PAPPE web - SIO SLAM

PharmaCantine

Application web de gestion de la cantine d'entreprise





Emma ZHAO - BTS SIO - 2026
Institution Des Chartreux, 38 rue Pierre Dupont - 69004 Lyon

Table des matières

1. Contexte 2
2. Présentation du projet 2
Objectifs 2
Périmètre fonctionnel 2
3. Contraintes du projet 3
3.1 Ergonomie 3
3.2 Environnement de développement 3
3.3 Accès et rôles 3
3.4 Contraintes métier 3
4. Architecture technique 4
4.1 Architecture globale 4
4.2 Arborescence du projet 4
4.3 Routes de l'application 4
Routes publiques (sans authentification) 4
Routes communes (tous rôles authentifiés) 4
Espace employé (can:employe) 4
Espace administrateur (can:admin) 5
Espace cantine - admin + cuisinier (can:manage-cantine) 5
5. Base de données 6
5.1 Modèle relationnel 6
5.2 Description des tables 6
5.3 Soft delete et statuts 6
6. Sécurité et authentification 8
6.1 Authentification 8
6.2 Gestion des rôles - Laravel Gates 8
6.3 Middleware de journalisation 8
6.4 Protection CSRF et sessions 8
6.5 Déploiement conteneurisé 8
7. Réalisation du projet 9
7.1 Page de connexion 9
7.2 Dashboard administrateur 9
7.3 Gestion des utilisateurs (admin) 10
7.4 Statistiques (admin) 11
7.5 Gestion des plats (admin + cuisinier) 11
7.6 Gestion des menus (admin + cuisinier) 13
..... 13
7.7 Gestion des réservations côté cantine (admin + cuisinier) 14
7.8 Dashboard cuisinier 14
7.9 Planning hebdomadaire (employé) 15
7.10 Mes réservations (employé) 16
7.11 Paramètres du compte 17
8. Versioning et déploiement 18
9. Conclusion 18

1. Contexte

Le laboratoire Galaxy Swiss Bourdin (GSB) est une entreprise pharmaceutique issue de la fusion entre le géant américain Galaxy, spécialisé dans le secteur des maladies virales, et le conglomérat européen Swiss Bourdin. Cette entité regroupe plusieurs centaines de collaborateurs répartis sur différents sites en France.

Dans ce contexte, la gestion de la cantine d'entreprise se faisait jusqu'alors de manière informelle : aucun suivi centralisé des réservations, difficultés pour les cuisiniers à anticiper les quantités à préparer, et absence de vision claire sur la fréquentation des services de déjeuner et de dîner.

L'entreprise souhaite moderniser ce processus en déployant une application web accessible depuis le réseau interne, permettant à chaque collaborateur de consulter les menus de la semaine et de réserver son repas en quelques clics.

2. Présentation du projet

Le projet PharmaCantine consiste à développer une application web de gestion de la cantine interne de GSB. Elle permet aux employés de consulter le planning hebdomadaire des menus et de réserver leur repas, aux cuisiniers de gérer les plats et de composer les menus quotidiens par service (déjeuner et dîner), et aux administrateurs de piloter les comptes utilisateurs et de consulter les statistiques de fréquentation.

Objectifs

- Permettre aux employés de consulter le planning hebdomadaire des menus
- Permettre la réservation et l'annulation de repas par service (déjeuner / dîner)
- Permettre aux cuisiniers de gérer le catalogue de plats (CRUD avec soft delete)
- Permettre aux cuisiniers de composer et modifier les menus quotidiens par service
- Permettre aux administrateurs de gérer les comptes utilisateurs
- Fournir un tableau de bord statistique (fréquentation, taux d'annulation, plat le plus populaire)
- Tracer toutes les actions des utilisateurs via un middleware de journalisation

Périmètre fonctionnel

L'application couvre trois rôles distincts :

Rôle	Fonctionnalités
Employé	Consulter le planning des menus, réserver un créneau (déjeuner/dîner), consulter et annuler ses réservations, modifier son profil et mot de passe
Cuisinier	CRUD plats (avec filtres par catégorie), création et modification des menus quotidiens par service, consultation et annulation de toutes les réservations
Administrateur	Toutes les fonctionnalités cuisinier + CRUD utilisateurs, tableau de bord statistique complet

3. Contraintes du projet

3.1 Ergonomie

L'application doit être accessible à des collaborateurs non techniques. L'interface adopte un design professionnel avec Tailwind CSS, une palette de couleurs cohérente (bleu primaire #1A5EA8 sur fond clair) et une typographie lisible. Chaque rôle dispose d'une sidebar de navigation adaptée à ses besoins spécifiques. Les actions destructives (suppression, annulation) sont systématiquement précédées d'une page de confirmation dédiée.

3.2 Environnement de développement

Technologie	Usage
PHP 8.2 / Laravel 12	Framework backend - MVC, routing, ORM Eloquent, middleware
Blade	Moteur de templates HTML côté serveur
Tailwind CSS 4 (Vite)	Framework CSS utilitaire
MariaDB 11 (Docker)	Base de données relationnelle conteneurisée (base : pharmanutri)
Docker / PHP-FPM	Environnement de développement conteneurisé (port 8000)
opcodesio/log-viewer	Interface web de visualisation des logs Laravel

3.3 Accès et rôles

L'application ne propose pas d'inscription autonome. Les comptes sont créés exclusivement par l'administrateur. Trois rôles sont définis dans la base de données : employe, cuisinier, admin. La gestion des droits est assurée par des Laravel Gates déclarées manuellement dans AppServiceProvider. Après connexion, l'utilisateur est redirigé automatiquement vers son dashboard selon son rôle.

3.4 Contraintes métier

- Un seul menu actif (non annulé) est autorisé par date et par service (déjeuner ou dîner).
- Un employé ne peut pas re-réserver un repas pour un menu qu'il a préalablement annulé.
- Un compte désactivé (statut = inactif) ne peut plus se connecter, même avec un mot de passe correct.

4. Architecture technique

4.1 Architecture globale

L'architecture suit le patron MVC (Modèle-Vue-Contrôleur) natif de Laravel. L'application est full server-side : les pages HTML sont rendues côté serveur par le moteur de templates Blade. Il n'y a pas d'API REST dédiée ni de framework frontend réactif..

4.2 Arborescence du projet

Dossier / Fichier	Description
app/Http/Controllers/	6 contrôleurs : AuthController, UserController, PlatController, MenuController, ReservationController, ParametreController
app/Models/	7 modèles Eloquent : User, Plat, Menu, MenuPlat (pivot), Reservation, Service
app/Http/Middleware/	LogUserActions - journalisation des requêtes (POST/PUT/PATCH/DELETE)
app/Providers/AppServiceProvider.php	Déclaration des Gates d'autorisation (admin, cuisinier, employe, manage-cantine)
database/migrations/	Migrations : users, plats (soft delete), menus, menus_plats, reservations, services
resources/views/	Templates Blade organisés par rôle (dashboard_admin, dashboard_cuisinier, employe) et fonctionnalité (plat_*, menu_*, reservation_*)
resources/views/layouts.blade.php	Layout principal : sidebar, header, imports CDN (Chart.js, Alpine.js, Tailwind)
resources/views/partials/	Composants réutilisables : header, sidebar
routes/web.php	Toutes les routes groupées par middleware (auth, can:role)
docker-compose.yml	Orchestration des services php, web (Nginx) et db (MariaDB)

4.3 Routes de l'application

Routes publiques (sans authentification)

Méthode	URL	Action
GET	/	Affiche le formulaire de connexion
POST	/login	Traitement de la connexion, redirection selon le rôle

Routes communes (tous rôles authentifiés)

Méthode	URL	Action
POST	/logout	Déconnexion, invalidation de session
GET	/dashboard	Redirection vers le dashboard du rôle connecté
GET	/parametres	Page profil et paramètres
PATCH	/parametres/statut	Mise à jour du statut du compte
PATCH	/parametres/password	Changement de mot de passe sécurisé

Espace employé (can:employe)

Méthode	URL	Action
GET	/employe	Dashboard - réservations actives de l'employé
GET	/employe/planning/{date?}	Planning hebdomadaire des menus (date optionnelle)
GET	/employe/reservations	Historique complet de toutes les réservations
POST	/reservation/create	Créer une réservation
POST	/reservation/cancel	Annuler une réservation

