BTS SIO — Option SLAM

Documentation Technique de Réalisation

Application de gestion des visites médicales

Client lourd PyQt6 — GSB AppliVisiteurs

Candidat	Cyrille COUR
Établissement	Aristée
Organisation	Galaxy Swiss Bourdin (GSB) — fictive
Stack technique	Python 3.13 / PyQt6 / MySQL (Laragon)
Date	Juin 2026 -> mai 2027

Sommaire

Sommaire 2
1. Mise en place de l'environnement 4
1.1 Création du dossier projet 4
1.2 Environnement virtuel Python 4
1.3 Installation des dépendances 4
1.4 Structure de fichiers (architecture MVC) 4
1.5 Fichiers de documentation 4
1.6 Initialisation Git & correction du tracking 4
2. Mise en place de la base de données 5
2.1 Choix technique 5
2.2 Création de la BDD 5
2.3 Adaptation de la table visiteur 5
2.4 Transfert des visiteurs depuis AppliFrags 5
2.5 Données de test 6
3. Page de login 6
3.1 Dépendance supplémentaire 7
3.2 Fichiers créés 7
3.3 Design et comportement 7
3.4 Configuration BDD 8
4. Architecture graphique core & identité visuelle 9
4.1 Structure de la fenêtre principale et menu latéral 9
4.2 Bandeau supérieur de bienvenue dynamique (Topbar) 9
4.3 Gestion des assets système (assets/logo.ico) 10
5. Logique métier, CRUD & persistance des données SQL 10
5.1 Module de connexion et d'authentification sécurisé 11
5.2 Le tableau de bord centralisé (Flat Design) 11
5.3 Gestionnaire métier des comptes-rendus 12
5.4 Module d'agenda graphique et interactif 14
5.5 Page Archives 14
6. Module Délégué régional 15
6.1 Périmètre fonctionnel couvert 16
6.2 Architecture et navigation 16
6.2.1 Nouveaux fichiers créés 16
6.3 Vue d'activité de la région 17
6.3.1 Fonctionnement 17

6.4 Statistiques graphiques 18
6.4.1 Les quatre graphiques (PyQt6-Charts) 18
6.5 Gestion des échantillons 20
6.5.1 Onglet « Demandes & Distributions » 20
6.5.2 Onglet « Historique complet » 20
6.5.3 Statuts calculés 20
6.6 Modifications transverses liées au module 22
6.6.1 Nouvelle table de liaison rapport_visite_medicament 22
6.6.2 Formulaire de compte-rendu (comptes_rendues_page.py) 22
6.6.3 Page Dashboard (dashboard_page.py) 22
6.7 Synthèse de conformité 23
7. Module Responsable de secteur 24
7.1 Périmètre fonctionnel couvert 24
7.2 Architecture et navigation 24
7.2.1 Nouveaux fichiers créés 24
7.3 Comptes-rendus du secteur 24
7.4 Statistiques du secteur 25
7.4.1 Les quatre graphiques (PyQt6-Charts) 25
7.5 Statistiques par visiteur 25
7.6 Contrôle des stocks d'échantillons 25
7.6.1 Onglet « Vue globale du secteur » 26
7.6.2 Onglet « Anomalies & alertes » 26
7.6.3 Statuts calculés 26
7.7 Gestion des collaborateurs 26
7.8 Synthèse de conformité 27

1. Mise en place de l'environnement

1.1 Création du dossier projet

Création d'un dossier dédié sur le bureau :

```
C:\Users\courc\Desktop\GSB-Applivisiteur\
```

1.2 Environnement virtuel Python

Dans le terminal PowerShell, à la racine du projet :

```
python -m venv venv
venv\Scripts\activate
```

Le préfixe (venv) confirme que l'environnement est actif.

1.3 Installation des dépendances

```
pip install PyQt6 PyQt6-Charts mysql-connector-python
```

Package	Rôle
`PyQt6`	Framework interface graphique
`PyQt6-Charts`	Graphiques et statistiques
`mysql-connector-python`	Connexion à la base de données MySQL

1.4 Structure de fichiers (architecture MVC)

```
mkdir controllers, views, views\visiteur, views\delegue, views\responsable, models,
services, assets

New-Item main.py, config.py, controllers\auth_controller.py, models\visiteur.py,
  models\praticien.py, models\rapport.py, services\db_service.py,
  views\login_window.py, views\main_window.py -ItemType File
```

1.5 Fichiers de documentation

- README.md — résumé du projet, stack technique, structure des dossiers, étapes de démarrage à chaque session de dev
- .gitignore — exclusion du venv/, de config.py (credentials BDD), des fichiers système et temporaires

1.6 Initialisation Git & correction du tracking

Le .gitignore ayant été ajouté après le premier push, nettoyage du cache Git pour appliquer les règles rétroactivement :

```
git rm -r --cached .
git add .
git commit -m "fix: remove tracked files that should be ignored"
git push
```

2. Mise en place de la base de données

2.1 Choix technique

Base de données MySQL hébergée localement via Laragon, accessible sur localhost:3306. La BDD se nomme gsb-applivisiteurs.

2.2 Création de la BDD

Import du script create_db.sql via phpMyAdmin (onglet SQL).

Tables créées :

Table	Rôle
`secteur`	Secteurs géographiques (niveau responsable)
`region`	Régions rattachées à un secteur (niveau délégué)
`visiteur`	Comptes utilisateurs (repris d'Applifrais)
`famille`	Familles de médicaments
`medicament`	Catalogue produits du laboratoire
`type_praticien`	Type et lieu d'exercice
`specialite`	Spécialités médicales
`praticien`	Médecins, pharmaciens, infirmiers visités
`rapport_visite`	Comptes-rendus de visite
`rapport_visite_medicament`	Médicaments présentés + échantillons par visite
`dotation`	Stocks d'échantillons attribués (géré par le délégué)

2.3 Adaptation de la table visiteur

La table visiteur d'Applifrais a été adaptée pour AppliVisiteurs :

Colonne	Modification
`fraisKM`	Supprimée (spécifique AppliFrags)
`region_id`	Ajoutée (rattachement hiérarchique)
`role`	Étendu à visiteur, delegue, responsable

2.4 Transfert des visiteurs depuis Applifrais

Plutôt qu'un export/import SQL manuel, un script Python transfert_visiteurs.py a été utilisé pour transférer directement les données entre les deux BDD via le connecteur MySQL.

- 26 visiteurs transférés depuis gsb-applifrais
- 1 erreur ignorée : compte g33 sans adresse (compte admin Applifrais, non pertinent pour AppliVisiteurs)
- region_id laissé à NULL pour tous — renseigné ultérieurement lors du développement des modules Délégué et Responsable

- Le script a été supprimé après utilisation (données de connexion en dur)

2.5 Données de test

Données fictives insérées directement dans le script SQL :

- 3 praticiens (généraliste, spécialiste, pharmacien)
- 3 médicaments sur 3 familles différentes
- 1 secteur et 2 régions de test

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	varchar(10)	utf8mb4_0900_ai_ci		No	None			Change Drop More
2	nom	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
3	prenom	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
4	login	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
5	mdp	varchar(255)	utf8mb4_0900_ai_ci		No	None			Change Drop More
6	adresse	varchar(100)	utf8mb4_0900_ai_ci		No	None			Change Drop More
7	cp	varchar(5)	utf8mb4_0900_ai_ci		No	None			Change Drop More
8	ville	varchar(30)	utf8mb4_0900_ai_ci		No	None			Change Drop More
9	dateEmbauche	date			No	None			Change Drop More
10	role	enum('visiteur', 'delegue', 'responsable')	utf8mb4_0900_ai_ci		No	visiteur			Change Drop More
11	region_id	varchar(10)	utf8mb4_0900_ai_ci		Yes	NULL			Change Drop More

3. Page de login

3.1 Dépendance supplémentaire

```
pip install bcrypt
```

Nécessaire pour vérifier les mots de passe hashés en bcrypt (format $2y$... importés depuis Applifrais).

