept droits, vérifiés sur chaque adresse de l'API, servie sous `/api/v1/contacts`.
- Hors du module : la connexion des contacts (Portail client), les factures et la devise du client (Facturation), les opportunités (Pipeline), les pays et villes (socle).

1. Identité (prête pour `module.php`)

```
return [
    'key'         => 'contacts',
    'name'        => 'Contacts',
    'version'     => '0.1.0',
    'requires'    => [],
    'sellable'    => true,
    'permissions' => [
        'contacts.view', 'contacts.create', 'contacts.update',
'contacts.archive',
        'contacts.import', 'contacts.export',
        'contacts.professions.manage',
    ],
    'settings'    => [],
    'schedule'    => [],
    'provider'    => \Modules\Contacts\ContactsServiceProvider::class,
];
```

Test des 4 questions : un client s'en sert seul comme carnet d'adresses ✓ · menu et écrans propres (Contacts, fiche contact, Professions) ✓ · droits propres ✓ · retirable quand aucun module qui le requiert n'est actif ; sinon le système refuse, ce qui est normal ✓ → module vendable.

Droits. Seule la clé d'un droit est en anglais ; son libellé est traduit dans `modules/Contacts/lang/fr.json` et `en.json` (`contacts.archive` → « Archiver un contact »). L'ancien code en compte huit dans `ContactPermissionEnum`, plus ceux de `ProfessionPermissionEnum`. `contact_delete` devient `contacts.archive` (on n'efface plus un contact, § 8.1). `contact_invoice_view` et `contact_estimate_view` quittent le module : ce sont des droits de Facturation.

2. Ce que le module comporte

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

Ensemble	Contenu
Fiches	Créer, modifier, archiver et réactiver un contact. Type : particulier, société ou influenceur. Rôles cumulables : client, fournisseur, sous-traitant, partenaire, prospect. Identité, poste, date de naissance, emails supplémentaires, plusieurs téléphones, site web, avatar. Liste, recherche, filtres par rôle et par type, recherche par email.
Adresses	Plusieurs adresses par contact, dont une par défaut. Pays, ville et zone pris dans les référentiels du socle ; champs cadastraux (quartier, îlot, parcelle, maison).
Interlocuteurs d'une société	Les personnes rattachées à un contact société : les ajouter, les modifier, les retirer. (Ancien nom : « collaborateurs », à ne pas confondre avec le module RH.)
Professions	La liste des métiers, choisie sur la fiche.
Import et export	Import Excel avec modèle à télécharger, export Excel.
Statistiques	Nombre de contacts, répartition par type et par rôle.

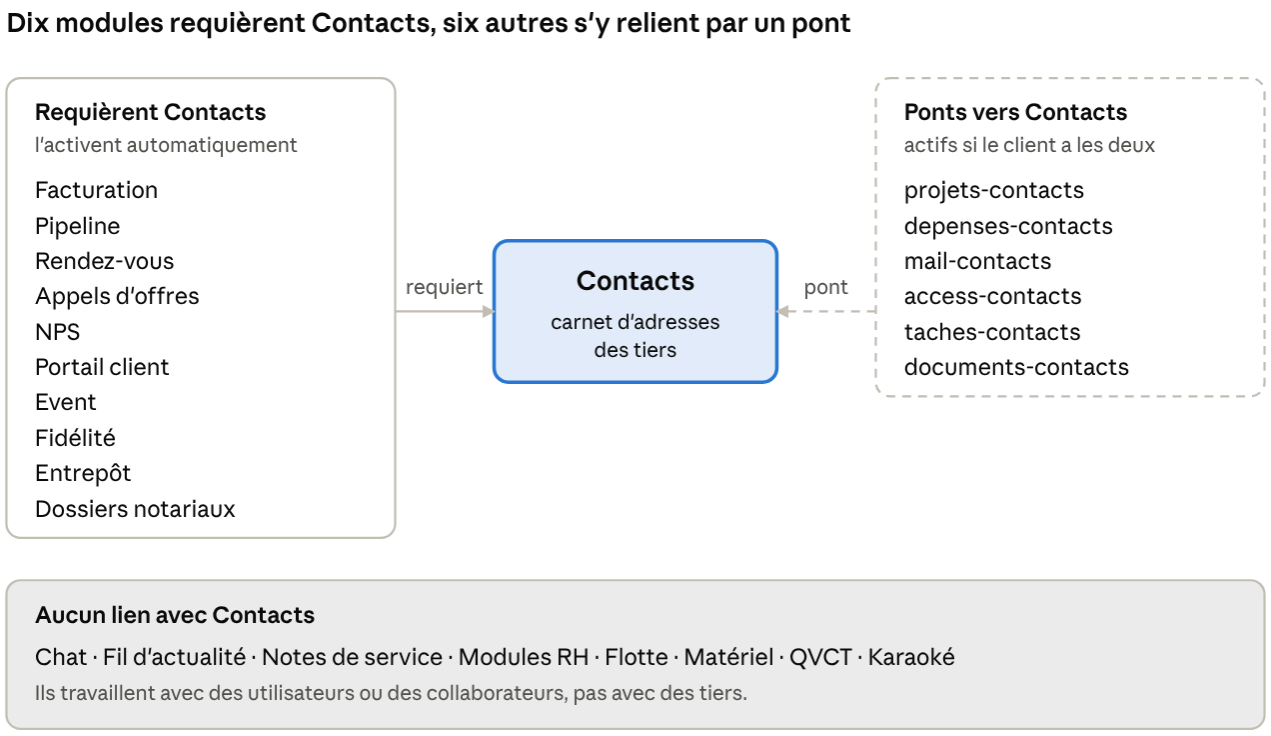
Contact, utilisateur et collaborateur sont trois choses différentes : un **utilisateur** se connecte à BY+ (socle), un **contact** est un tiers extérieur (ce module), un **collaborateur** est un salarié (module Collaborateurs).

3. Hors périmètre (renvois explicites)

Besoin	Où il vit
Connexion d'un contact, invitation, mot de passe, validation ou rejet d'un compte client, applications externes (`contact_applications`, `auth.contact`)	Module Portail client (§ 8.2)
Factures et devis d'un contact, statistiques de facturation, meilleurs clients	Module Facturation, qui ajoute l'onglet « Factures » sur la fiche
Devise et taux de change d'un client	Module Facturation (§ 8.3)

Besoin	Où il vit
Opportunités d'un contact	Module Pipeline, qui ajoute l'onglet « Opportunités » sur la fiche
Projets d'un contact, statut du contact dans un projet	Pont `projets-contacts`
Achats en caisse d'un contact	Pont `pos-contacts` (document POS, § 8.1)
Pays, villes, zones	Référentiels du socle (§ 8.4)
Société et agence de rattachement	Brique Companies du socle
Avatar	Brique Files du socle

Vue d'ensemble des modules liés à Contacts :



Contacts et les autres modules · requis, ponts, aucun lien

4. Entités et tables (base de chaque client)

Règles appliquées à toutes les tables : préfixe `contacts_` ; clé primaire UUID (`uuid('id')->primary()` et trait `HasUuids`) ; clés étrangères seulement vers le socle ; pays par son code (brique Countries).