Espace administrateur (can:admin)

Méthode	URL	Action
GET	/admin	Dashboard admin (stats rapides + dernières réservations)
GET	/admin/users	Liste paginée des utilisateurs
GET/POST	/admin/users/create	Formulaire de création / Enregistrement
GET/PUT	/admin/users/update/{id}	Formulaire de modification / Sauvegarde
GET/DELETE	/admin/users/delete/{id}	Confirmation de suppression / Suppression
GET	/admin/statistiques	Tableau de bord statistique complet

Espace cantine - admin + cuisinier (can:manage-cantine)

Plats :

Méthode	URL	Action
GET	/cantine/plats	Liste tous les plats avec compteurs
GET/POST	/cantine/plats/create	Formulaire / Enregistrement d'un plat
GET/PUT	/cantine/plats/update/{id}	Modification d'un plat
GET/DELETE	/cantine/plats/delete/{id}	Confirmation / Soft delete
GET	/cantine/plats/{Entrees ou Plats ou Desserts}	Filtre par catégorie

Menus :

Méthode	URL	Action
GET	/cantine/menus	Liste tous les menus avec statut
GET/POST	/cantine/menus/create	Formulaire / Enregistrement d'un menu
GET/PUT	/cantine/menus/update/{id}/{service}	Modification d'un menu par service
GET/PUT	/cantine/menus/delete/{id}	Confirmation / Annulation (statut = annule)
GET	/cantine/menus/{actifs|annules|archives}	Filtre par statut

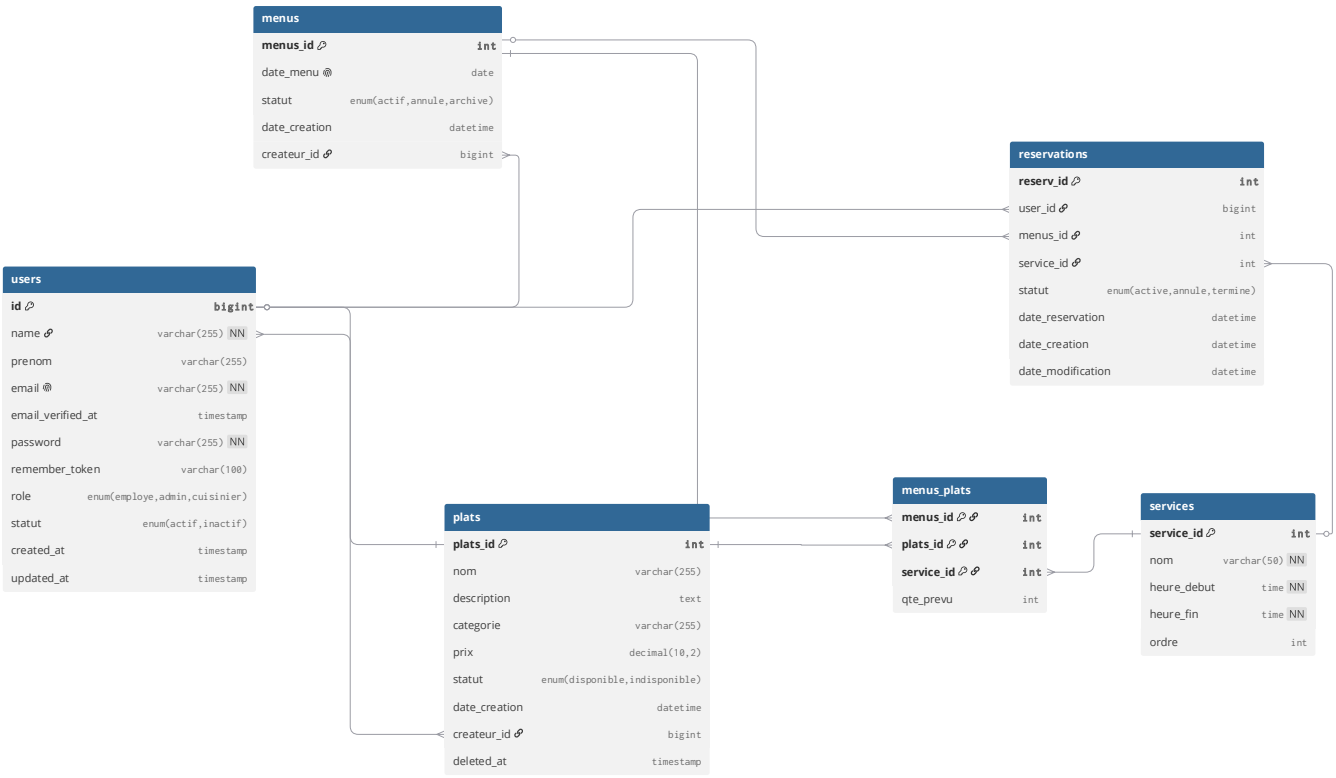
Réservations cantine :

Méthode	URL	Action
GET	/cantine/reservations	Liste toutes les réservations
GET/POST	/cantine/reservations/store	Formulaire / Réserver pour un employé
PATCH	/cantine/reservations/cancel/{id}	Annuler une réservation

5. Base de données

5.1 Modèle relationnel

La base de données MariaDB (pharmanutri) est composée de 6 tables métier. Le schéma a été conçu pour gérer les menus, les réservations et les utilisateurs.



5.2 Description des tables

Table	Description et colonnes principales
users	id, name, prenom, email (unique), password (bcrypt), role (employe/cuisinier/admin), statut (actif/inactif), created_at, updated_at
plats	plats_id, nom, description, categorie (Entrée/Plat/Dessert), prix, statut (disponible/indisponible), createur_id, deleted_at (soft delete)
menus	menus_id, date_menu (unique), statut (actif/annule/archive), date_creation, createur_id
menus_plats	Table pivot (Menu - Plat – Service). Clé composé (menus_id, plats_id, service_id). Stocke la quantité prévue (qte_prevu).
reservations	reserv_id, user_id, menus_id, service_id, statut (active/annule/termine), date_reservation, date_creation, date_modification
services	service_id, nom, heure_debut, heure_fin, ordre. Deux entrées fixes : Déjeuner (12h-14h) et Dîner (19h-21h).

5.3 Soft delete et statuts

Les suppressions de plats sont logiques et non physiques : le champ deleted_at est renseigné via le softdelete de Laravel. Cela préserve l'intégrité référentielle avec les menus existants et permet de conserver l'historique.

Les menus ne sont jamais supprimés physiquement : ils passent au statut annule ou archive. L'annulation d'un menu déclenche automatiquement l'annulation de toutes ses réservations actives via une requête UPDATE en transaction (rollback en cas d'erreur). Les utilisateurs sont désactivés via le champ statut = inactif sans suppression physique.

5.4 Événements planifiés (SQL Events)

Deux événements SQL sont définis directement au niveau de MariaDB et s'exécutent automatiquement chaque jour. Cela délègue la maintenance des statuts à la base de données plutôt qu'à l'application, garantissant ainsi que les mises à jour ont lieu même si l'application n'est pas utilisé.

L'event scheduler est activé via le paramètre --event_scheduler=ON dans le service db du docker-compose.yml.

archive_menus_event Chaque jour, cet événement passe au statut archive tous les menus dont le statut est encore actif et dont la date_menu est antérieure à la date du jour. Cela évite que d'anciens menus restent visibles comme actifs dans les listes de gestion.

```
CREATE EVENT archive_menus_event
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP
DO
BEGIN
   UPDATE menus
   SET statut = 'archive'
   WHERE statut = 'actif'
   AND date_menu < CURDATE();
END
```

termine_reservations_event Chaque jour, cet événement passe au statut termine toutes les réservations dont le statut est encore active et dont la date_reservation est antérieure au moment présent.

```
CREATE EVENT termine_reservations_event
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP
DO
BEGIN
   UPDATE reservations
   SET statut = 'termine'
   WHERE statut = 'active'
   AND date_reservation < NOW();
END
```

6. Sécurité et authentification

6.1 Authentification

L'authentification repose sur le système natif de Laravel (Auth::attempt / Auth::logout), sans package tiers. À la connexion, après vérification du mot de passe, le statut du compte est contrôlé : si statut = inactif, la session est immédiatement invalidée avec un message d'erreur. La session est régénérée à chaque connexion et invalidée à la déconnexion.