3.2 Fichiers créés

Fichier	Rôle
`views/login_window.py`	Interface graphique de la page de connexion
`controllers/auth_controller.py`	Logique d'authentification (requête BDD + vérification bcrypt)
`config.py`	Paramètres de connexion à la BDD
`main.py`	Point d'entrée de l'application
`views/main_window.py`	Fenêtre principale

3.3 Design et comportement

- Fenêtre sans bordure Windows (FramelessWindowHint) avec fond dégradé bleu marine
- Carte centrale arrondie aux couleurs GSB (bleu & blanc)
- Barre de titre custom avec boutons réduire / agrandir / fermer (✕ rouge au survol)
- Fenêtre déplaçable par drag sur la barre de titre
- Champs login et mot de passe avec validation à l'enregistrement
- Erreur générique en rouge (« Identifiant ou mot de passe incorrect. ») — sans préciser lequel des deux est faux, conformément au cahier des charges
- Touche Entrée déclenche la connexion
- Le hash du mot de passe n'est jamais transmis dans la session après authentification

3.4 Configuration BDD

Dans config.py :

```
DB_HOST     = "localhost"
DB_PORT     = 3306
DB_NAME     = "gsb-applivisiteurs"
DB_USER     = "root"
DB_PASSWORD = ""
```

config.py est dans .gitignore — ne jamais le commiter.

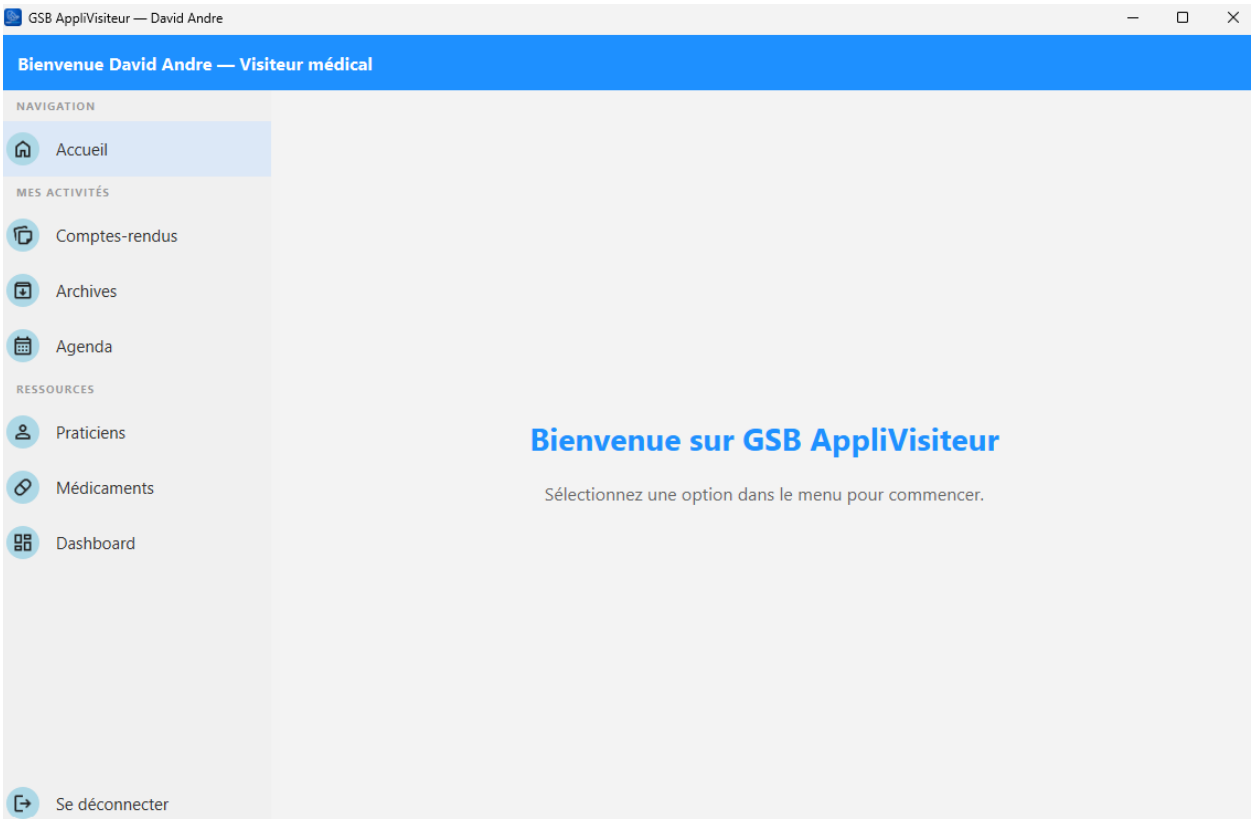
4. Architecture graphique core & identité visuelle

Cette phase décrit la conception de la coquille principale de l'application, l'ergonomie de navigation et l'intégration de la charte graphique de l'entreprise.

4.1 Structure de la fenêtre principale et menu latéral

La fenêtre principale (MainWindow) a été découpée à l'aide de gestionnaires de mise en forme (QHBoxLayout) pour séparer l'écran en deux zones distinctes : un menu de navigation fixe à gauche et un espace central dynamique à droite utilisant un QStackedWidget pour commuter entre les pages.

- Le menu latéral : conçu via un QVBoxLayout, il intègre des boutons de commande textuels associés à des icônes vectorielles
- Les ressources graphiques : pour garantir la légèreté de l'application et éviter la pixellisation, les icônes ont été directement extraites au format vectoriel depuis la bibliothèque Google Fonts
- Le logo circulaire : le logo officiel de GSB a été intégré en haut de cette sidebar, avec un masque de découpe circulaire appliqué au conteneur de l'image



4.2 Bandeau supérieur de bienvenue dynamique (Topbar)

Placé tout en haut de l'espace de travail central, ce bandeau horizontal sert de repère visuel majeur en reprenant la couleur bleue institutionnelle du laboratoire GSB.

- Logique de session : ce composant intercepte l'objet d'état (le dictionnaire Python self.visiteur) généré après l'authentification
- Affichage : extraction et concaténation des chaînes de caractères pour afficher dynamiquement le message « Bonjour [Prénom] [NOM] — [Rôle] », adapté selon que l'utilisateur est visiteur, délégué ou responsable

4.3 Gestion des assets système (assets/logo.ico)

Pour que l'application s'intègre parfaitement à l'environnement d'exécution Windows, l'arborescence du code source a été dotée d'un dossier assets/ regroupant les médias.