Modèle (`Modules\Contacts\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Contact`	`contacts_contacts` (`contacts`)	`type` (`particular`, `society`, `influencer`), `gender`, `lastname`, `firstname`, `birthday`, `job_title`, `status`, `email`, `additional_emails` (JSON), `phones` (JSON), `website`, `is_customer`, `is_supplier`, `is_subcontractor`, `is_partner`, `is_prospect`, `organization_id` (contact société de l'interlocuteur), `profession_id`, `company_id`, `office_id`, `archived_at`. **Change** : `company_id` désigne désormais la société du socle et remplace `society_id` ; l'ancien `company_id` (lien vers le contact société) devient `organization_id` ; `contactType` devient `type` ; avatar par `HasAttachments` ; plus de `Authenticatable` ni de colonnes de connexion (§ 8.2) ; plus de `currency_id` ni `exchange_rate` (§ 8.3) ; archivage par `archived_at` (§ 8.1).

Modèle (`Modules\Contacts\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Address`	`contacts_addresses` (`addresses`)	`contact_id`, `address`, `address_complement`, `phone`, `zipcode`, `is_default`, `country_code`, `city_id`, `area_id`, `district`, `block`, `square_number`, `plot`, `house`. **Change** : `country_code` remplace `country_id` ; `addressComp` devient `address_complement`.
`Profession`	`contacts_professions` (`professions`)	`name`, `company_id`.

Anciennes tables non reprises : `project_contacts` et `contact_statuses` (pont `projets-contacts`) ; `contact_applications` et `contact_application_api_keys` (Portail client).

Règles métier reprises :

- La création d'un contact, de son adresse par défaut et de ses interlocuteurs se fait en une seule transaction.
- Un contact a au plus une adresse par défaut.
- Les rôles se cumulent : une société peut être à la fois cliente et fournisseur.
- Un interlocuteur est rattaché à un contact société par `organization_id`.
- On n'efface pas un contact, on l'archive : il disparaît des listes et des choix, mais reste visible dans les documents qui le citent.

Anomalies de l'ancien code, à ne pas reproduire : la route `DELETE contact/{contact}/delete/collaborators` est déclarée deux fois (`routes/api.php:38-39`) ; `ContactResource` expose des colonnes passées dans `addresses` et une relation `department` qui n'existe pas ; `getFirstAddressInfosAttribute` teste `addressCom` au lieu de `addressComp` (`Contact.php:126-128`) ; le droit `contact_delete` existe sans route.

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

```
modules/Contacts/Public/
├── Events/ContactCreated.php      # un contact est créé
├── Events/ContactUpdated.php      # sa fiche ou son adresse par défaut
change
├── Events/ContactArchived.php     # il est archivé
└── Contracts/ContactDirectory.php # trouver, chercher, lister par rôle,
photo d'un contact
```

`ContactDirectory` renvoie des objets de données, jamais des modèles : aucun `use Modules\Contacts\Models\…` hors du module. Il fournit aussi la **photo** d'un contact (nom, adresse par défaut, identifiants), que Facturation enregistre sur un devis ou une facture.

Facturation et Pipeline requièrent Contacts : ils utilisent `ContactDirectory` et écoutent ses événements directement, sans pont. Les modules facultatifs (Projets, POS…) passent par un pont, avec un écouteur qui étend `Core\Modules\Public\ModuleListener`.

Pas de point d'extension en 0.1.0 : il ne se crée que pour un besoin réel d'un client.

6. Droits, écrans, routes

Écrans : Contacts (liste) · Fiche contact · Professions · Import, plus un onglet ajouté par chaque module ou pont actif (Factures, Opportunités, Projets, Achats).

Routes : `modules/Contacts/routes/tenant.php`, servies sous `/api/v1/contacts` sur le sous-domaine du client. Le socle pose trois gardes : le client (tenancy), la connexion (`RequireUser`, 401 sinon) et le module (`module:contacts`, 403 sinon). Chaque action vérifie ensuite son droit par `$this->authorize('contacts.…')`.

Périmètre : un utilisateur ne voit que les contacts des sociétés qui lui sont accessibles.

Adresse (sous `/api/v1`)	Rôle · droit
`GET·POST /contacts`, `GET·PATCH /contacts/{id}`	Fiches ; filtres `role`, `type`, `search`, `email` (remplacent les listes clients, fournisseurs, sans société, par email) · `contacts.view`, `contacts.create`, `contacts.update`
`POST /contacts/{id}/archive`, `POST /contacts/{id}/restore`	Archiver, réactiver · `contacts.archive`
`POST /contacts/{id}/avatar`	Avatar · `contacts.update`

Adresse (sous `/api/v1`)	Rôle · droit
`GET·POST /contacts/{id}/addresses`, `PATCH·DELETE …/{addressId}`, `POST …/{addressId}/default`	Adresses · `contacts.update`
`GET·POST·DELETE /contacts/{id}/people`	Interlocuteurs d'une société · `contacts.update`
`GET·POST /contacts/professions`, `PATCH·DELETE …/{id}`	Professions · `contacts.professions.manage` (lecture : `contacts.view`)
`POST /contacts/import`, `GET /contacts/import/template`, `GET /contacts/export`	Import et export · `contacts.import`, `contacts.export`
`GET /contacts/statistics`, `GET /contacts/enums`	Statistiques, listes de valeurs · `contacts.view`

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

Brique	Guichet ou trait	Usage par Contacts
Kernel	`TenantController`, `Context`	Contrôleurs, utilisateur et société de la requête, format des réponses
Auth	droits `contacts.*`	Les droits du manifeste rejoignent seuls le catalogue des droits
Companies	`CompanyDirectory`, `CurrentCompany`, trait `BelongsToCompany`	Un contact appartient à une société, éventuellement à une agence
Files	`Attachments`, trait `HasAttachments`	Avatar
Audit	`Journal`	Création, modification, archivage, import
Countries et référentiels	`Countries`, référentiel des villes et zones	Adresses

Brique	Guichet ou trait	Usage par Contacts
Modules	`ModuleListener`, les cinq portes	Routes et écouteurs refusés au client qui n'a pas le module

Dépendance à installer : `maatwebsite/excel`, déjà utilisé par l'ancien code pour l'import et l'export.

8. Décisions pour la version 0.1.0

#	Sujet	Décision	État	Pourquoi
1	Suppression	On n'efface pas un contact, on l'archive ; `contacts.archive` remplace `contact_delete`.	Décidé	Un contact est cité par des factures, devis et projets.
2	Connexion des contacts	Colonnes de connexion, `Authenticatable`, invitation, validation de compte, `contact_applications` et `auth.contact` passent au Portail client.	Décidé (à confirmer en atelier pour les applications)	Le Portail fournit les comptes externes ; Contacts reste un carnet d'adresses.
3	Devise du client	`currency_id` et `exchange_rate` passent à Facturation.	Décidé	La devise ne sert qu'à facturer.
4	Pays et villes	Référentiels du socle.	Décidé	Utiles à tous les modules.
5	Onglets de la fiche	Factures et Opportunités sont ajoutées par Facturation et Pipeline ; Projets et Achats par leurs ponts.	Décidé (catalogue)	Contacts ne connaît aucun autre module.
6	Lien interlocuteur → société	`organization_id` remplace l'ancien `company_id`.	Décidé	`company_id` désigne la société du socle dans toute l'architecture.

#	Sujet	Décision	État	Pourquoi
7	Adresses de l'API	Nouvelles adresses REST sous `/api/v1/contacts` ; les anciennes ne sont pas reprises.	Proposé	Même forme que POS et tout le nouveau socle.
8	Clé du module	`contacts`.	Proposé	Cohérente avec le catalogue et ses ponts `*-contacts`.