6.2 Gestion des rôles - Laravel Gates

Les autorisations sont gérées manuellement via Gate::define() dans AppServiceProvider, sans package tiers.

Gate	Condition d'accès	Rôles
admin	user->role === 'admin'	admin uniquement
cuisinier	user->role === 'cuisinier'	cuisinier uniquement
employe	user->role === 'employe'	employé uniquement
manage-cantine	in_array(role, ['admin','cuisinier'])	admin + cuisinier

Les routes sont protégées par les middlewares can:admin, can:cuisinier, can:employe et can:manage-cantine. Les vues Blade utilisent les directives @can / @cannot pour conditionner l'affichage des éléments de navigation selon le rôle connecté.

6.3 Middleware de journalisation

Un middleware custom LogUserActions est appliqué à toutes les routes authentifiées. Il s'exécute après la réponse et enregistre uniquement les requêtes mutatives (POST, PUT, PATCH, DELETE) des utilisateurs connectés. Les données loggées incluent : user_id, email, rôle, méthode HTTP, URL complète, adresse IP, user-agent et code de réponse HTTP. La route /dashboard est exclue de la journalisation.

6.4 Protection CSRF et sessions

La protection CSRF de Laravel est active sur toutes les routes web. Le token est injecté via @csrf dans tous les formulaires Blade. Pour les appels Axios asynchrones des modales de réservation, le token CSRF est transmis manuellement dans le header X-CSRF-TOKEN. Les sessions sont stockées en base de données (table sessions, durée 120 minutes, cookie httpOnly avec SameSite=Lax).

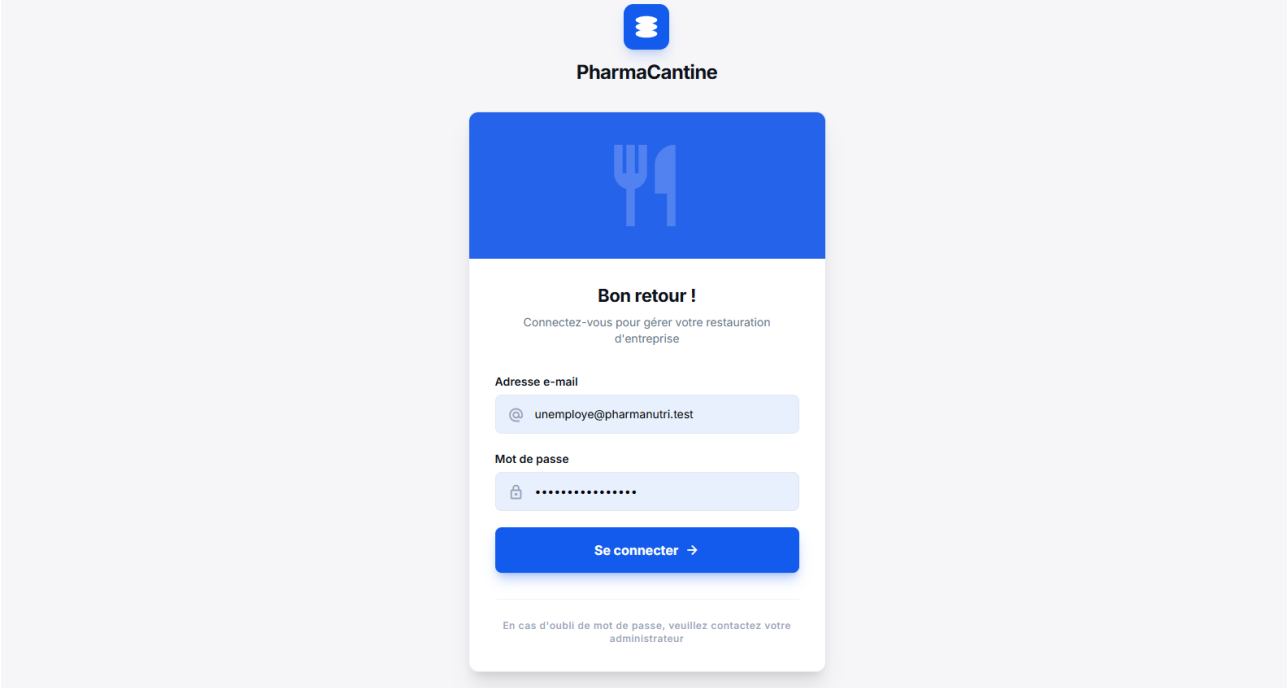
6.5 Déploiement conteneurisé

L'application est conteneurisée via Docker Compose avec trois services : PHP 8.3-FPM pour l'exécution de Laravel, Nginx stable-alpine en reverse proxy sur le port 8000, et MariaDB 11 pour la base de données. Les variables sensibles (APP_KEY, credentials BDD, JWT_SECRET) sont injectées via le fichier .env qui n'est pas versionné dans le dépôt Git (.gitignore).

7. Réalisation du projet

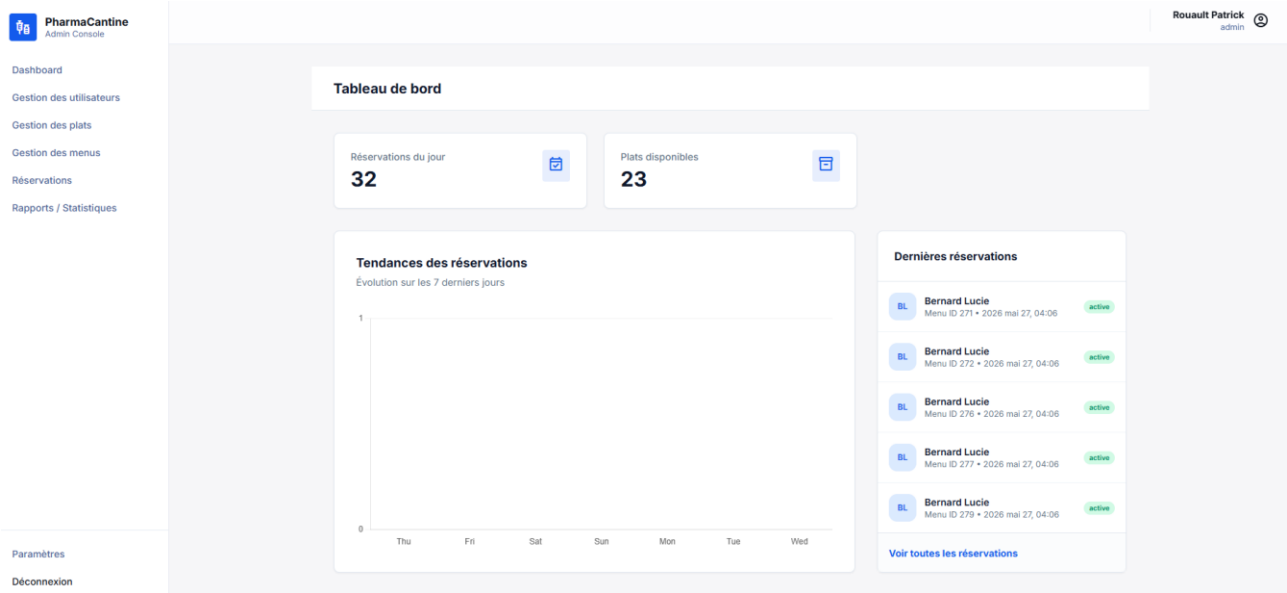
7.1 Page de connexion

La page de connexion est commune à tous les rôles. Elle comporte un formulaire avec validation des champs email et mot de passe. Après authentification, l'utilisateur est redirigé automatiquement vers le dashboard correspondant à son rôle.