- Le logo du projet a été converti au format multi-résolution .ico, avec un .png en complément pour fiabiliser le rendu sur certaines configurations Windows
- À l'initialisation, app.setWindowIcon(QIcon("assets/logo.ico")) remplace le logo d'exécution standard de Python par l'identité visuelle de GSB

5. Logique métier, CRUD & persistance des données SQL

Cette phase détaille l'intelligence de l'application : l'interconnexion avec MySQL (Laragon), la sécurisation des requêtes et le traitement des données métiers.

5.1 Module de connexion et d'authentification sécurisé

Avant d'accéder au cœur de l'application, l'utilisateur est bloqué par une interface de contrôle d'accès (LoginWindow).

- Sécurisation des requêtes : pour interroger la table visiteur, le script utilise le mécanisme des requêtes préparées fourni par mysql.connector. Les arguments saisis sont passés via un tuple isolé de la structure textuelle de la requête

```
SELECT * FROM visiteur WHERE login = %s AND mdp = %s
```

Impact : neutralisation totale des tentatives d'attaques par injection SQL.

- Initialisation de la session : si la base renvoie une ligne valide, les données (ID, Nom, Prénom, Rôle) sont packagées dans un dictionnaire persistant transmis au constructeur de la fenêtre principale : MainWindow(resultat_bdd)
- Résolution d'architecture : pour éviter le plantage d'importation circulaire (le Login appelant la MainWindow et vice versa), l'import de la classe MainWindow a été déplacé localement, directement au sein de la méthode de redirection

5.2 Le tableau de bord centralisé (Flat Design)

Le Dashboard fait office d'écran de pilotage d'activité. Son interface a été épurée pour adopter un style « Flat Design » sans fioritures ni bordures doubles imbriquées.

- Les cartes d'indicateurs (KPI) : chaque bloc est un QFrame doté d'une feuille de style QSS globale définissant un arrière-plan teinté et une bordure colorée pleine de 2px
- □ TOTAL PRATICIENS → SELECT COUNT(*) FROM praticien
- □ MES RAPPORTS (CE MOIS) → filtre l'historique du visiteur connecté sur le mois et l'année en cours via MONTH(CURRENT_DATE()) et YEAR(CURRENT_DATE())
- □ TOTAL MES RAPPORTS → calcule l'implication globale du visiteur depuis son embauche
- □ VISITES À VENIR → compte les lignes de la table agenda dont l'échéance est supérieure ou égale à l'instant T
- Le flux d'actualité sécurisé : un QListWidget affiche par jointure SQL les 5 derniers rapports rédigés. Les interactions ont été verrouillées pour éviter tout comportement instable

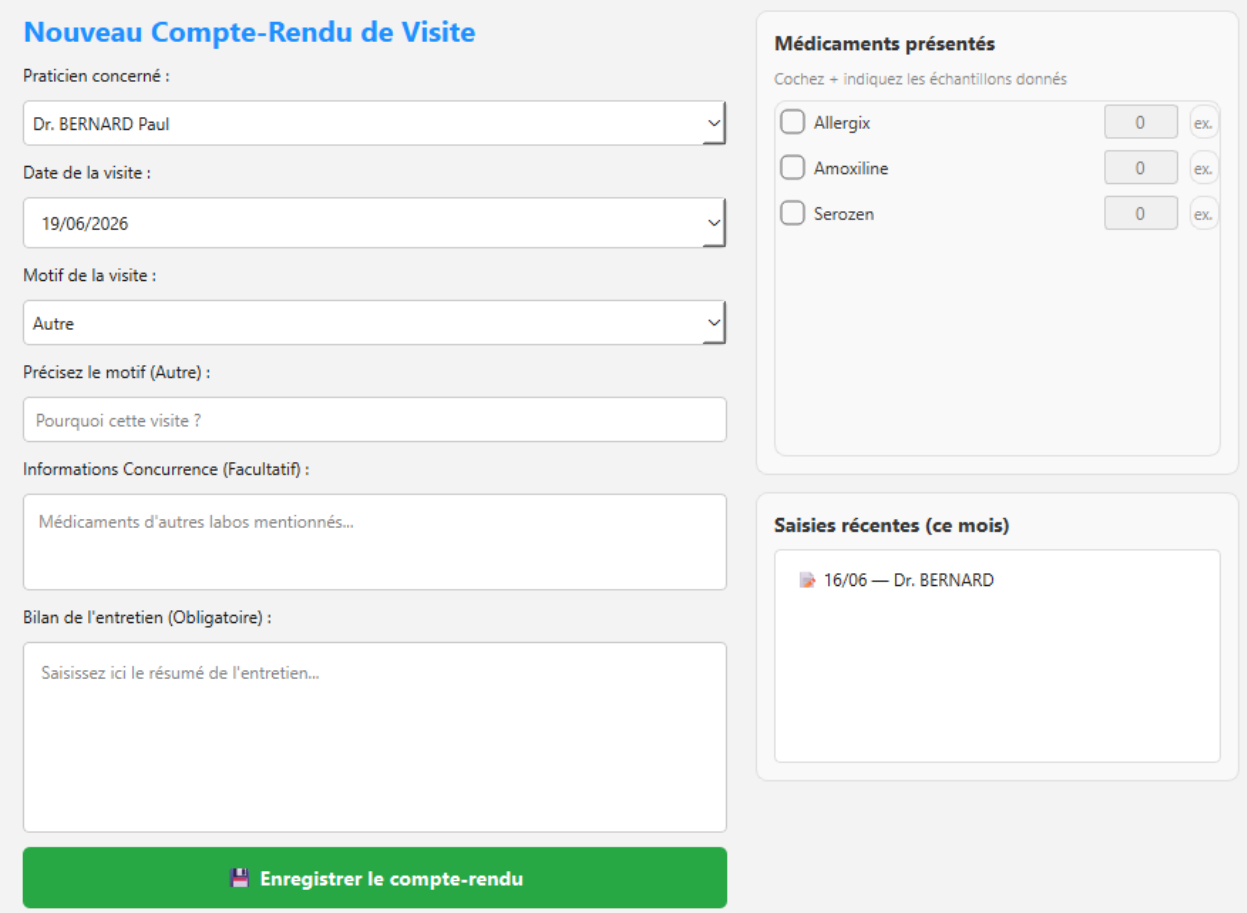
```
self.recent_reports_list.setSelectionMode(QListWidget.SelectionMode.NoSelection)
self.recent_reports_list.setFocusPolicy(Qt.FocusPolicy.NoFocus)
```

La page se rafraîchit automatiquement à chaque retour dessus via la méthode showEvent(), pour refléter immédiatement les comptes-rendus qui viennent d'être saisis.

5.3 Gestionnaire métier des comptes-rendus

Cette interface permet la rédaction des comptes-rendus de visite obligatoires après chaque entretien avec un médecin.