Encore à trancher : la place de `post_id` (table `post_offices`) ; le sens de la colonne `status`, à côté de `archived_at`.

9. Ordre d'intégration

1. Générer le module : `php artisan byplus:make-module contacts`.
2. Lot 1 : fiches, rôles, archivage, recherche, vitrine `ContactDirectory` et événements. Pipeline et Facturation peuvent démarrer dès ce lot.
3. Lot 2 : adresses, interlocuteurs, professions.
4. Lot 3 : import, export, statistiques.
5. Ensuite : le Portail client reprend la connexion des contacts ; les ponts `projets-contacts` et `pos-contacts` au fil des modules voisins.
6. Reprise des données : `society_id` → `company_id`, ancien `company_id` → `organization_id`, `country_id` → `country_code`, colonnes de connexion → Portail client, devise et taux → Facturation.

Chaque lot est livré avec ses tests dans `modules/Contacts/tests`, vérifié sur une vraie base client, puis fusionné.

Terminé quand le test d'activation passe (règle 10 de l'architecture) : sur un client neuf, on active Contacts seul ; ses écrans répondent, ceux des autres modules sont refusés ; on crée une société avec son adresse par défaut et deux interlocuteurs, on l'archive, elle disparaît des listes ; un utilisateur sans `contacts.archive` ne peut pas archiver ; activer Facturation laisse Contacts actif et empêche de le retirer. PHPStan et Deptrac sont verts.

BY+ · DOCUMENTATION MODULE

Module Pipeline — document de définition

Version de travail du 5 octobre 2026.

Statut : version de travail, à valider en atelier, puis à suivre pour `php artisan byplus:make-module pipeline`. Dans le § 8, les décisions marquées « Décidé » sont prises ; celles marquées « Proposé » attendent l'atelier.

Références : `docs/catalogue-modules.pdf` (§ 4 : module Pipeline, requiert Contacts ; § 6 : ponts `pipeline-facturation`, `pipeline-rdv`, `appels-offres-pipeline`), `docs/architecture.pdf`, `docs/socle-mise-en-place.pdf`, documents de définition des modules POS et Contacts.

Ancien code de référence : module `Pipeline` de l'ancien BY+ (opportunités, étapes, activités, objectifs de vente). Il sert de référence métier, pas de base à copier.

En bref

- Un module, clé `pipeline`, vendable, qui requiert le module Contacts.
- Huit ensembles fonctionnels : opportunités, étapes et critères, conclusion, entourage d'une opportunité, activités, paramétrage, objectifs de vente, statistiques et prévisions.
- Dix-neuf tables préfixées `pipeline_`, en UUID, dans la base de chaque client.
- Neuf droits, vérifiés sur chaque adresse de l'API, servie sous `/api/v1/pipeline`.
- Hors du module : les fiches contacts (Contacts), les produits proposés, devis et factures d'une opportunité (pont `pipeline-facturation`), les comptes, fichiers et rappels (socle).

1. Identité (prête pour `module.php`)

```
return [
    'key'         => 'pipeline',
    'name'        => 'Pipeline',
    'version'     => '0.1.0',
    'requires'    => ['contacts'],
    'sellable'    => true,
    'permissions' => [
        'pipeline.opportunities.view', 'pipeline.opportunities.manage',
        'pipeline.opportunities.close', 'pipeline.opportunities.delete',
        'pipeline.activities.manage',
        'pipeline.settings.manage', 'pipeline.targets.manage',
        'pipeline.statistics.view', 'pipeline.export',
    ],
    'settings'    => [],
    'schedule'    => [], // rappels des activités : mécanisme à préciser (§
8)
    'provider'    => \Modules\Pipeline\PipelineServiceProvider::class,
];
```

Test des 4 questions : un commercial s'en sert seul, avec Contacts activé automatiquement ✓ · menu et écrans propres (Pipeline, Opportunité, Objectifs, Statistiques, Paramètres) ✓ · droits propres ✓ · retirable sans casser un autre module, le pont `pipeline-facturation` se désactive seul ✓ → module vendable.

Droits. Seule la clé d'un droit est en anglais ; son libellé est traduit dans `modules/Pipeline/lang/fr.json` et `en.json` (`pipeline.opportunities.close` → « Marquer une opportunité gagnée ou perdue »). Les droits sont regroupés par objet, en gardant à part ce qui engage le résultat commercial : conclure une opportunité et la supprimer. Les droits de l'ancien code n'étaient pas dans l'état des lieux : à rapprocher de ces neuf droits.

2. Ce que le module comporte

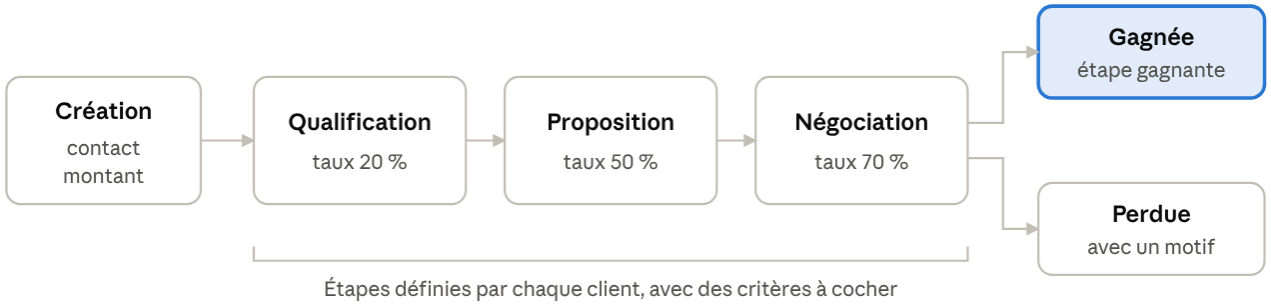
Pas de sous-modules : huit ensembles fonctionnels dans un seul module.

Ensemble	Contenu
Opportunités	Créer, modifier, dupliquer, supprimer, archiver une opportunité : nom, description, montant attendu, priorité, ressenti du vendeur, date de clôture prévue, source, contact, vendeur. Liste, tableau, opportunités perdues à part.

Ensemble	Contenu
Étapes et critères	Étapes propres à chaque client : nom, ordre, active ou non, étape gagnante, taux de closing. Critères à cocher dans chaque étape. Faire passer une opportunité d'une étape à l'autre.
Conclusion	Gagnée (passage dans l'étape gagnante, date de conclusion) ou perdue (motif, commentaire, date). Archivage pour sortir l'opportunité du pipeline sans la perdre.
Entourage d'une opportunité	Suiveurs (utilisateurs), interlocuteurs côté client (contacts), étiquettes, pièces jointes, rattachement d'un prospect.
Activités	Note, appel, email, rendez-vous ou activité libre, avec type (icône, couleur), date, durée, priorité, rappel et pièces jointes.
Paramétrage	Sources, motifs de perte, étiquettes, types d'activité.
Objectifs de vente	Objectif par société et par période (montant à atteindre) ; objectif en cours et montant gagné.
Statistiques et prévisions	Opportunités par étape, prévisions, répartition par source, comptages, export Excel.