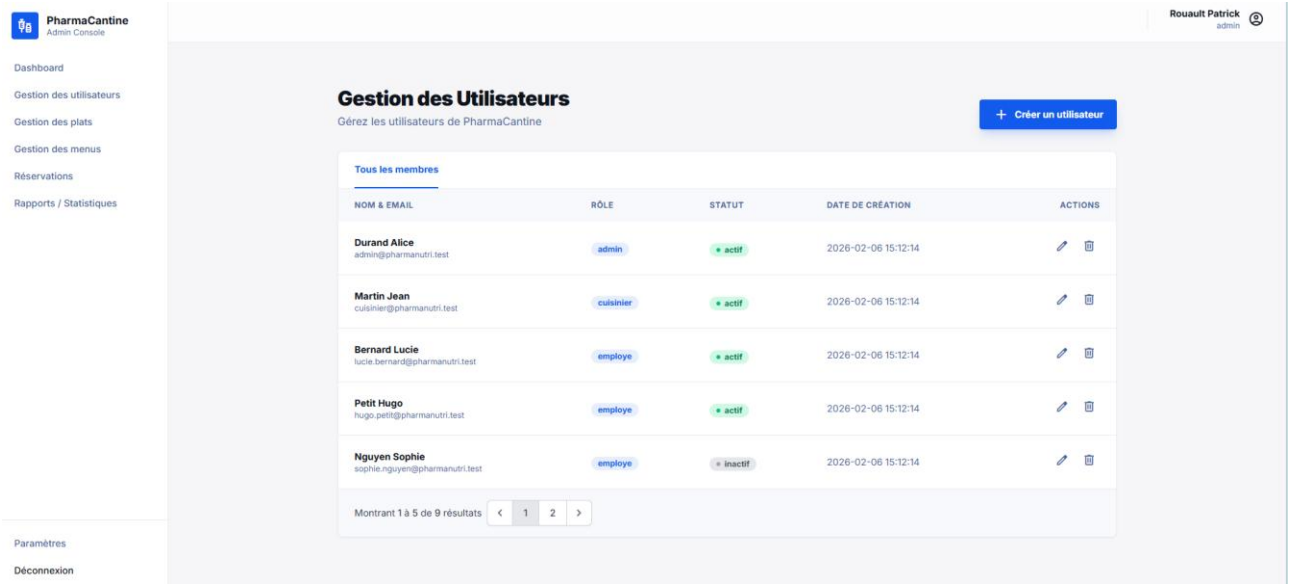
7.2 Dashboard administrateur

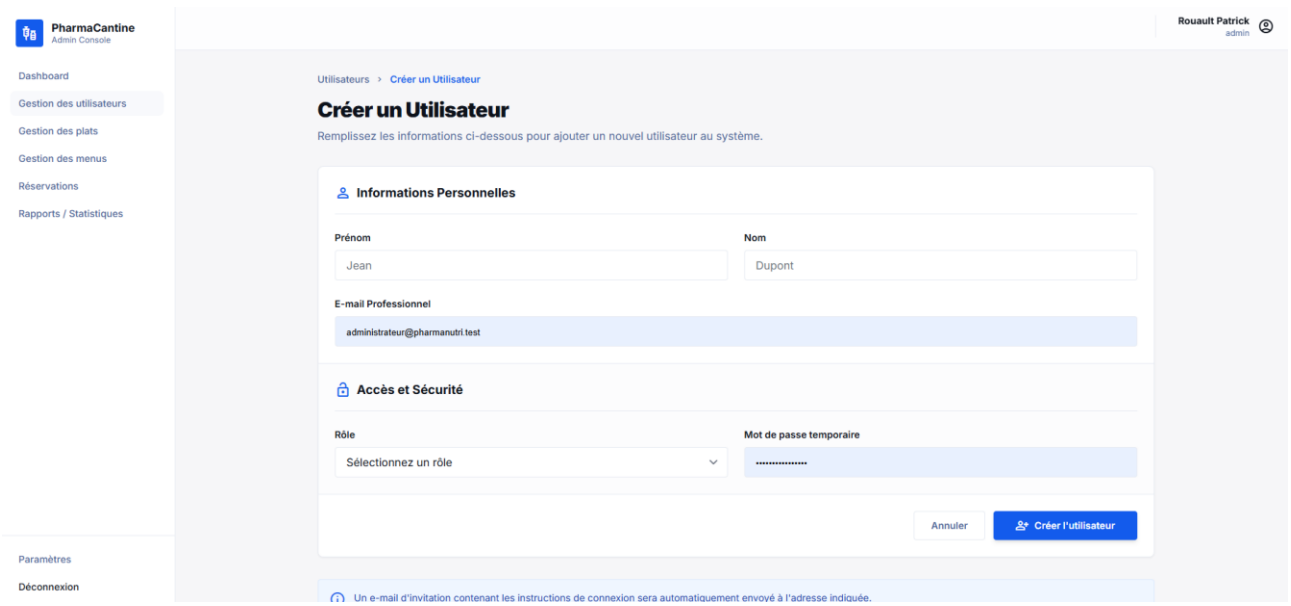
Le dashboard admin affiche en temps réel les 5 dernières réservations actives, les compteurs globaux (nombre de plats, nombre de réservations), ainsi qu'un graphique représentant l'évolution des réservations sur les 7 derniers jours. C'est le point d'entrée central de l'espace administrateur.



7.3 Gestion des utilisateurs (admin)

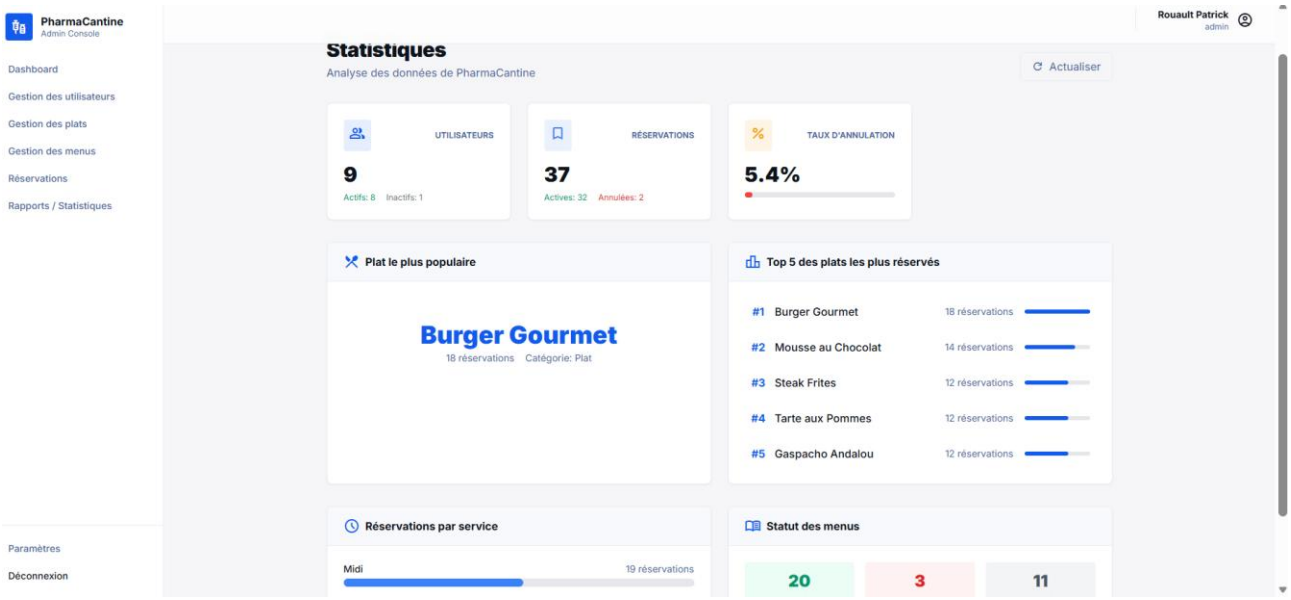
L'administrateur accède à la liste paginée de tous les utilisateurs (5 par page). Il peut créer un compte en définissant le prénom, le nom, l'email, le rôle (employe/cuisinier/admin) et le mot de passe initial. La modification permet de changer toutes ces informations ainsi que le statut du compte via une case à cocher. Chaque suppression est précédée d'une page de confirmation dédiée.