- Alimentation dynamique : au chargement, une requête parcourt la table praticien pour injecter la liste des médecins disponibles sous la forme « Dr. [NOM] [Prénom] » dans un QComboBox. Une table de hachage interne (self.praticiens_map) mémorise la correspondance entre l'index du menu et l'ID réel en base
- Contrôle événementiel du formulaire : le menu déroulant du motif de visite écoute les changements d'état (currentTextChanged). Si l'utilisateur clique sur « Autre », un signal active instantanément le champ de saisie complémentaire ; sinon, le champ se vide et se grise automatiquement
- Veille concurrentielle : un champ de texte étendu (QTextEdit) « Concurrence » permet de consigner les mouvements des laboratoires concurrents signalés par le médecin (colonne concurrence de la table rapport_visite)
- Médicaments présentés : une liste scrollable de médicaments cochables, chacun révélant un champ de quantité d'échantillons donnés, alimente la table de liaison rapport_visite_medicament
- Historique glissant : à droite, un tableau récapitule les saisies récentes du visiteur, limité au mois en cours


Nouveau Compte-Rendu de Visite
Praticien concerné :
Dr. BERNARD Paul
Date de la visite :
19/06/2026
Motif de la visite :
Autre
Précisez le motif (Autre) :
Pourquoi cette visite ?
Informations Concurrence (Facultatif) :
Médicaments d'autres labos mentionnés...
Bilan de l'entretien (Obligatoire) :
Saisissez ici le résumé de l'entretien...
Enregistrer le compte-rendu
Médicaments présentés
Cochez + indiquez les échantillons donnés
Allergix
0
ex.
Amoxiline
0
ex.
Serozen
0
ex.
Saisies récentes (ce mois)
16/06 — Dr. BERNARD

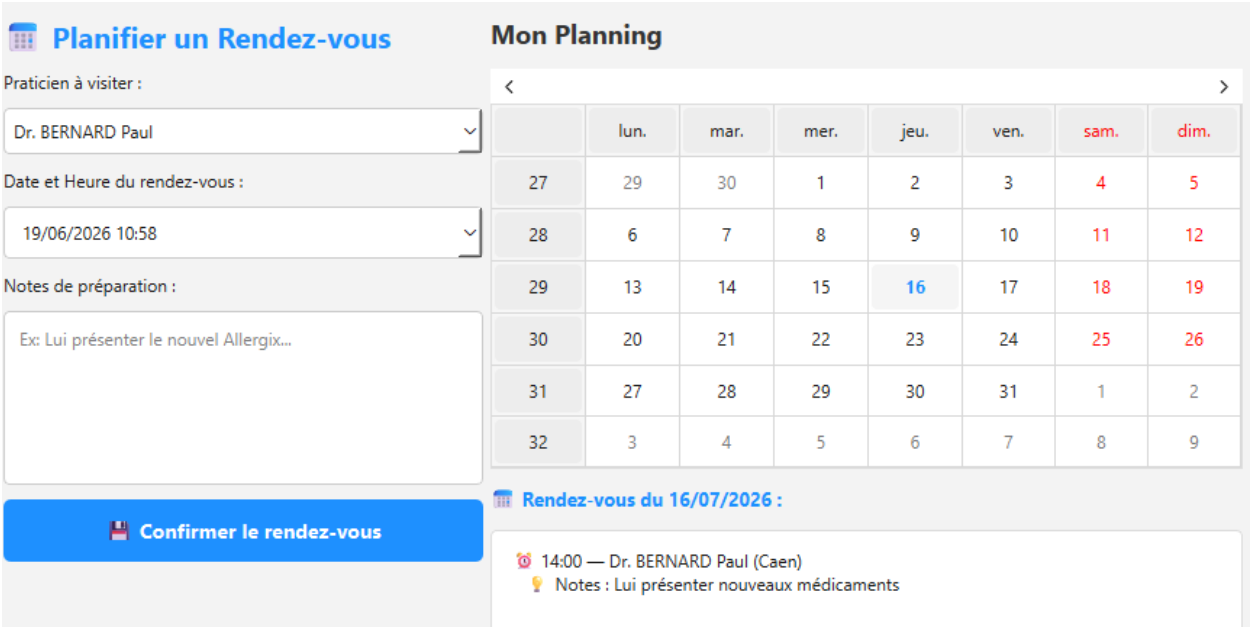
5.4 Module d'agenda graphique et interactif

Pour moderniser le suivi des tournées, l'affichage des lignes de rendez-vous a été remplacé par un calendrier visuel interactif basé sur le widget QCalendarWidget.

- Création de la table de persistance : une table agenda a été modélisée sous Laragon pour enregistrer les futurs rendez-vous

```
CREATE TABLE `agenda` (
  `id` INT AUTO_INCREMENT PRIMARY KEY,
  `date_heure` DATETIME NOT NULL,
  `statut` ENUM('planifie', 'annule', 'realise') DEFAULT 'planifie',
  `commentaire` TEXT,
  `visiteur_id` VARCHAR(10),
  `praticien_id` INT,
  FOREIGN KEY (`visiteur_id`) REFERENCES `visiteur`(`id`),
  FOREIGN KEY (`praticien_id`) REFERENCES `praticien`(`id`)
);
```

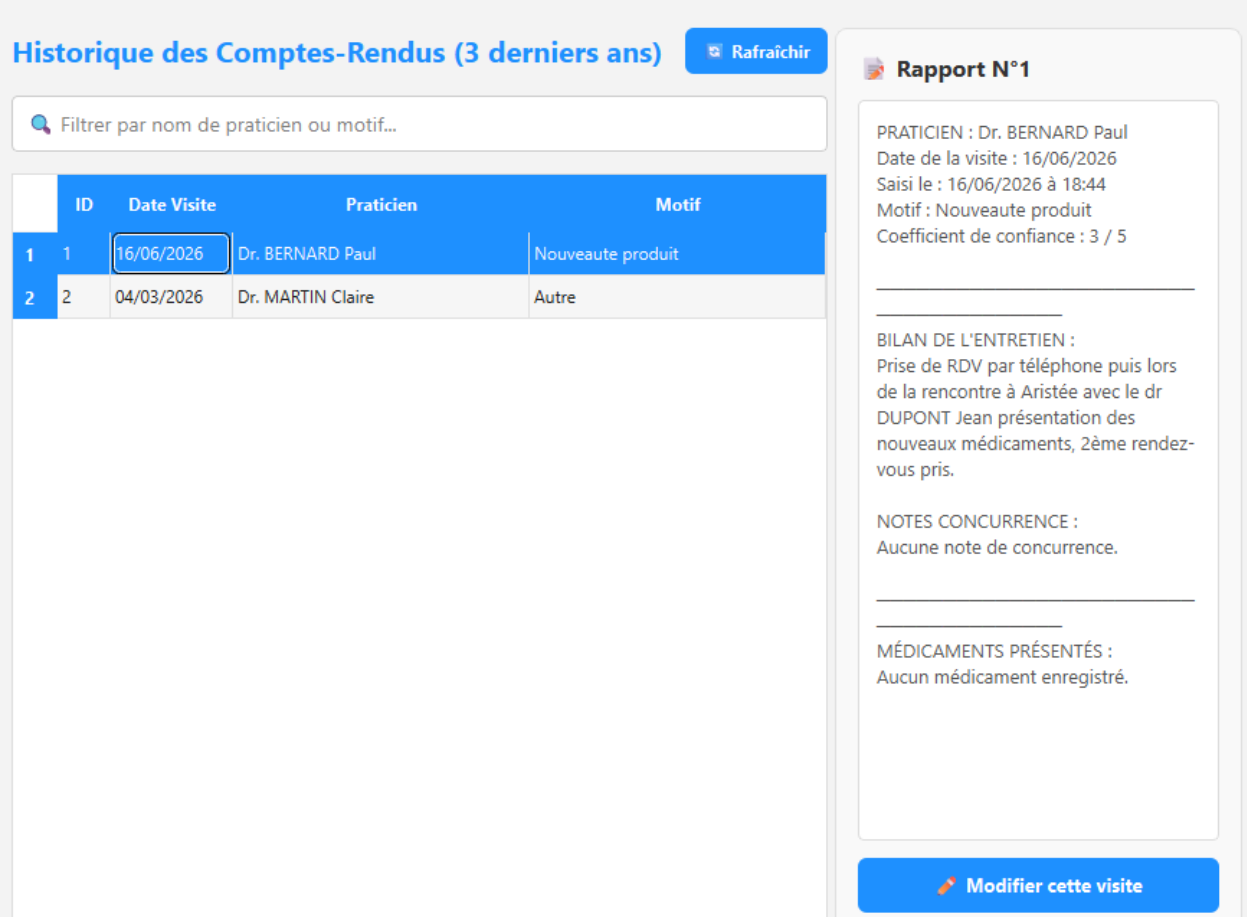
- Planification spécifique : la zone de gauche intègre un formulaire de saisie (QComboBox pour le médecin, QDateTimeEdit pour le jour et l'heure, QTextEdit pour les objectifs de la visite)
- Rendu visuel et coloration dynamique : load_agenda_data extrait toutes les planifications actives du visiteur connecté ; les jours possédant un rendez-vous basculent en caractères gras et en couleur bleue (QTextCharFormat)
- Filtrage événementiel synchrone : le calendrier écoute le signal selectionChanged ; au clic, update_day_details() affiche instantanément le détail des visites prévues pour ce jour



