

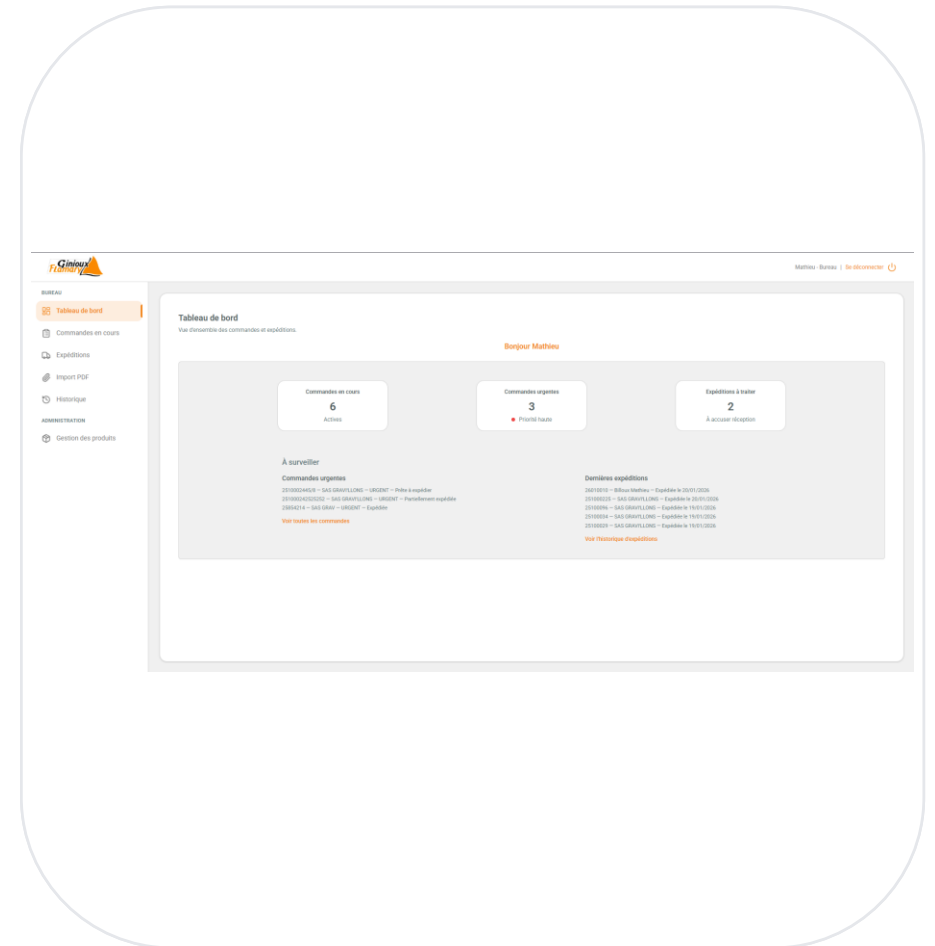
# Ginioux Flamary App

Application web interne — Bureau / Production / Admin

Démonstration projet | Auteur : Mathieu Billoux | 2026

GitHub : [github.com/Luxibol/ginioux\\_flamary\\_app](https://github.com/Luxibol/ginioux_flamary_app)

- Centraliser le suivi des commandes et leurs lignes produits
- Permettre à la Production de déclarer l'avancement (quantités prêtes)
- Suivre les expéditions (chargement, départ camion) + validation Bureau (ACK)
- Fournir un historique consultable
- Sécuriser les accès selon le rôle (ADMIN / BUREAU / PRODUCTION)



Exemple : tableau de bord Bureau

# Rôles & périmètre fonctionnel

## BUREAU

- Import PDF
- Suivi commandes (liste + détail)
- Validation expéditions (ACK)
- Historique

## PRODUCTION

- Commandes à produire
- Saisie quantités prêtes
- Expéditions à charger
- Départ camion

## ADMIN

- Pilotage (indicateurs)
- Catalogue produits
- Gestion utilisateurs

## Inclus

- Import commandes PDF
- Suivi Bureau
- Production
- Expéditions + ACK
- Historique