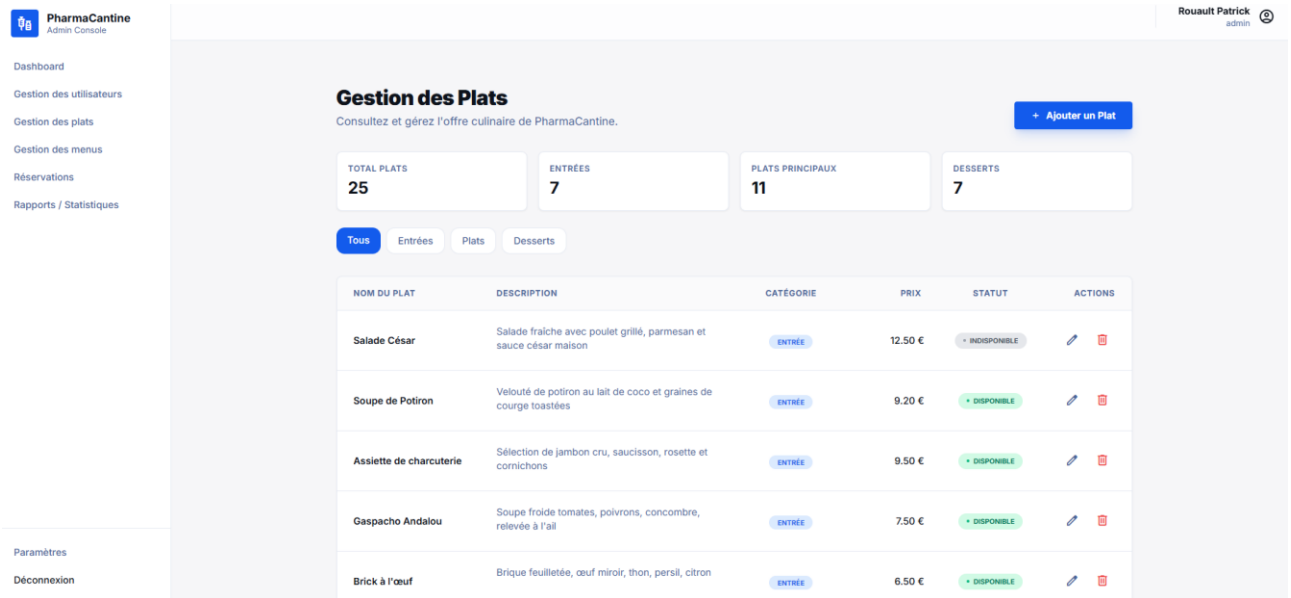
7.4 Statistiques (admin)

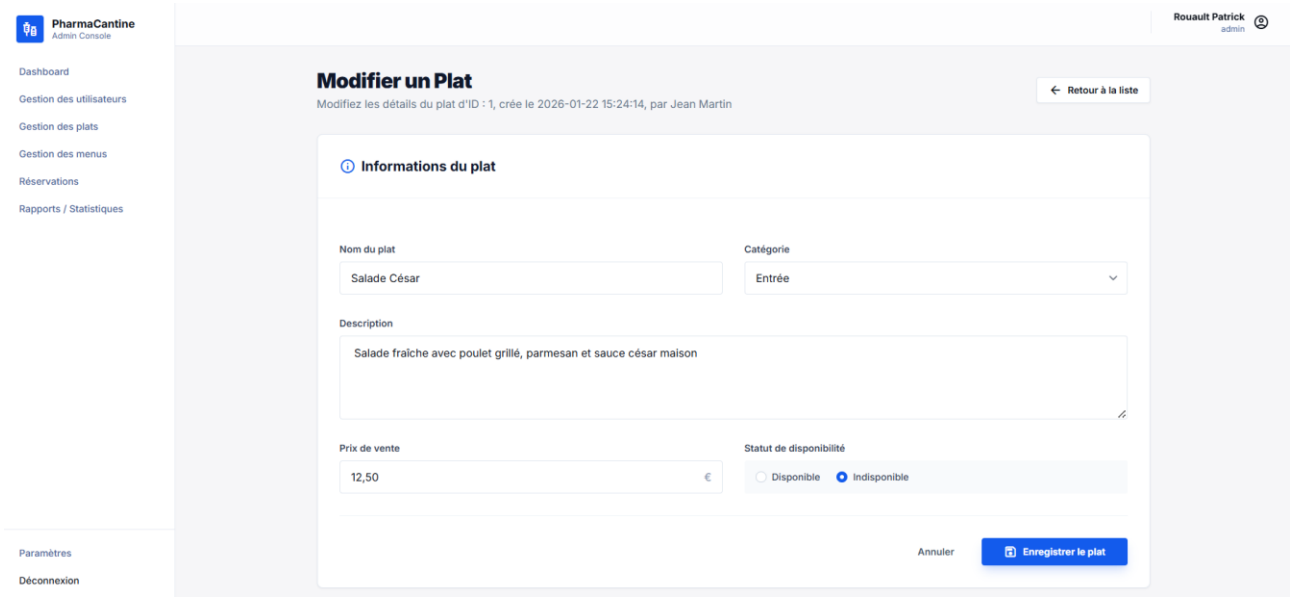
La page statistiques fournit une vue d'ensemble complète de l'activité de la cantine : nombre total d'utilisateurs par statut (actif/inactif), répartition des réservations (active/annulée/terminée), répartition midi/soir, et classement des 5 plats les plus réservés.



7.5 Gestion des plats (admin + cuisinier)

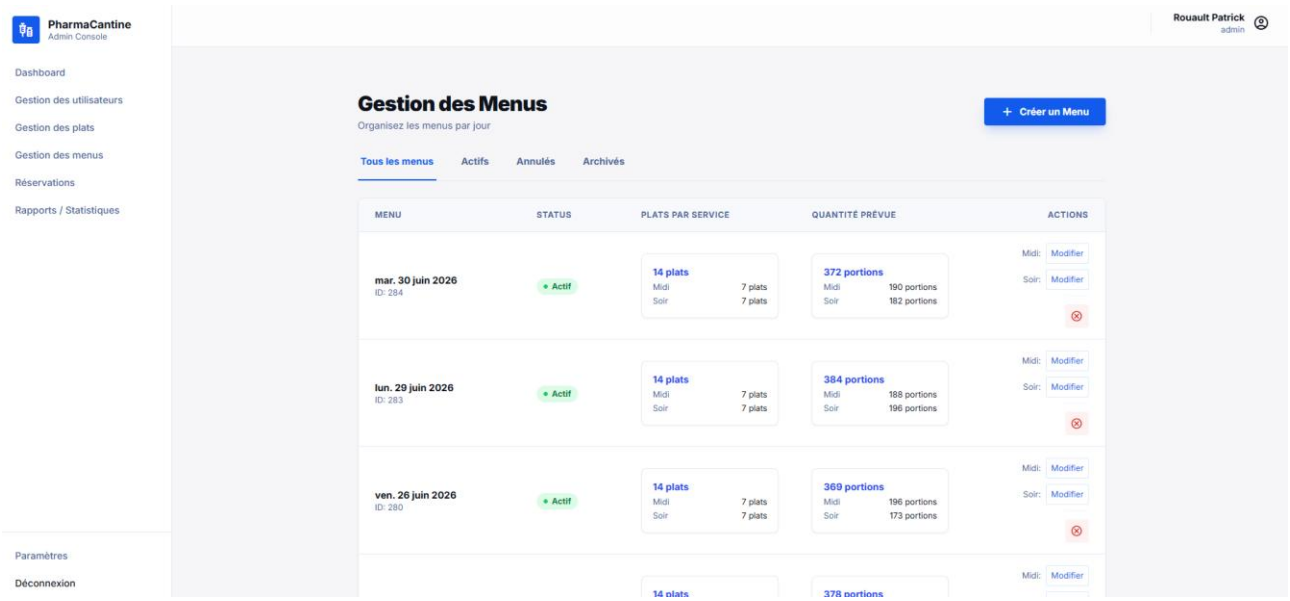
L'espace de gestion des plats présente le catalogue complet avec des filtres par catégorie (Entrées, Plats, Desserts) et des compteurs dynamiques. Chaque fiche affiche le nom, la catégorie, le prix et le statut du plat (disponible/indisponible). Le formulaire de création et de modification valide le nom, la description, la catégorie, le prix (0 à 100€) et le statut. La suppression est un soft delete : le plat reste en base avec un deleted_at renseigné et n'apparaît plus dans les listes.