5.5 Page Archives

La page Archives donne accès à l'historique complet des comptes-rendus du visiteur sur les trois dernières années, avec recherche en direct et panneau de détail.

- Recherche en direct par nom de praticien ou motif
- Panneau de détail affichant le bilan complet, les notes de concurrence et désormais la liste des médicaments présentés avec leur nombre d'échantillons
- Modification possible des comptes-rendus du mois en cours uniquement, via une fenêtre de dialogue réutilisant le formulaire de saisie



6. Module Délégué régional

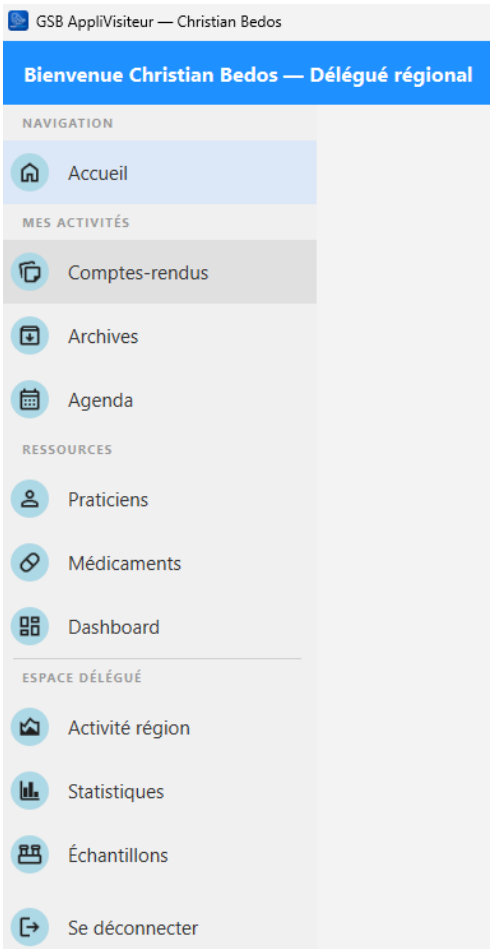
Le Délégué régional est un visiteur médical qui consacre une partie de son temps à l'encadrement de l'équipe de sa région. Il conserve donc l'intégralité des fonctionnalités du profil Visiteur (saisie de comptes-rendus, agenda, consultation des praticiens et médicaments), auxquelles s'ajoutent quatre fonctionnalités propres à son rôle d'encadrement.

6.1 Périmètre fonctionnel couvert

- Vue d'activité de sa région — consultation des comptes-rendus de toute l'équipe
- Statistiques graphiques — visualisation de l'activité par visiteur ou pour toute la région
- Détail d'un visiteur — zoom sur l'activité d'un collaborateur en particulier
- Gestion des échantillons — enregistrement des dotations demandées et distribuées

6.2 Architecture et navigation

Le module Délégué repose sur une sidebar dynamique : la fenêtre principale (main_window.py) construit ses boutons de navigation en fonction du rôle de l'utilisateur connecté. Pour un délégué, deux blocs apparaissent : la section commune « Visiteur » puis, après un séparateur visuel et le libellé « Espace Délégué », les trois pages exclusives : Activité région, Statistiques, Échantillons.



6.2.1 Nouveaux fichiers créés

Fichier	Rôle
`views/pages/delegue/__init__.py`	Marqueur de package Python (vide)
`views/pages/delegue/activite_region_page.py`	Page de consultation des comptes-rendus de l'équipe
`views/pages/delegue/stats_region_page.py`	Page des 4 graphiques PyQt6-Charts
`views/pages/delegue/echantillons_page.py`	Page de gestion des dotations d'échantillons

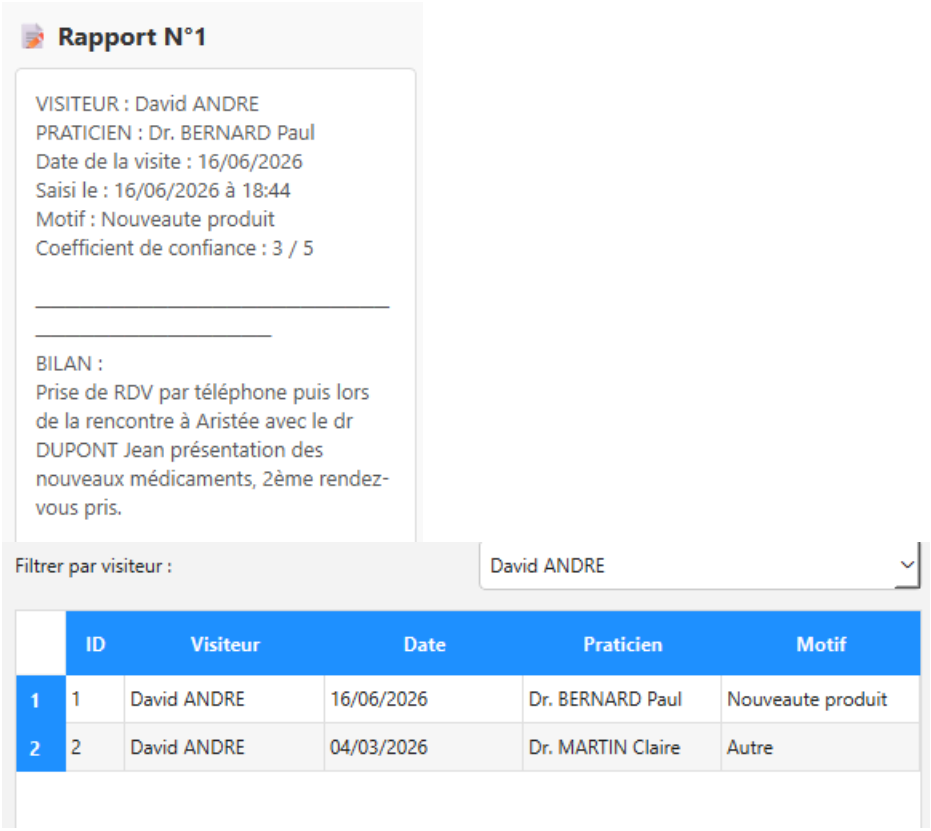
6.3 Vue d'activité de la région

Fichier concerné : **`views/pages/delegue/activite_region_page.py`**

Cette page répond à l'exigence fonctionnelle : « le délégué visualise l'ensemble des comptes-rendus rédigés par les visiteurs qui lui sont rattachés ». Elle interroge la base sur le critère region_id du délégué connecté, en jointure avec les tables visiteur et praticien.