Le cycle d'une opportunité :

Une opportunité avance d'étape en étape, puis se gagne ou se perd



Gagnée ou perdue, l'opportunité peut ensuite être archivée : elle sort du pipeline. Noms d'étapes et taux donnés en exemple.

La vie d'une opportunité · étapes, puis gagnée ou perdue

Les noms d'étapes et les taux sont des exemples : chaque client définit les siens.

3. Hors périmètre (renvois explicites)

Besoin	Où il vit
Fiches des contacts et des interlocuteurs	Module Contacts (requis), par sa vitrine `ContactDirectory`
Produits proposés dans une opportunité	Pont `pipeline-facturation` (§ 8.2)
Devis et factures d'une opportunité ; devis signé → opportunité gagnée	Pont `pipeline-facturation`
Rendez-vous du module Rendez-vous dans l'opportunité	Pont `pipeline-rdv` (hors périmètre de cette documentation)
Appel d'offres gagné → opportunité	Pont `appels-offres-pipeline` (hors périmètre)
Vendeurs, créateurs, suiveurs	Brique Auth du socle : ce sont des utilisateurs
Pièces jointes des opportunités et des activités	Brique Files du socle
Envoi des rappels	Brique Notifications du socle
Société et agence de rattachement	Brique Companies du socle

4. Entités et tables (base de chaque client)

Règles appliquées à toutes les tables : préfixe `pipeline_` ; clé primaire UUID (`uuid('id')->primary()` et trait `HasUuids`) ; clés étrangères seulement vers le socle ou vers Contacts, qui est requis.

Modèle (`Modules\Pipeline\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Opportunity`	`pipeline_opportunities` (`opportunities`)	`name`, `description`, `expected_income`, `status`, `priority`, `feeling`, `creation_date`, `closing_date`, `state_date`, `won_lost_date`, `archived_at`, `stage_id`, `source_id`, `contact_id` (Contacts), `seller_id`, `creator_id` (utilisateurs), `company_id`, `office_id`. **Change** : `company_id` remplace `society_id` ; pièces jointes par `HasAttachments`.
`OpportunityLoss`	`pipeline_opportunity_losses` (`lost_opportunities`)	`opportunity_id`, `loss_reason_id`, `date`, `comment` (ancien `loss_motif`).
`OpportunityCriterion`	`pipeline_opportunity_criteria` (`opportunity_criterias`)	`opportunity_id`, `stage_criterion_id`, `checked`.
— (pivot)	`pipeline_opportunity_followers` (`opportunity_followers`)	`opportunity_id`, `user_id` (`core_users`).
— (pivot)	`pipeline_opportunity_contacts` (`opportunity_collaborators`)	`opportunity_id`, `contact_id` : les interlocuteurs côté client.
— (pivot)	`pipeline_opportunity_tags` (`opportunity_tags`)	`opportunity_id`, `tag_id`.
`Stage`	`pipeline_stages` (`opportunity_states`)	`name`, `order`, `is_winning` (ancien `winner_state`), `closure_rate`, `active`, `company_id`.
`StageCriterion`	`pipeline_stage_criteria` (`opportunity_state_criterias`)	`stage_id`, `title`.
`Source`	`pipeline_sources` (`opportunity_sources`)	`name`, `company_id`.
`LossReason`	`pipeline_loss_reasons` (`reasons`)	`name`, `company_id`.

Modèle (`Modules\Pipeline\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Tag`	`pipeline_tags` (`tags`)	`name`, `company_id` (place à trancher, § 8).
`ActivityType`	`pipeline_activity_types` (`type_activities`)	`name`, `type`, `icon`, `color`, `active`, `default`, `company_id`.
`SalesTarget`	`pipeline_sales_targets` (`sales_targets`)	`name`, `start_date`, `end_date`, `target_amount`, `company_id`. **Change** : ajout de `company_id` (§ 8.3).
`Activity`	`pipeline_activities` (`activities`)	`opportunity_id`, `activity_type_id`, `creator_id`, `detail_type` / `detail_id`.
`NoteActivity`, `CallActivity`, `EmailActivity`, `MeetingActivity`, `CustomActivity`	`pipeline_note_activities`, `pipeline_call_activities`, `pipeline_email_activities`, `pipeline_meeting_activities`, `pipeline_custom_activities` (`note_activities`, `call_activities`, `email_activities`, `rdv_activities`, `custom_activities`)	Le détail de chaque activité, comme dans l'ancien code (titre, note, date, durée, priorité, rappel ; destinataires et contenu pour un email).

Anciennes tables non reprises : `opportunity_files` et `activity_files` (brique Files) ; `file_activities` (doublon sans modèle) ; `opportunity_product_sale_elements` (pont `pipeline-facturation`) ; `opportunity_notes` et `opportunity_state_histories` (déjà supprimées).

Règles métier reprises :

- Une opportunité est toujours dans une étape ; changer d'étape met à jour `state_date`.
- Gagner une opportunité la place dans l'étape gagnante et date sa conclusion ; la perdre demande un motif.
- Les critères cochés appartiennent à l'étape où ils ont été définis.
- Une activité est faite d'une ligne `pipeline_activities` et d'un détail dans l'une des cinq tables.
- La prévision vaut, en principe, montant attendu × taux de closing de l'étape (à vérifier dans l'ancien code).

Anomalies de l'ancien code, à ne pas reproduire : `PATCH hide/opportunity/{opportunity}` et `GET get/opportunity/{opportunity}/histories` pointent vers des méthodes absentes ; `percentage-by-source` et `count/statistics` utilisent un enum de permission inexistant ; `file_activities` fait doublon avec `activity_files`.

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

```
modules/Pipeline/Public/
├── Events/OpportunityCreated.php        # une opportunité est créée
├── Events/OpportunityStageChanged.php   # elle change d'étape
├── Events/OpportunityWon.php            # elle est gagnée
├── Events/OpportunityLost.php           # elle est perdue, avec son motif
└── Contracts/OpportunityDirectory.php   # trouver une opportunité, celles
d'un contact, la marquer gagnée
```

Aucun `use Modules\Pipeline\Models\…` hors du module. Le pont `pipeline-facturation` écoute l'approbation d'un devis dans Facturation et appelle `OpportunityDirectory` pour marquer l'opportunité gagnée ; il étend `Core\Modules\Public\ModuleListener` (méthode `whenActive`). Pipeline requiert Contacts : il ajoute lui-même l'onglet « Opportunités » sur la fiche contact, sans pont.

6. Droits, écrans, routes

Écrans : Pipeline (colonnes par étape) · Opportunité · Opportunités perdues · Objectifs · Statistiques · Paramètres, plus un onglet ajouté par chaque pont actif (Devis et factures).

Routes : `modules/Pipeline/routes/tenant.php`, servies sous `/api/v1/pipeline`. Le socle pose trois gardes : le client, la connexion (`RequireUser`, 401 sinon) et le module (`module:pipeline`, 403 sinon). Chaque action vérifie ensuite son droit par `$this->authorize('pipeline.…')`.

Périmètre : un utilisateur ne voit que les opportunités des sociétés qui lui sont accessibles.