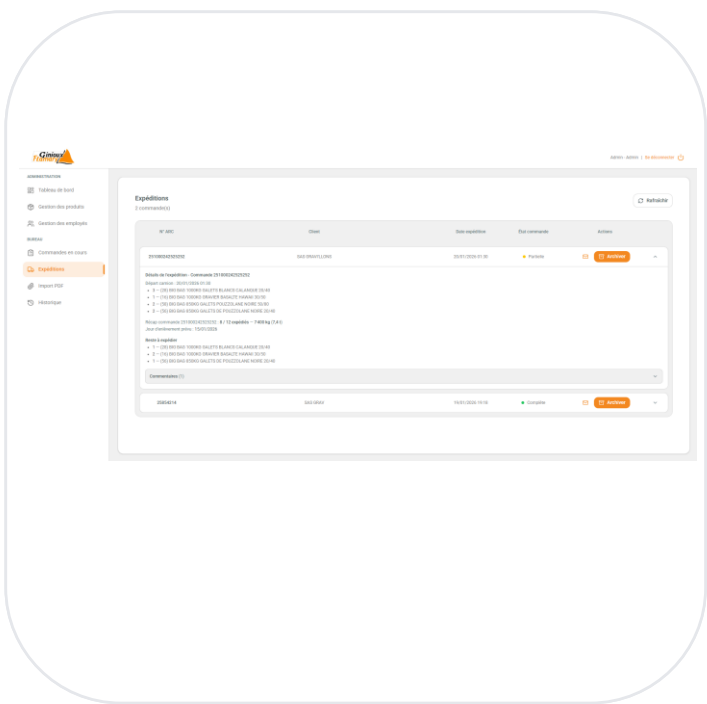
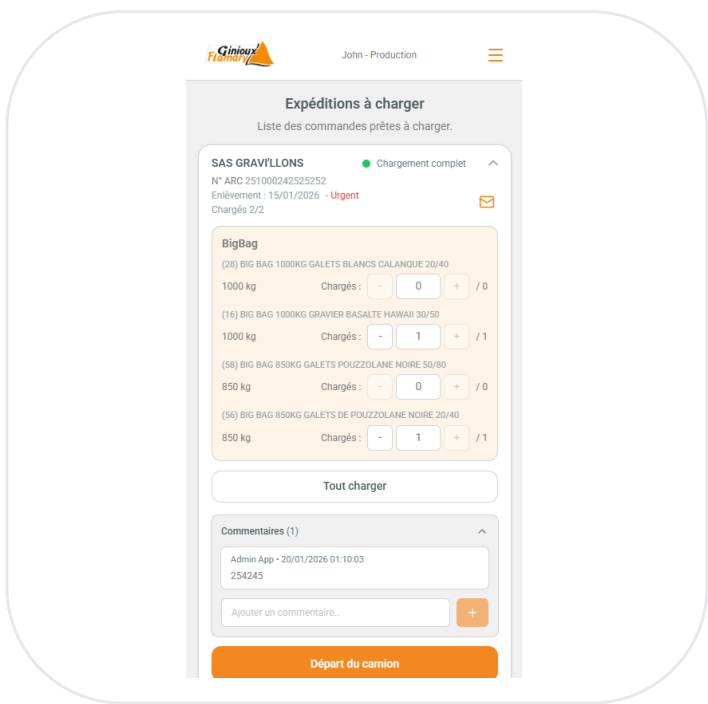
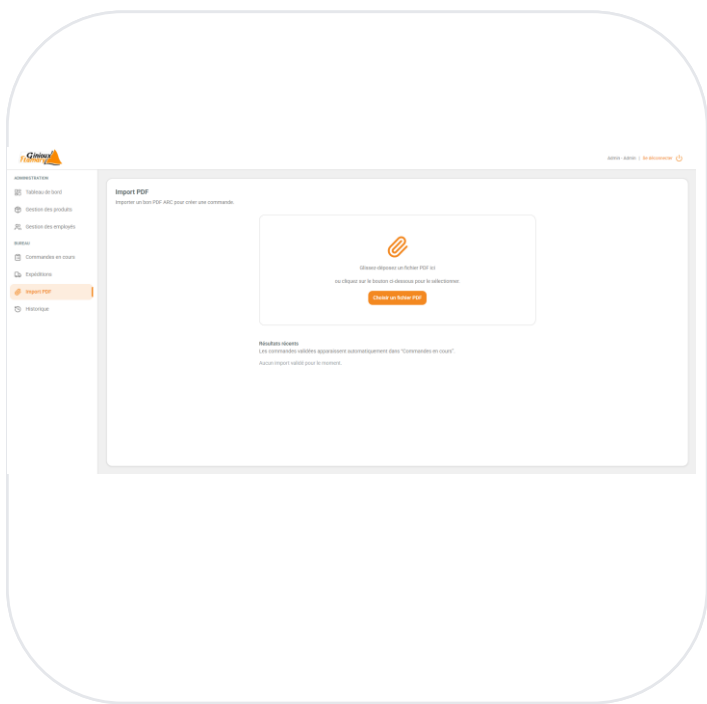
## Hors périmètre (à ce stade)

- Intégration directe au logiciel (API/exports) — l'import se fait via PDF/ARC
- Automatisations (notifications)
- Multi-sites avancé