6.3.1 Fonctionnement

- Tableau listant tous les comptes-rendus de la région (colonnes : ID, Visiteur, Date, Praticien, Motif)
- Combo de filtre permettant d'isoler les comptes-rendus d'un seul visiteur, ou de revenir à « Tous les visiteurs »
- Panneau de détail à droite : au clic sur une ligne, affichage complet du compte-rendu (praticien, dates, motif, coefficient de confiance, bilan, notes concurrence)



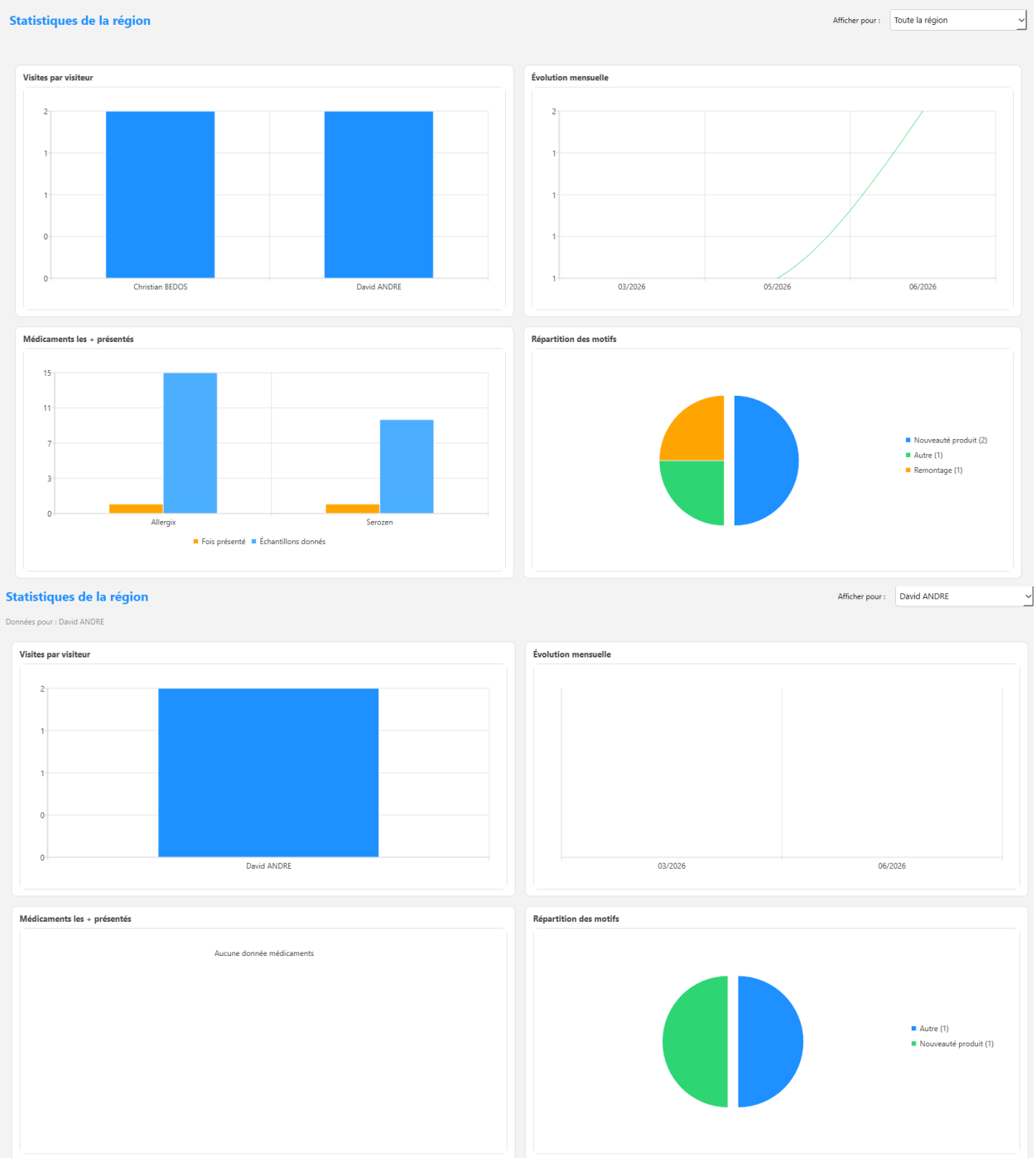
6.4 Statistiques graphiques

Fichier concerné : **`views/pages/delegue/stats_region_page.py`**

Cette page couvre à la fois l'exigence « statistiques graphiques » et « détail d'un visiteur » du cahier des charges. Un unique combo de filtre en haut de la page (« Toute la région » ou un visiteur nommé) pilote simultanément les quatre graphiques affichés. Sélectionner un visiteur revient donc à « zoomer » sur son activité individuelle.

6.4.1 Les quatre graphiques (PyQt6-Charts)

- Visites par visiteur — diagramme en barres montrant le nombre de comptes-rendus rédigés par chaque membre de l'équipe
- Évolution mensuelle — courbe (QSplineSeries) du nombre de visites mois par mois
- Médicaments les plus présentés — barres groupées comparant la fréquence de présentation et le total d'échantillons distribués pour chaque médicament, via la table de jonction rapport_visite_medicament
- Répartition des motifs de visite — camembert (QPieSeries) avec le motif dominant mis en valeur (slice explodée)


Statistiques de la région
Afficher pour :
Toute la région
Visites par visiteur
2
1
1
0
Christian BEDOS
David ANDRE
Évolution mensuelle
2
1
1
1
1
03/2026
05/2026
06/2026
Médicaments les + présentés
15
11
7
3
0
Allergix
Serozen
Fois présenté
Échantillons donnés
Répartition des motifs
Nouveauté produit (2)
Autre (1)
Remontage (1)
Statistiques de la région
Afficher pour :
David ANDRE
Données pour : David ANDRE
Visites par visiteur
2
1
1
0
0
David ANDRE
Évolution mensuelle
03/2026
06/2026
Médicaments les + présentés
Aucune donnée médicaments
Répartition des motifs
Autre (1)
Nouveauté produit (1)