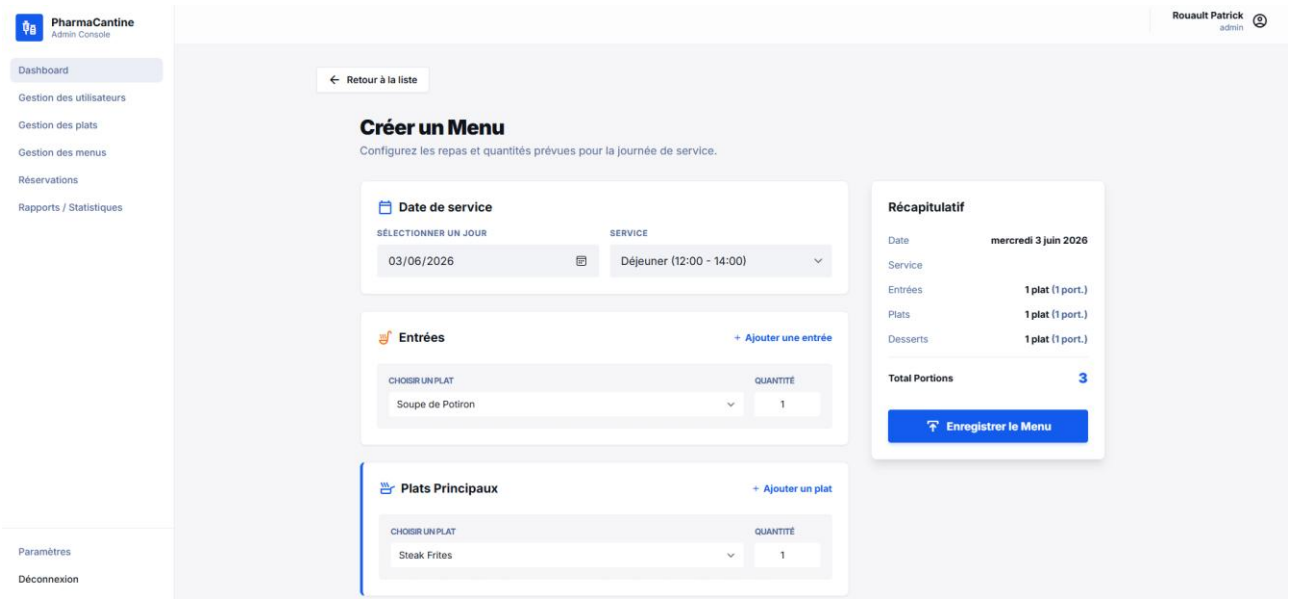



PharmaCantine
Admin Console
Dashboard
Gestion des utilisateurs
Gestion des plats
Gestion des menus
Réservations
Rapports / Statistiques
Paramètres
Déconnexion
Rouault Patrick
admin
Modifier un Plat
Modifiez les détails du plat d'ID : 1, crée le 2026-01-22 15:24:14, par Jean Martin
Retour à la liste
Informations du plat
Nom du plat
Salade César
Catégorie
Entrée
Description
Salade fraîche avec poulet grillé, parmesan et sauce césar maison
Prix de vente
12,50
€
Statut de disponibilité
Disponible
Indisponible
Annuler
Enregistrer le plat

7.6 Gestion des menus (admin + cuisinier)

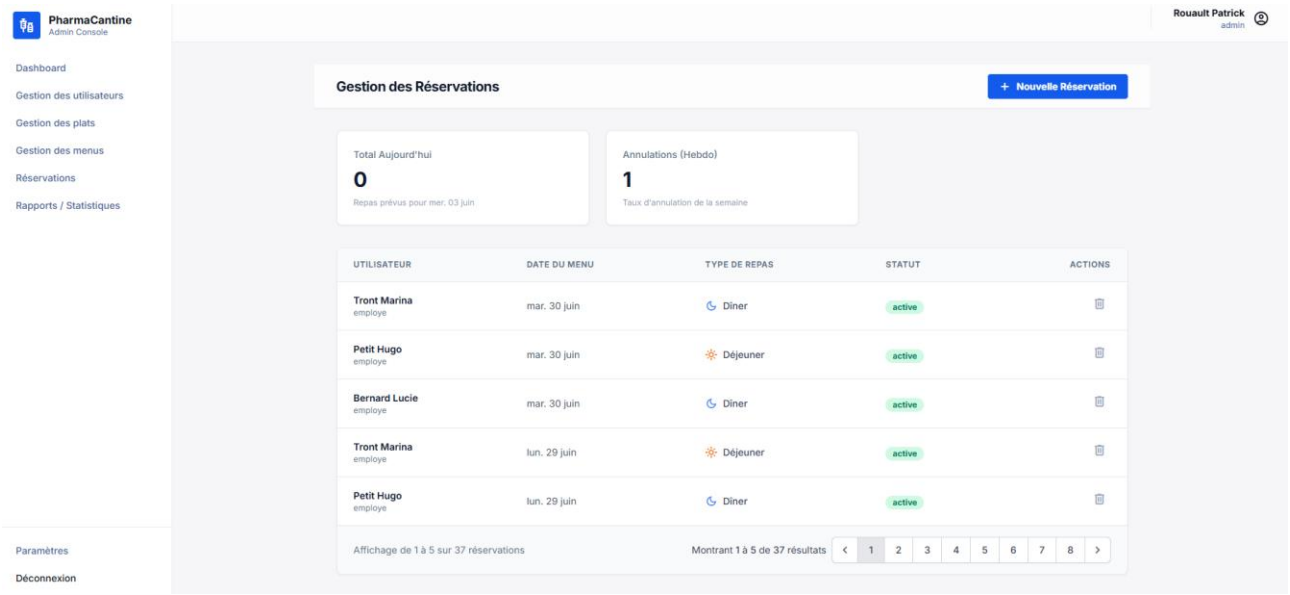
La liste des menus présente tous les menus avec leur date, leur statut (actif/annulé/archivé) et le nombre de plats par service. Des onglets de filtre permettent d'isoler les menus par statut. La création d'un menu implique de choisir une date future, un service (déjeuner ou dîner), et de composer le menu en sélectionnant au moins une entrée, un plat principal et un dessert parmi les plats disponibles, chacun avec une quantité prévue (1 à 500). La modification permet d'ajuster la composition et les quantités d'un service.





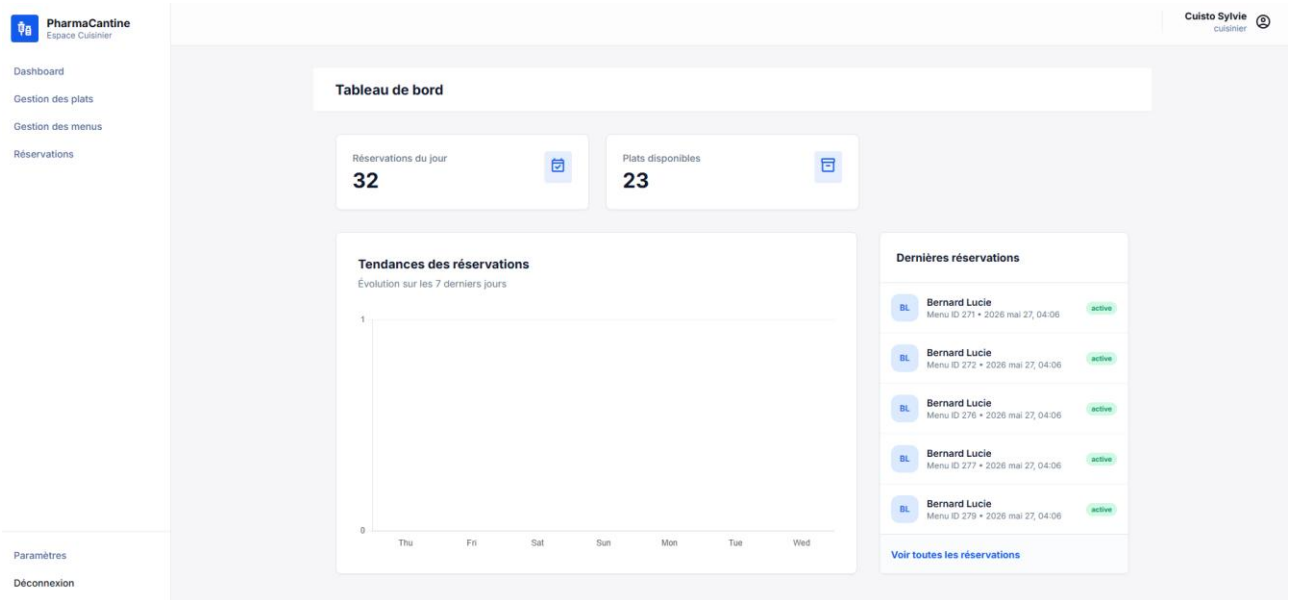
7.7 Gestion des réservations côté cantine (admin + cuisinier)

L'admin et le cuisinier accèdent à la liste complète de toutes les réservations, avec le nom de l'employé, le menu, le service et le statut. Ils peuvent créer une réservation pour le compte d'un employé spécifique en sélectionnant l'utilisateur, le menu et le service, avec vérification de la capacité (somme des qte_prevu) et des doublons. L'annulation d'une réservation est bloquée si la date est passée.