Adresse (sous `/api/v1`)	Rôle · droit
`GET·POST /pipeline/opportunities`, `GET·PATCH /pipeline/opportunities/{id}`	Opportunités · `pipeline.opportunities.view`, `pipeline.opportunities.manage`
`DELETE /pipeline/opportunities/{id}`	Supprimer · `pipeline.opportunities.delete`

Adresse (sous `/api/v1`)	Rôle · droit
`POST …/{id}/duplicate`, `POST …/{id}/stage`, `PATCH …/{id}/criteria`, `POST …/{id}/feeling`, `POST …/{id}/archive`	Dupliquer, changer d'étape, cocher les critères, ressenti, archiver · `pipeline.opportunities.manage`
`POST …/{id}/win`, `POST …/{id}/lose`	Conclure · `pipeline.opportunities.close`
`GET·POST·DELETE …/{id}/followers`, `…/{id}/contacts`, `…/{id}/tags`, `POST …/{id}/files`	Entourage · `pipeline.opportunities.manage`
`GET·POST …/{id}/activities`, `PATCH·DELETE /pipeline/activities/{id}`	Activités · `pipeline.activities.manage` (lecture : `pipeline.opportunities.view`)
`GET·POST /pipeline/stages`, `PATCH·DELETE …/{id}`, `POST /pipeline/stages/reorder`	Étapes et critères · `pipeline.settings.manage`
`GET·POST /pipeline/sources`, `/pipeline/loss-reasons`, `/pipeline/tags`, `/pipeline/activity-types`, `PATCH·DELETE …/{id}`	Paramétrage · `pipeline.settings.manage`
`GET·POST /pipeline/sales-targets`, `GET /pipeline/sales-targets/current`	Objectifs · `pipeline.targets.manage` (lecture : `pipeline.statistics.view`)
`GET /pipeline/statistics/…`	Par étape, prévisions, par source, comptages · `pipeline.statistics.view`
`GET /pipeline/opportunities/export`	Export Excel · `pipeline.export`
`GET /pipeline/enums`	Priorités, statuts, types d'appel et d'email, icônes, rappels · `pipeline.opportunities.view`

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

Brique	Guichet ou trait	Usage par Pipeline
Kernel	`TenantController`, `Context`	Contrôleurs, utilisateur et société de la requête, format des réponses

Brique	Guichet ou trait	Usage par Pipeline
Auth	Modèle `User` (lecture), droits `pipeline.*`	Vendeurs, créateurs, suiveurs
Companies	`CompanyDirectory`, `CurrentCompany`, trait `BelongsToCompany`	Opportunités, paramétrage et objectifs par société
Files	`Attachments`, trait `HasAttachments`	Pièces jointes des opportunités et des activités
Audit	`Journal`	Création, changement d'étape, gain, perte, suppression
Notifications	`Notifier`	Rappels des activités, information des suiveurs
Modules	`ModuleListener`, les cinq portes	Routes et écouteurs refusés au client qui n'a pas le module

Dépendance à installer : `maatwebsite/excel` pour l'export.

8. Décisions pour la version 0.1.0

#	Sujet	Décision	État	Pourquoi
1	Suppression	La suppression d'une opportunité est conservée, avec un droit à part ; l'archivage reste disponible.	Décidé	Une opportunité n'est pas un document comptable.
2	Produits proposés	`opportunity_product_sale_elements` passe dans le pont `pipeline-facturation`.	Décidé	Facturation requiert déjà le Catalogue ; Pipeline n'en a pas besoin.
3	Objectifs de vente	Par société : ajout de `company_id`.	Décidé	Comme le reste du paramétrage.

#	Sujet	Décision	État	Pourquoi
4	Devis et factures d'une opportunité	Dans le pont `pipeline-facturation`.	Décidé (catalogue)	Pipeline et Facturation vivent chacun sans l'autre.
5	Pièces jointes	Par la brique Files ; `opportunity_files`, `activity_files` et `file_activities` disparaissent.	Proposé	Un seul stockage pour tous les modules.
6	Motifs de perte	Propres au Pipeline (`pipeline_loss_reasons`).	Proposé	Seule une opportunité peut être perdue.
7	Adresses de l'API	Nouvelles adresses REST sous `/api/v1/pipeline` ; les anciennes ne sont pas reprises.	Proposé	Même forme que tout le nouveau socle.
8	Clé du module	`pipeline`.	Proposé	Cohérente avec le catalogue et ses ponts `pipeline-*`.

Encore à trancher : `status`, étape gagnante et `archived_at` disent en partie la même chose, laquelle fait foi ? ; étiquettes propres au Pipeline ou partagées dans le socle ? ; historique des étapes à recréer, ou couvert par le journal (Audit) ? ; mécanisme des rappels d'activité ; formule exacte de la prévision.

9. Ordre d'intégration

1. Prérequis : Contacts, lot 1 (fiches et vitrine `ContactDirectory`).
2. Générer le module : `php artisan byplus:make-module pipeline`.
3. Lot 1 : paramétrage (étapes, critères, sources, motifs, étiquettes, types d'activité).
4. Lot 2 : opportunités, changement d'étape, conclusion, archivage, suppression, entourage.
5. Lot 3 : activités et rappels.
6. Lot 4 : objectifs de vente, statistiques, prévisions, export.

7. Ensuite : pont `pipeline-facturation` dès que Facturation a ses devis.
8. Reprise des données : `society_id` → `company_id` ; `company_id` des objectifs rempli avec la société principale du client ; fichiers vers la brique Files ; produits proposés vers la table du pont.

Chaque lot est livré avec ses tests dans `modules/Pipeline/tests`, vérifié sur une vraie base client, puis fusionné.

Terminé quand le test d'activation passe : sur un client neuf, on active Pipeline, ce qui active Contacts ; les écrans de Pipeline répondent, ceux des autres modules sont refusés ; un vendeur crée une opportunité pour un contact, la fait avancer en cochant les critères, la gagne ; une autre est perdue avec un motif ; un utilisateur sans `pipeline.opportunities.delete` ne peut pas supprimer. PHPStan et Deptrac sont verts.

BY+ · DOCUMENTATION MODULE

Module Facturation — document de définition

Version de travail du 5 octobre 2026.

Statut : version de travail, à valider en atelier, puis à suivre pour `php artisan byplus:make-module facturation`. Dans le § 8, les décisions marquées « Décidé » sont prises ; celles marquées « Proposé » attendent l'atelier.

Références : `docs/catalogue-modules.pdf` (§ 4 : module Facturation, requiert Catalogue et Contacts ; § 6 : ponts `pipeline-facturation`, `pos-facturation`, `facturation-fidelite`, `event-facturation`, `budget-facturation`, `portail-facturation`), `docs/architecture.pdf`, `docs/socle-mise-en-place.pdf`, documents de définition des modules POS, Contacts et Pipeline.

Ancien code de référence : `app/v1/Estimate` (devis : environ 5 600 lignes, 4 tables, 16 routes, 1 fichier de tests) et `app/v1/Invoice` (factures : environ 16 400 lignes, 11 tables, 37 routes, 6 fichiers de tests). Il sert de référence métier, pas de base à copier.

En bref

- Un module, clé `facturation`, vendable, qui requiert le module de base Catalogue et le module Contacts.
- Onze ensembles fonctionnels : devis, conversion, factures, validation, avoirs, acomptes, paiements, factures récurrentes, relances, documents et envois, suivi par client.
- Douze tables préfixées `facturation_`, en UUID, dans la base de chaque client.
- Dix droits, vérifiés sur chaque adresse de l'API, servie sous `/api/v1/facturation`.
- Hors du module : les produits et taxes (Catalogue), les clients (Contacts), le lien avec les opportunités (pont `pipeline-facturation`), la numérotation, les moyens et conditions de paiement, les circuits de validation (socle).