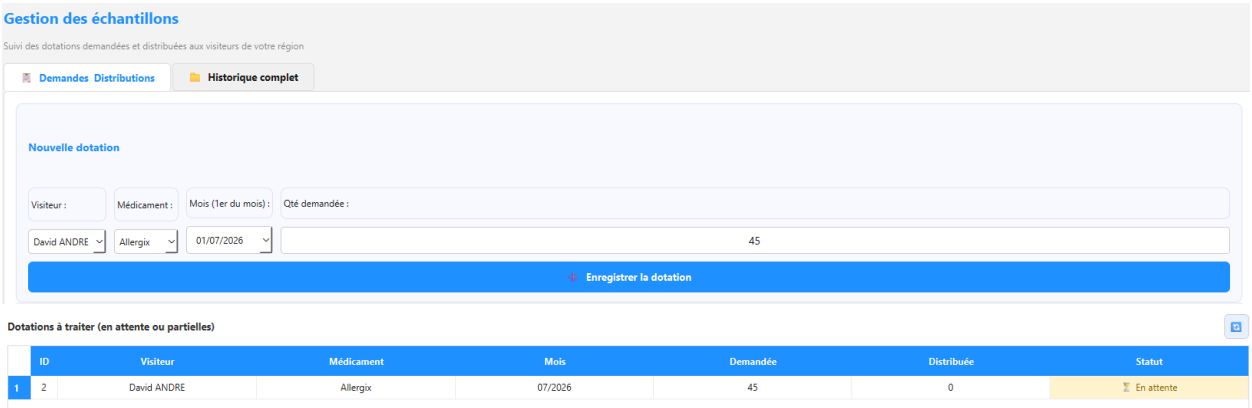
6.5 Gestion des échantillons

Fichier concerné : **`views/pages/delegue/echantillons_page.py`**

Cette page répond à l'exigence légale et comptable de suivi à l'unité près des dotations d'échantillons. Elle s'appuie sur la table dotation existante (mois, qtedemandee, qtedistribuee, visiteur_id, medicament_id) et propose une organisation en deux onglets.

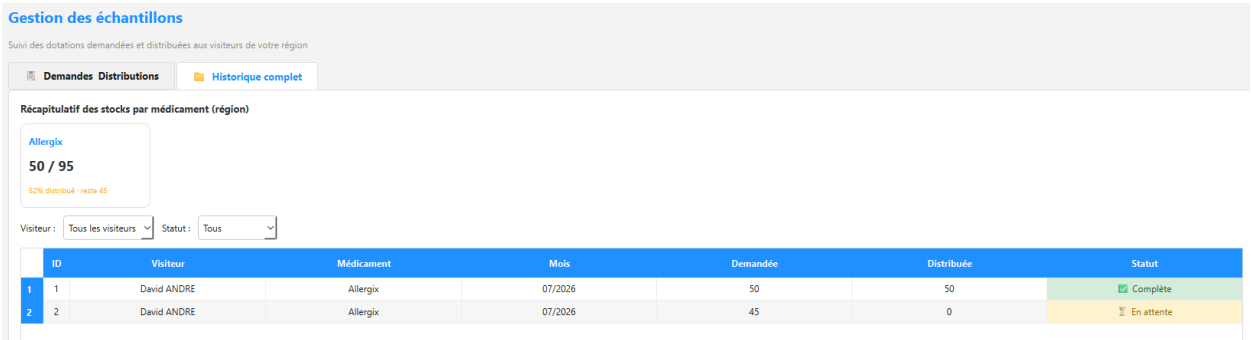
6.5.1 Onglet « Demandes & Distributions »

- Formulaire de création d'une nouvelle dotation : sélection du visiteur, du médicament, du mois concerné et de la quantité demandée
- Tableau des dotations en attente ou partielles (qtedistribuee < qtedemandee)
- Au clic sur une ligne, un panneau de distribution rapide apparaît, pré-rempli avec la quantité restante à distribuer
- Contrôle métier : la quantité distribuée ne peut jamais dépasser la quantité demandée



6.5.2 Onglet « Historique complet »

- Récapitulatif visuel en haut de page : une carte par médicament, affichant le total distribué/demandé pour toute la région, le taux de distribution et l'écart restant
- Filtres combinés par visiteur et par statut (en attente / partielle / complète)
- Tableau complet de toutes les dotations de la région, avec code couleur sur la colonne statut



6.5.3 Statuts calculés

Le statut de chaque dotation n'est pas stocké en base mais calculé dynamiquement à l'affichage, à partir des deux quantités :

- En attente — qtedistribuee = 0
- Partielle — 0 < qtedistribuee < qtedemandee
- Complète — qtedistribuee = qtedemandee

6.6 Modifications transverses liées au module

Le développement du module Délégué a nécessité plusieurs évolutions sur des pages déjà existantes du module Visiteur, afin de garantir la cohérence des données affichées dans les statistiques et l'historique.

6.6.1 Nouvelle table de liaison rapport_visite_medicament

Création d'une table de jonction pour modéliser la relation plusieurs-à-plusieurs entre un compte-rendu et les médicaments présentés lors de la visite, avec la quantité d'échantillons donnés pour chacun :

```
rapport_visite_medicament (rapport_id, medicament_id, nb_echantillons)
```

6.6.2 Formulaire de compte-rendu (comptes_rendues_page.py)

- Ajout d'une liste scrollable de médicaments cochables dans le panneau de droite
- Chaque médicament coché révèle un champ de saisie de quantité d'échantillons donnés
- Pré-remplissage automatique de ces cases en mode édition d'un compte-rendu existant

6.6.3 Page Dashboard (dashboard_page.py)

Ajout d'un rafraîchissement automatique des indicateurs (showEvent) à chaque retour sur la page, pour que les chiffres reflètent immédiatement les comptes-rendus qui viennent d'être saisis.

6.7 Synthèse de conformité

Le tableau ci-dessous met en correspondance chaque exigence des spécifications fonctionnelles avec son implémentation dans l'application.

Exigence du cahier des charges	Implémentation
Vue d'activité de sa région	✓ `activite_region_page.py` — tableau + filtre + détail
Statistiques graphiques par visiteur	✓ `stats_region_page.py` — 4 graphiques PyQt6-Charts
Détail d'un visiteur (zoom collaborateur)	✓ Filtre combo dans `stats_region_page.py`
Gestion des échantillons (demandé/distribué, suivi légal)	✓ `echantillons_page.py` — 2 onglets, contrôle métier, récap
Conservation des fonctionnalités Visiteur	✓ Sidebar dynamique dans `main_window.py`

Le module Délégué régional est considéré comme fonctionnellement complet au regard des spécifications fournies.

7. Module Responsable de secteur

Le Responsable de secteur supervise l'ensemble des régions et des collaborateurs rattachés à son périmètre. Contrairement au délégué, il n'a pas de rôle de visiteur médical actif ; son interface est entièrement centrée sur la supervision, les statistiques globales et la gestion des ressources humaines de son secteur.

7.1 Périmètre fonctionnel couvert

- Comptes-rendus du secteur — consultation de tous les rapports de visite, filtrés par région, visiteur et période
- Statistiques du secteur — 4 graphiques PyQt6-Charts avec filtre région/visiteur
- Statistiques par visiteur — zoom individuel sur un collaborateur (4 graphiques dédiés)
- Contrôle des stocks — surveillance de la cohérence entre dotations demandées et distribuées
- Collaborateurs — consultation et modification des informations RH des visiteurs et délégués du secteur

7.2 Architecture et navigation

Comme pour le module Délégué, la sidebar de la fenêtre principale (main_window.py) est construite dynamiquement selon le rôle de l'utilisateur. Pour un responsable, un unique bloc « Espace Responsable » apparaît, sans section visiteur, car ce rôle n'inclut pas de fonctionnalités de terrain. Les quatre pages exclusives sont : Comptes-rendus, Statistiques secteur, Stats par visiteur, Stocks et Collaborateurs.