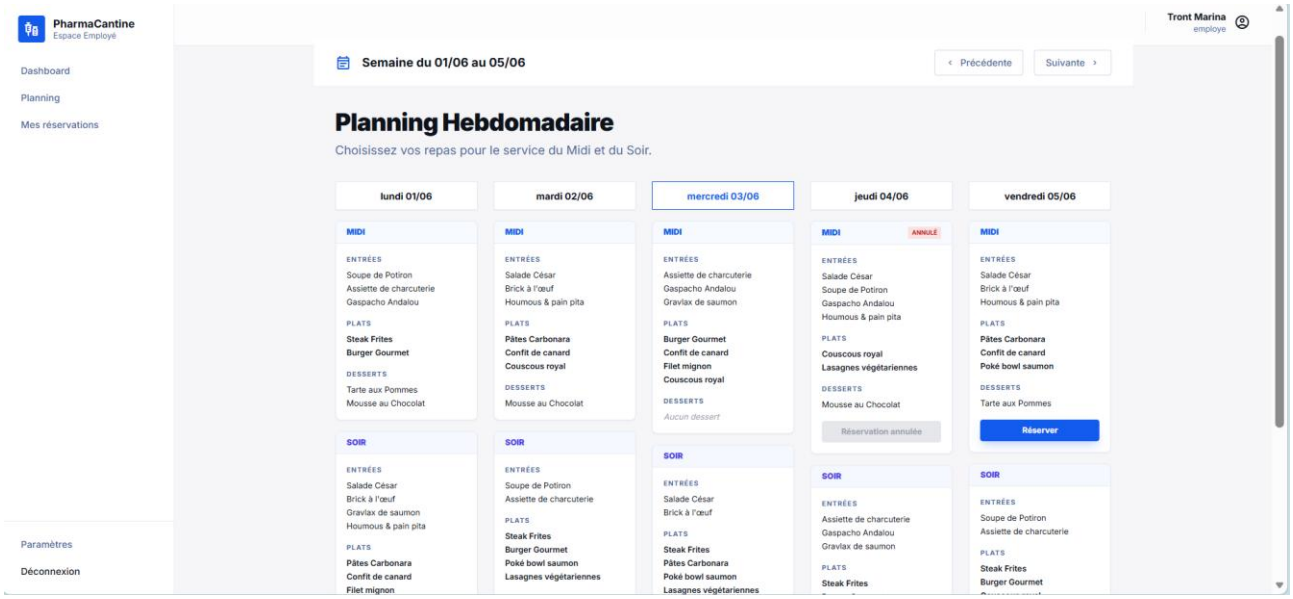
7.8 Dashboard cuisinier

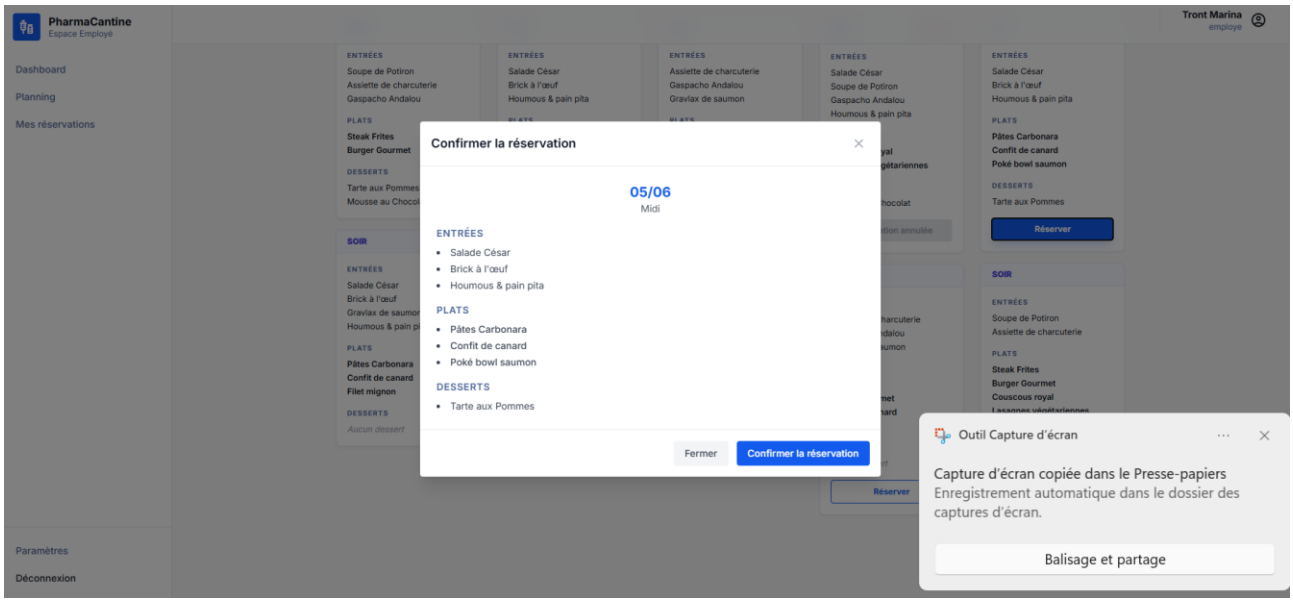
Le cuisinier dispose d'un dashboard dédié donnant accès rapide à la gestion des plats, des menus et des réservations. Il n'a pas accès à la gestion des utilisateurs ni aux statistiques globales de l'administrateur.

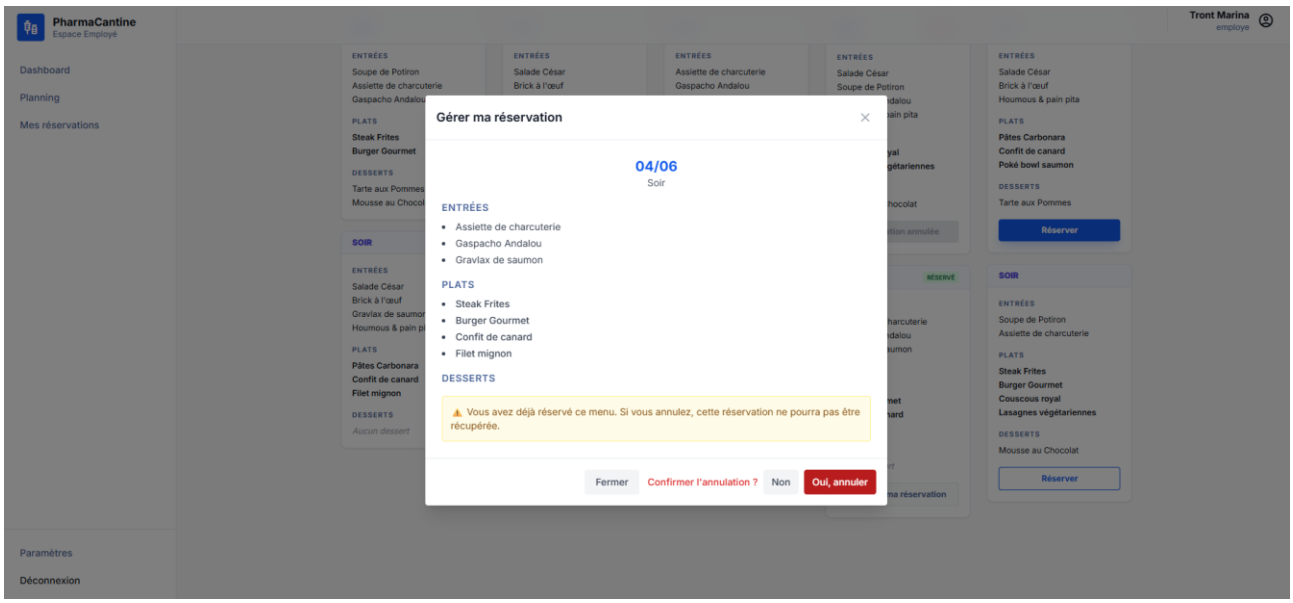


7.9 Planning hebdomadaire (employé)

L'employé accède au planning de la semaine courante (lundi au vendredi), navigable semaine par semaine via des paramètres de date en URL. Pour chaque jour, les menus du déjeuner et du dîner sont affichés avec leurs plats et leur disponibilité. Depuis cette vue, l'employé peut réserver ou annuler un créneau via une modale. Un indicateur visuel distingue les créneaux réservés, disponibles et complets.