# Parcours métier (end-to-end)

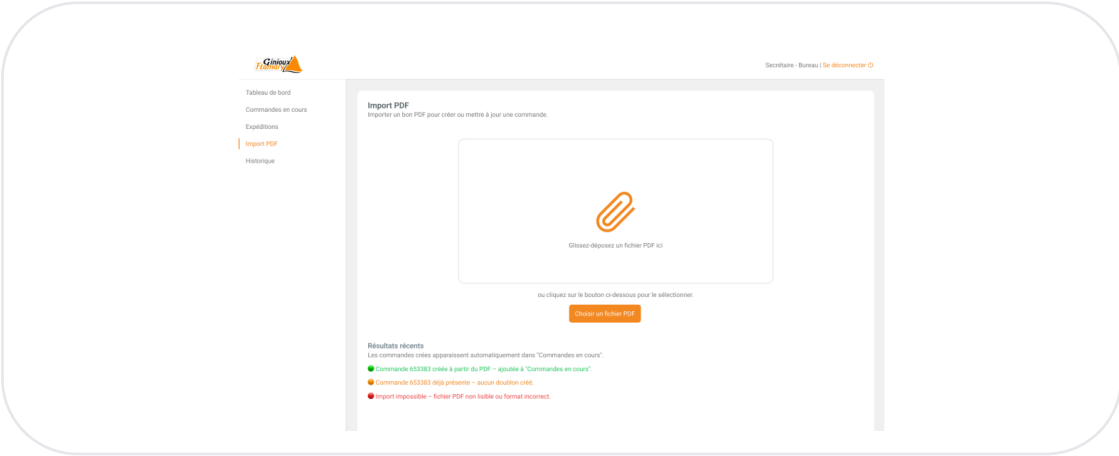


## Exemples d'écrans dans le flux

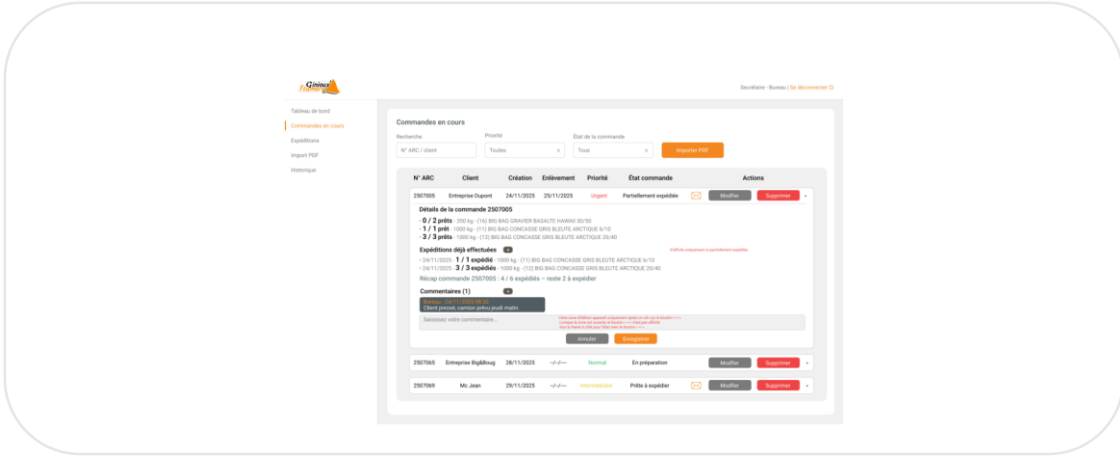


# Maquettes (Figma) — écrans clés

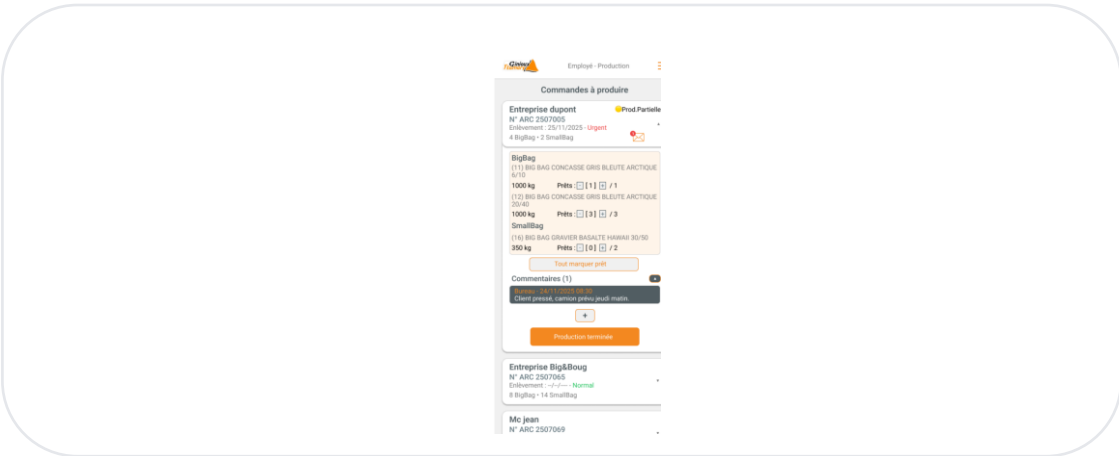
## Import PDF (Bureau)