1. Identité (prête pour `module.php`)

```
return [
    'key'         => 'facturation',
    'name'        => 'Facturation',
    'version'     => '0.1.0',
```

```
    'requires'    => ['catalogue', 'contacts'],
    'sellable'    => true,
    'permissions' => [
        'facturation.estimates.view', 'facturation.estimates.manage',
        'facturation.invoices.view', 'facturation.invoices.manage',
        'facturation.invoices.credit_note',
        'facturation.payments.view', 'facturation.payments.manage',
        'facturation.payments.cancel',
        'facturation.recurring.manage', 'facturation.statistics.view',
    ],
    'settings'    => [],
    'schedule'    => [
        'facturation:generate-recurring'     => '01:00',
        'facturation:remind-unpaid-invoices' => '01:00',
        'facturation:remind-unpaid-advances' => '01:00',
    ],
    'provider'    => \Modules\Facturation\FacturationServiceProvider::class,
];
```

Test des 4 questions : un client s'en sert seul, avec Catalogue et Contacts activés automatiquement ✓ · menu et écrans propres (Devis, Factures, Paiements, Acomptes, Récurrentes) ✓ · droits propres ✓ · retirable sans casser un autre module, les ponts se désactivent seuls ✓ → module vendable.

Droits. Seule la clé d'un droit est en anglais ; son libellé est traduit dans `modules/Facturation/lang/fr.json` et `en.json` (`facturation.invoices.credit_note` → « Émettre un avoir »). Les droits sont regroupés par objet, en gardant à part ce qui engage de l'argent : émettre un avoir, enregistrer ou annuler un paiement. Les anciens `contact_invoice_view` et `contact_estimate_view` du module Contacts deviennent `facturation.invoices.view` et `facturation.estimates.view` . La validation d'une facture ne passe plus par un droit : c'est le circuit de validation du socle qui désigne qui valide (§ 8.3).

2. Ce que le module comporte

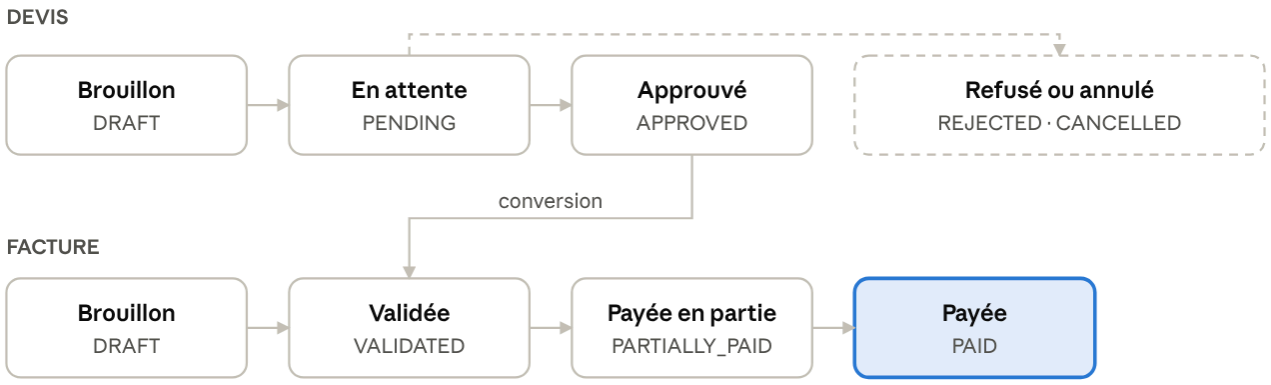
Pas de sous-modules : les anciens modules Devis et Factures forment un seul module, en onze ensembles.

Ensemble	Contenu
Devis	Créer en brouillon, modifier, supprimer un brouillon, dupliquer ; statuts `DRAFT` , `PENDING` , `APPROVED` , `REJECTED` , `CANCELLED` ; fichiers de validation signés.

Ensemble	Contenu
Conversion	Un devis approuvé devient une facture, une seule fois.
Factures	Créer, modifier un brouillon, archiver ; lignes avec prix, libellé et taxes photographiés ; remises ; montants HT, taxes, TTC et reste à payer ; photo du client et de la société ; devise et taux de change ; échéancier de paiement.
Validation	Passage de `DRAFT` à `VALIDATED` par le circuit de validation du socle, avec justificatifs.
Avoirs	Une facture « à l'envers », totale ou partielle, reliée à la facture d'origine.
Acomptes	Une part payée d'avance (montant ou pourcentage), avec son reste et son statut.
Paiements	Un ou plusieurs paiements par facture ou acompte, avec moyen de paiement et justificatifs ; annulation ; reçu PDF. Statuts `UNPAID`, `PAID`, `CANCEL`.
Factures récurrentes	Un modèle de facture répété : fréquence, limite (nombre ou date), prochaine date ; statuts `active`, `on_hold`, `completed`.
Relances	Relance automatique des factures et des acomptes impayés.
Documents et envois	PDF des devis, factures, reçus et acomptes ; envoi par email et annulation de l'envoi.
Suivi par client	Impayés, factures et devis d'un contact, meilleurs clients ; onglet « Factures » sur la fiche contact.

Le cycle d'un devis et d'une facture :

Un devis approuvé devient une facture, qui avance jusqu'au paiement



Autres statuts de facture : avoir (CREDITNOTE), annulée (CANCELED), archivée (ARCHIVED).
Aujourd'hui, la conversion crée la facture directement en VALIDATED, sans vérifier le statut du devis.

La vie d'une facture · du devis au paiement

L'envoi par email ne change pas le statut : il est suivi à part, dans `send_status`.

3. Hors périmètre (renvois explicites)

Besoin	Où il vit
Produits, prix, variantes, taxes	Module de base Catalogue (requis)
Clients et leur photo sur un document	Module Contacts (requis), par sa vitrine `ContactDirectory`
Lien d'un devis ou d'une facture avec une opportunité, produits proposés, devis approuvé → opportunité gagnée	Pont `pipeline-facturation` (§ 8.2)
Facture à partir d'un ticket de caisse	Pont `pos-facturation` (hors périmètre de cette documentation)
Points de fidélité, billetterie, budget, portail client	Ponts `facturation-fidelite`, `event-facturation`, `budget-facturation`, `portail-facturation` (hors périmètre)
Numérotation des devis, factures, avoirs, acomptes et paiements	Brique Paramètres du socle, numérotations (§ 8.6)
Moyens et conditions de paiement	Brique Paramètres du socle (§ 8.4)
Devises	Brique Currencies du socle
Qui valide une facture ou un devis	Circuits de validation du socle (§ 8.3)

Besoin	Où il vit
Modèles de documents (`template_id`)	Brique Personnalisation du socle
Envoi des emails et des relances	Brique Notifications du socle
Justificatifs et PDF enregistrés	Brique Files du socle

4. Entités et tables (base de chaque client)

Règles appliquées à toutes les tables : préfixe `facturation_` ; clé primaire UUID (`uuid('id')->primary()` et trait `HasUuids`) ; clés étrangères seulement vers le socle, le Catalogue et Contacts, qui sont requis ; devise par son code ISO, `$table->char('currency_code', 3)` (brique Currencies).