7.2.1 Nouveaux fichiers créés

Fichier	Rôle
`views/pages/responsable/__init__.py`	Marqueur de package Python (vide)
`views/pages/responsable/comptes_rendus_secteur_page.py`	Page de consultation des CR de tout le secteur
`views/pages/responsable/stats_secteur_page.py`	Page des 4 graphiques PyQt6-Charts (niveau secteur)
`views/pages/responsable/stats_visiteur_page.py`	Page de statistiques individuelles par visiteur (4 graphiques)
`views/pages/responsable/stocks_secteur_page.py`	Page de contrôle des stocks d'échantillons (lecture seule)
`views/pages/responsable/collaborateurs_page.py`	Page de gestion des profils RH des collaborateurs

7.3 Comptes-rendus du secteur

Fichier concerné : `views/pages/responsable/comptes_rendus_secteur_page.py`

Cette page permet au responsable de consulter l'intégralité des comptes-rendus de visite produits par tous les visiteurs de son secteur. Elle interroge la base en jointure sur les tables rapport_visite, visiteur, region et praticien, filtrée sur le secteur_id récupéré dynamiquement depuis la region_id du compte connecté.

- Tableau principal : colonnes ID, Visiteur, Région, Date, Praticien, Motif — tri par date décroissante
- Filtres combinés : combo Région (avec rechargement dynamique du combo Visiteur), plage de dates (QDateEdit), boutons « Appliquer » et « Tout afficher »
- Compteur de résultats en temps réel (« N rapport(s) affiché(s) »)
- Panneau de détail à droite (QSplitter 66/34) : au clic sur une ligne, affichage complet du compte-rendu (visiteur, région, praticien, dates, motif, coefficient de confiance, bilan, notes de concurrence)

7.4 Statistiques du secteur

Fichier concerné : `views/pages/responsable/stats_secteur_page.py`

Cette page offre une vision graphique globale de l'activité du secteur via quatre graphiques PyQt6-Charts organisés en grille 2×2 scrollable. Un filtre hiérarchique (combo Région → combo Visiteur) pilote simultanément tous les graphiques.

7.4.1 Les quatre graphiques (PyQt6-Charts)

- Visites par région / visiteur — QBarSeries affichant le nombre de CR par région ; en mode zoom visiteur, bascule automatiquement en barres par mois
- Évolution mensuelle — QSplineSeries de l'ensemble des visites du secteur mois par mois
- Médicaments les plus présentés — QBarSeries groupées (fréquence de présentation + total échantillons) via la table offrir ; top 8
- Répartition des motifs de visite — QPieSeries avec le motif dominant mis en valeur (slice explodée)

7.5 Statistiques par visiteur

Fichier concerné : `views/pages/responsable/stats_visiteur_page.py`

Cette page permet un zoom individuel sur un visiteur sélectionné dans un combo déroulant. Elle charge tous les visiteurs du secteur (rôle visiteur uniquement) et génère dynamiquement quatre graphiques de performance à chaque changement de sélection. Un bandeau d'information rappelle le nom, la région et la date d'embauche du collaborateur sélectionné.

- Activité mensuelle — QSplineSeries du nombre de visites effectuées par ce visiteur, mois par mois
- Répartition des motifs — QPieSeries des motifs de visite propres à ce collaborateur
- Praticiens les plus visités — QBarSeries top 8 des médecins rencontrés
- Médicaments les plus présentés — QBarSeries groupées (fois présenté + échantillons totaux), top 8

7.6 Contrôle des stocks d'échantillons

Fichier concerné : `views/pages/responsable/stocks_secteur_page.py`

Cette page en lecture seule donne au responsable une vue de cohérence entre les dotations demandées par les délégués et les quantités effectivement distribuées aux visiteurs. Elle s'appuie sur la table dotation (même source que le module Délégué) mais couvre l'ensemble du secteur. Elle est organisée en deux onglets.

7.6.1 Onglet « Vue globale du secteur »

- Cartes de récapitulatif par région (défilables horizontalement) : nom de la région, ratio distribué/demandé, pourcentage coloré (vert ≥100 %, orange ≥50 %, rouge <50 %), compteur d'anomalies
- Filtres combinés : région (avec rechargement du combo visiteur), visiteur, statut (En attente / Partielle / Complète / Anomalie)
- Tableau : ID, Région, Visiteur, Médicament, Mois, Qté demandée, Qté distribuée, Statut (coloré)

7.6.2 Onglet « Anomalies & alertes »

Cet onglet isole automatiquement les dotations présentant une incohérence : quantité distribuée supérieure à la demande (anomalie) ou dotation restée entièrement non distribuée (en attente). Un message récapitulatif en haut de page signale immédiatement le nombre d'anomalies détectées ou confirme l'absence de problème.

7.6.3 Statuts calculés

Le statut de chaque dotation est calculé à l'affichage, non stocké en base, à partir des deux quantités :

- En attente — qtedistribuee = 0
- Partielle — 0 < qtedistribuee < qtedemandee
- Complète — qtedistribuee = qtedemandee
- Anomalie — qtedistribuee > qtedemandee

7.7 Gestion des collaborateurs

Fichier concerné : `views/pages/responsable/collaborateurs_page.py`

Cette page donne au responsable une vision RH de l'ensemble de ses équipes. Elle liste tous les visiteurs et délégués du secteur dans un tableau (panneau gauche), avec un formulaire de consultation et d'édition des coordonnées dans le panneau droit (QSplitter).

- Tableau principal : colonnes ID, Nom, Prénom, Rôle, Région — filtré par rôle (Tous / Visiteur médical / Délégué régional), région et recherche textuelle par nom
- Panneau de profil (droite) : informations d'identité en lecture seule (ID, nom, prénom, login, rôle, date d'embauche, région) et coordonnées modifiables (adresse, code postal, ville)
- Validation avant enregistrement : code postal obligatoire et format 5 chiffres vérifié — message d'erreur explicite en cas d'échec ; confirmation visuelle (QMessageBox) après sauvegarde réussie
- Mise à jour en base via requête préparée UPDATE visiteur SET adresse, cp, ville WHERE id — cache local rafraîchi immédiatement après

7.8 Synthèse de conformité

Exigence du cahier des charges	Implémentation
Consultation des CR de tout le secteur	✓ comptes_rendus_secteur_page.py — tableau + filtres + détail
Statistiques graphiques secteur	✓ stats_secteur_page.py — 4 graphiques PyQt6-Charts, filtre région/visiteur
Statistiques individuelles par visiteur	✓ stats_visiteur_page.py — 4 graphiques par collaborateur sélectionné
Contrôle des stocks secteur (lecture seule)	✓ stocks_secteur_page.py — 2 onglets, récap par région, alertes anomalies
Gestion RH des collaborateurs	✓ collaborateurs_page.py — consultation + édition adresse/CP/ville

Le module Responsable de secteur est considéré comme fonctionnellement complet au regard des spécifications fournies.