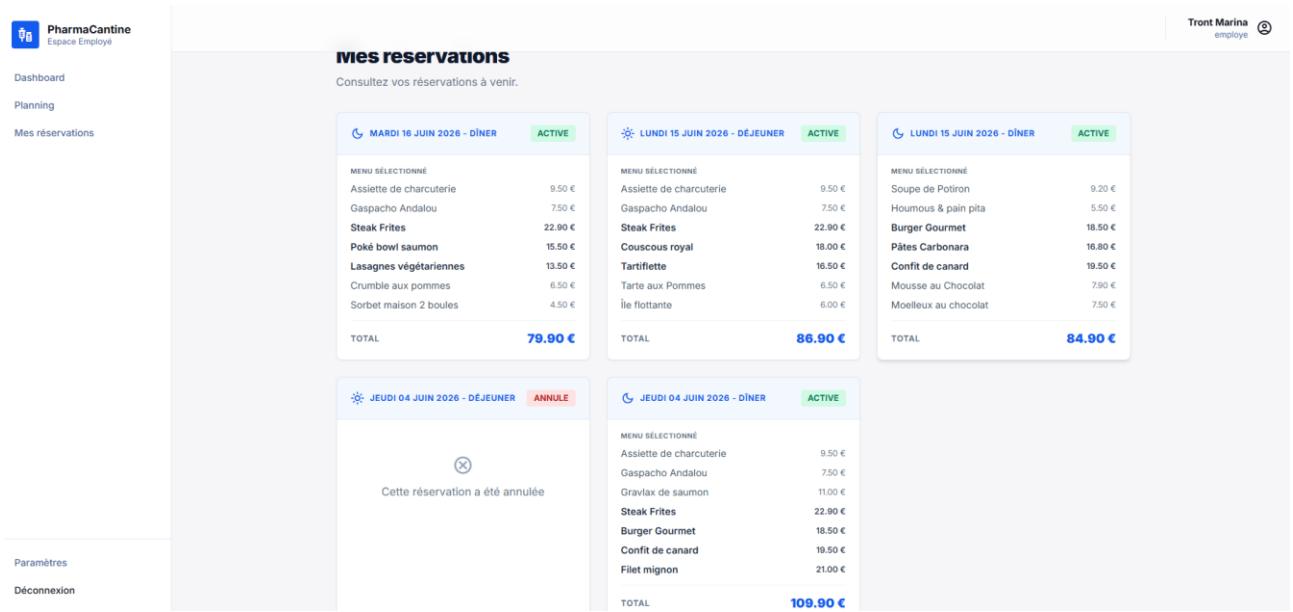






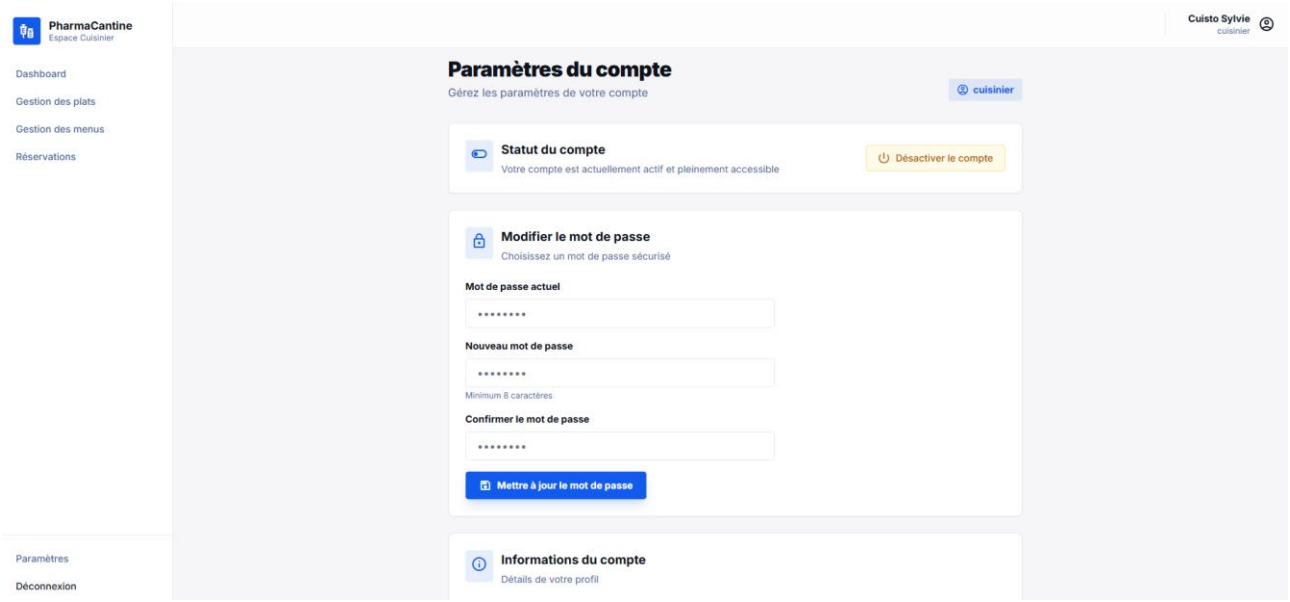
7.10 Mes réservations (employé)

L'employé consulte l'historique complet de ses réservations triées par date décroissante, avec le menu, le service, la date et le statut (active, annulée, terminée). Les prix sont la somme de tous les éléments du menu, le prix réel est affiché sur place selon ce que prend l'employé.



7.11 Paramètres du compte

Tous les rôles ont accès à une page de paramètres affichant leurs informations de profil. Le changement de mot de passe est sécurisé : l'ancien mot de passe est verifié avant la mise à jour, et la confirmation du nouveau mot de passe est obligatoire. Un utilisateur peut également désactiver son propre compte depuis cette page qui provoque une déconnexion immédiate.



8. Versioning et déploiement

Le projet a été versionné avec Git. Le dépôt est hébergé sur GitLab et accessible à l'adresse suivante :

https://gitlab.com/sco-chartreux/slam2-25-26/zhao-emma/pharmacantine

L'application est disponible a l'url : emmazh.fr

Les logins pour tester :

- admin : administrateur@pharmanutri.test, pharmaNutri2026 !
- employe : unemploye@pharmanutri.test, pharmaNutri2026!
- cuisinier : uncuisinier@pharmanutri.test, pharmaNutri2026!

Le dépôt est organisé selon la structure standard d'un projet Laravel. Le fichier docker-compose.yml à la racine permet de lancer l'environnement complet (application + base de données) avec la commande docker compose up -d.

L'application est déployée sur une VM Ubuntu hébergée sur l'infrastructure de l'établissement. Le backend est conteneurisé avec Docker Compose (services php, web et db). Le fichier .env contenant les credentials et la clé d'application APP_KEY n'est pas versionné dans le dépôt Git.

9. Conclusion

Ce projet m'a permis de développer une application web full stack complète en autonomie, de la conception de la base de données jusqu'au déploiement en environnement conteneurisé. J'ai approfondi mes compétences en architecture MVC avec Laravel 12, en gestion des autorisations par rôle via les Gates Laravel, et en composition d'interfaces serveur avec Blade et Tailwind CSS.

Les principales difficultés rencontrées ont été la gestion de la cohérence des statuts de menus et de réservations lors d'annulations en cascade et la configuration de l'environnement Docker pour le développement local sous WSL2.

Des axes d'amélioration sont identifiés pour une prochaine version : l'ajout de notifications par email lors de l'annulation d'un menu avec réservations actives, le développement d'une vue calendrier mensuelle pour les cuisiniers, et le renforcement de la sécurité avec un rate limiting sur le login et des headers HTTP de sécurité dans Nginx.