Modèle (`Modules\Facturation\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Estimate`	`facturation_estimates` (`estimates`)	`estimate_number` , `estimate_date` , `due_date` , `approved_date` , `status` , `send_status` , `notes` , `payment_delay` , montants (`sub_total` , `discount` , `total_ht` , `total_tax` , `total`), `currency_code` , `exchange_rate` , `contact_id` , `contact_data` et `company_data` (photos JSON), `legal_payment_information` , `payment_condition_id` , `payment_type_id` , `template_id` , `company_id` , `creator_id` , `model_type` / `model_id` . **Change** : `company_id` remplace `society_id` ; `currency_code` remplace `currency_id` ; `sent` supprimé (§ 8.8).
`EstimateLine`	`facturation_estimate_lines` (`estimate_product_sale_elements`)	`estimate_id` , `product_sale_element_id` (Catalogue), `name` , `description` , `price` , `quantity` , `discount_type` , `discount_value` .
`EstimateLineTax`	`facturation_estimate_line_taxes` (`estimate_product_sale_element_taxes`)	`estimate_line_id` , `tax_id` (Catalogue), `tax_name` , `tax_type` , `tax_amount` .

Modèle (`Modules\Facturation\Models\`)	Table (ancienne table)	Colonnes principales · ce qui change par rapport à l'ancien code
`Invoice`	`facturation_invoices` (`invoices`)	`invoice_number`, `invoice_date`, `due_date`, `payment_date`, `invoice_status` (`DRAFT`, `VALIDATED`, `PARTIALLY_PAID`, `PAID`, `CANCELED`, `CREDITNOTE`, `ARCHIVED`), `send_status`, `notes`, `payment_delay`, montants (`sub_total`, `total_ht`, `invoice_discount_value`, `invoice_total_tax`, `total`, `due_amount`), `currency_code`, `exchange_rate`, `contact_id`, `contact_data`, `company_data`, `legal_payment_information`, `payment_condition_id`, `payment_type_id`, `template_id`, `estimate_id`, `original_invoice_id` (avoir), `recurring_invoice_id`, `company_id`, `creator_id`, `model_type` / `model_id`. **Change** : comme pour les devis ; `sent` et `overdue` supprimés (§ 8.8).
`InvoiceLine`, `InvoiceLineTax`	`facturation_invoice_lines`, `facturation_invoice_line_taxes` (`invoice_product_sale_elements`, `invoice_product_sale_element_taxes`)	Comme les lignes et taxes de devis.
`InvoiceInstallment`	`facturation_invoice_installments` (`invoice_payment_conditions`)	`invoice_id`, `date`, `amount`, `percentage`, `status`, `payment_type_id` : l'échéancier.
`AdvanceInvoice`	`facturation_advances` (`advance_invoices`)	`advance_number`, `date`, `due_date`, `payment_date`, `type`, `amount`, `percentage`, `remaining_amount`, `status`, `send_status`, `invoice_id`, `payment_type_id`, `creator_id`.
`Payment`	`facturation_payments` (`payments`)	`payment_number`, `payment_date`, `due_date`, `amount`, `percentage`, `type`, `status` (`UNPAID`, `PAID`, `CANCEL`), `send_status`, `notes`, `payment_type_id`, `creator_id`, `payable_type` / `payable_id` (facture ou acompte). **Change** : justificatifs par `HasAttachments` ; `model_type` renommé `payable_type` pour ne pas le confondre avec le lien vers une opportunité.
`RecurringInvoice`	`facturation_recurring_invoices` (`recurring_invoices`)	`recurring_date`, `frequency`, `frequency_count`, `limit_by`, `limit_count`, `limit_date`, `next_recurring_at`, `status`, montants, `currency_code`, `exchange_rate`, `contact_id`, `payment_type_id`, `payment_condition_id`, `company_id`, `creator_id`.
`RecurringLine`, `RecurringLineTax`	`facturation_recurring_lines`, `facturation_recurring_line_taxes` (`recurring_product_sale_elements`, `recurring_product_sale_element_taxes`)	Comme les lignes et taxes de facture.

Anciennes tables non reprises : `invoice_taxes` (aucun modèle ; les taxes se recalculent depuis les lignes) ; `invoice_validates` et `estimate_validates` (circuit de validation et brique Files) ; `payment_files` (brique Files).

Règles métier reprises :

- Prix, libellés, taxes, client, société et devise sont photographiés : un document passé ne change pas quand le Catalogue ou la fiche contact change.
- Seul un devis brouillon se supprime ; une facture ne se supprime jamais : on l'archive ou on émet un avoir.
- Un devis se convertit une seule fois.
- Un avoir est une facture reliée à l'originale par `original_invoice_id`.
- Chaque paiement réduit le reste à payer ; la facture passe à `PARTIALLY_PAID`, puis `PAID` quand il est nul.
- Une facture récurrente génère ses factures chaque nuit, selon sa fréquence et sa limite.

Anomalies de l'ancien code, à ne pas reproduire : numéro calculé par `count() + 1` sans verrou (doublons possibles, `Invoice.php:73`, `Estimate.php:147`) ; conversion sans contrôle du statut du devis, facture créée directement en `VALIDATED`, sans transaction ; onze transactions jamais annulées en cas d'erreur ; `get/notPaid/collection` et `get/notPaid/advances` sans contrôle de droit ; `duplicate` et `followUp` des devis vers des méthodes absentes (erreur 500) ; relance des devis commentée dans `routes/console.php` ; `DeleteInvoiceRequest` et `DeletePaymentRequest` inutilisées et cassées ; verbes HTTP et noms de paramètres incohérents.

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

```
modules/Facturation/Public/
├── Events/EstimateApproved.php        # un devis passe à APPROVED
├── Events/InvoiceValidated.php        # une facture passe à VALIDATED
├── Events/InvoicePaid.php             # une facture est entièrement payée
├── Events/CreditNoteIssued.php        # un avoir, total ou partiel, est émis
├── Events/PaymentReceived.php         # un paiement est enregistré
├── Contracts/BillingDirectory.php     # devis et factures d'un contact,
reste à payer
└── Contracts/EstimateDrafts.php       # créer un devis brouillon à partir de
lignes, le rattacher à un objet
```

Aucun `use Modules\Facturation\Models\…` hors du module. Le pont `pipeline-facturation` utilise `EstimateDrafts` pour créer un devis depuis une opportunité et y écrire `model_type` / `model_id` : c'est le seul à écrire ce lien (§ 8.2). Il écoute

`EstimateApproved` pour marquer l'opportunité gagnée. `InvoicePaid` servira aux ponts Fidélité et Budget. Facturation requiert Contacts : elle ajoute elle-même l'onglet « Factures » sur la fiche contact, sans pont.

6. Droits, écrans, routes

Écrans : Devis · Factures · Avoirs · Paiements · Acomptes · Factures récurrentes, plus l'onglet « Factures » de la fiche contact et un onglet par pont actif.

Routes : `modules/Facturation/routes/tenant.php`, servies sous `/api/v1/facturation`. Le socle pose trois gardes : le client, la connexion (`RequireUser`, 401 sinon) et le module (`module:facturation`, 403 sinon). Chaque action vérifie ensuite son droit par `$this->authorize('facturation.…')`.

Périmètre : un utilisateur ne voit que les documents des sociétés qui lui sont accessibles.

Adresse (sous `/api/v1`)	Rôle · droit
`GET·POST /facturation/estimates`, `GET·PATCH·DELETE /facturation/estimates/{id}`	Devis ; suppression d'un brouillon seulement · `facturation.estimates.view`, `facturation.estimates.manage`
`POST …/estimates/{id}/status`, `POST …/{id}/duplicate`, `POST …/{id}/submit`	Changer le statut, dupliquer, soumettre au circuit de validation · `facturation.estimates.manage`
`POST /facturation/estimates/{id}/convert`	Convertir en facture · `facturation.invoices.manage`
`GET·POST /facturation/invoices`, `GET·PATCH /facturation/invoices/{id}`, `POST …/{id}/submit`, `POST …/{id}/archive`	Factures ; filtre `unpaid` (remplace `get/notPaid/collection`) · `facturation.invoices.view`, `facturation.invoices.manage`
`POST /facturation/invoices/{id}/credit-notes`	Émettre un avoir, total ou partiel · `facturation.invoices.credit_note`
`GET·POST /facturation/advances`, `GET /facturation/advances/{id}`, `POST …/{id}/pay`	Acomptes · `facturation.payments.view`, `facturation.payments.manage`
`GET·POST /facturation/payments`, `GET /facturation/payments/{id}`, `POST …/{id}/pay`	Paiements · `facturation.payments.view`, `facturation.payments.manage`

Adresse (sous `/api/v1`)	Rôle · droit
`POST /facturation/payments/{id}/cancel`, `POST /facturation/advances/{id}/cancel`	Annuler un paiement ou un acompte · `facturation.payments.cancel`
`GET …/{id}/pdf`, `POST …/{id}/send`, `POST …/{id}/send/cancel` (devis, factures, acomptes, paiements)	PDF et envoi · droit de lecture de l'objet
`GET·POST /facturation/recurring-invoices`, `GET·PATCH …/{id}`, `POST …/{id}/status`	Factures récurrentes · `facturation.recurring.manage`
`GET /facturation/contacts/{contactId}/summary`	Onglet « Factures » de la fiche contact · `facturation.invoices.view`
`GET /facturation/statistics/…`	Meilleurs clients, impayés · `facturation.statistics.view`
`GET /facturation/enums`	Statuts, fréquences · `facturation.estimates.view`

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

Brique	Guichet ou trait	Usage par Facturation
Kernel	`TenantController`, `Context`	Contrôleurs, utilisateur et société de la requête, format des réponses
Auth	Modèle `User` (lecture), droits `facturation.*`	Créateurs ; droits du manifeste
Companies	`CompanyDirectory`, `CurrentCompany`, trait `BelongsToCompany`	Société émettrice et sa photo sur chaque document
Paramètres	Numérotations, moyens et conditions de paiement	Numéros uniques ; listes de paiement
Circuits de validation	— (guichet à nommer dans le socle)	Validation des devis et des factures
Personnalisation	Modèles de documents	Mise en page des PDF
Files	`Storage`, `Attachments`, trait `HasAttachments`	Justificatifs, PDF

Brique	Guichet ou trait	Usage par Facturation
Audit	`Journal`	Création, validation, conversion, avoir, paiement, annulation
Notifications	`Notifier`	Envoi des documents par email, relances
Currencies	`Currencies`	Devise des documents et des clients
Modules	`ModuleListener`, les cinq portes	Routes, tâches planifiées et écouteurs refusés au client qui n'a pas le module

Dépendance à installer : un paquet de génération de PDF ; `barryvdh/laravel-dompdf`, déjà retenu pour POS, à confirmer pour les modèles de l'ancienne facturation.

8. Décisions pour la version 0.1.0

#	Sujet	Décision	État	Pourquoi
1	Devis et factures	Un seul module, Facturation.	Décidé (catalogue)	Un devis n'a de sens que pour facturer.
2	Lien avec une opportunité	`model_type` / `model_id` est gardé, mais seul le pont `pipeline-facturation` l'écrit.	Décidé	Facturation ne doit pas connaître Pipeline.
3	Validation	Par les circuits de validation du socle.	Décidé	Le même circuit sert à tous les modules.
4	Moyens et conditions de paiement	Dans le socle.	Décidé	Référentiels communs. POS utilise la même liste : le § 8.4 du document POS (table pos_payment_types) est à mettre à jour.

#	Sujet	Décision	État	Pourquoi
5	Devise du client	Pas de table à part : chaque devis, facture et facture récurrente porte sa devise dans `$table->char('currency_code', 3)` (code ISO).	Décidé	La devise ne sert qu'à facturer.
6	Numérotation	Numéros fournis par le socle, avec contrainte d'unicité en base.	Proposé	L'ancien `count() + 1` donne des doublons.
7	Conversion	Seul un devis `APPROVED` se convertit ; la facture naît en `DRAFT` à la date du jour ; toute la copie dans une transaction.	Proposé	Aujourd'hui, un devis refusé peut être facturé et une erreur laisse une facture incomplète.
8	Colonnes en double	`send_status` seul (plus de `sent`) ; `overdue` calculé (échéance passée et facture non payée).	Proposé	Deux colonnes pour une même information finissent par se contredire.
9	Adresses de l'API	Nouvelles adresses REST sous `/api/v1/facturation` ; les anciennes ne sont pas reprises.	Proposé	Même forme que tout le nouveau socle.
10	Clé du module	`facturation`.	Proposé	Cohérente avec le catalogue et ses ponts `*-facturation`.

Encore à trancher : archivage par le statut `ARCHIVED` ou par `archived_at` ; relance des devis et des factures non validées : à garder, ou laissée au circuit de validation ? ; préfixes de numérotation par défaut.

La normalisation DGI n'est pas prévue pour les factures en 0.1.0.

9. Ordre d'intégration

1. Prérequis : module de base Catalogue ; Contacts, lot 1 ; dans le socle, numérotations, moyens et conditions de paiement, circuits de validation.
2. Générer le module : `php artisan byplus:make-module facturation`.

3. Lot 1 : devis, PDF, envoi par email.
4. Lot 2 : factures, validation, conversion, avoirs.
5. Lot 3 : acomptes, paiements, échéanciers.
6. Lot 4 : factures récurrentes, relances, onglet « Factures » de la fiche contact, statistiques.
7. Ensuite : pont `pipeline-facturation` avec Pipeline ; ponts `pos-facturation`, `budget-facturation`, `portail-facturation` au fil des modules voisins.
8. Reprise des données : `society_id` → `company_id` ; identifiants de devise → `currency_code` ; numéros repris tels quels ; `sent` et `overdue` abandonnés ; fichiers de validation vers la brique Files ; devise des contacts abandonnée, chaque document portant la sienne ; liens vers les opportunités confiés au pont.

Chaque lot est livré avec ses tests dans `modules/Facturation/tests`, vérifié sur une vraie base client, puis fusionné.

Terminé quand le test d'activation passe : sur un client neuf, on active Facturation, ce qui active Catalogue et Contacts ; les écrans de Facturation répondent, ceux des autres modules sont refusés ; on crée un devis, on l'approuve, on le convertit en facture brouillon, on la fait valider, deux paiements la passent à `PAID` ; on émet un avoir partiel ; un devis refusé ne peut pas être converti ; deux factures créées en même temps reçoivent deux numéros différents ; un utilisateur sans `facturation.invoices.credit_note` ne peut pas émettre d'avoir. PHPStan et Deptrac sont verts.