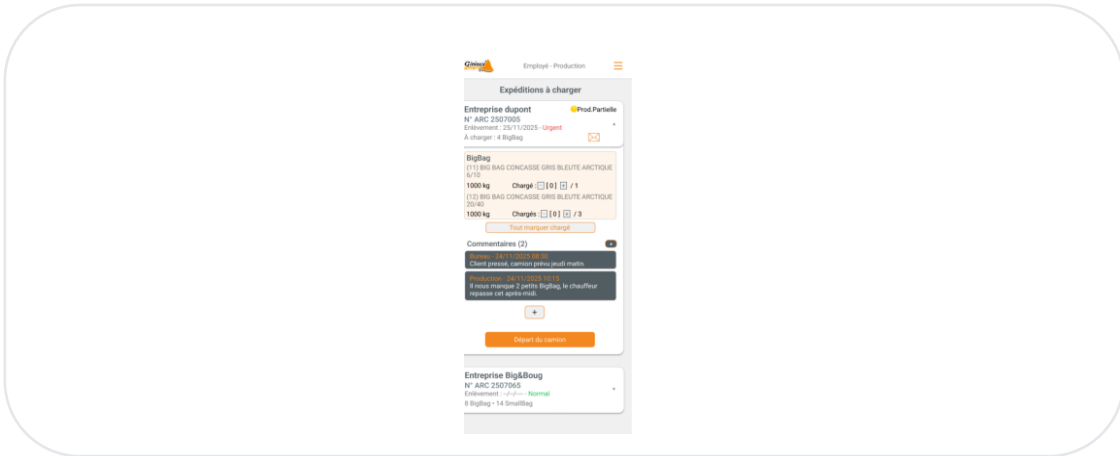
## Commandes en cours (Bureau)



## Commandes à produire (Production)

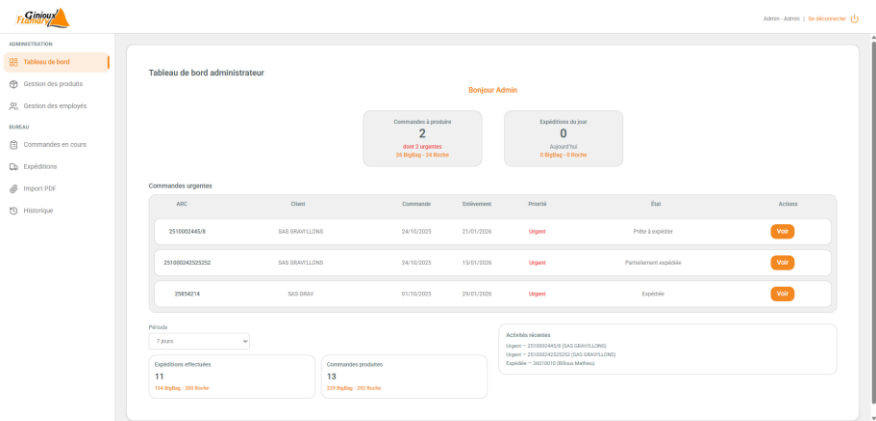


## Expéditions à charger (Production)



# Interfaces réalisées — aperçu

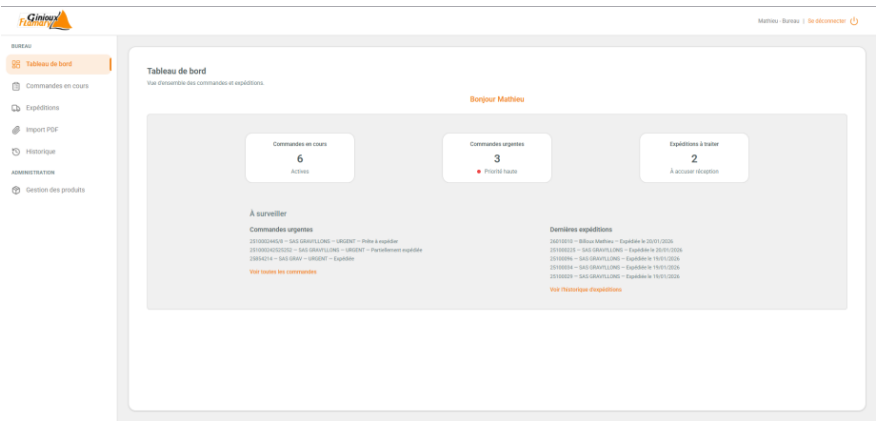
## Admin : pilotage



## Production : mobile

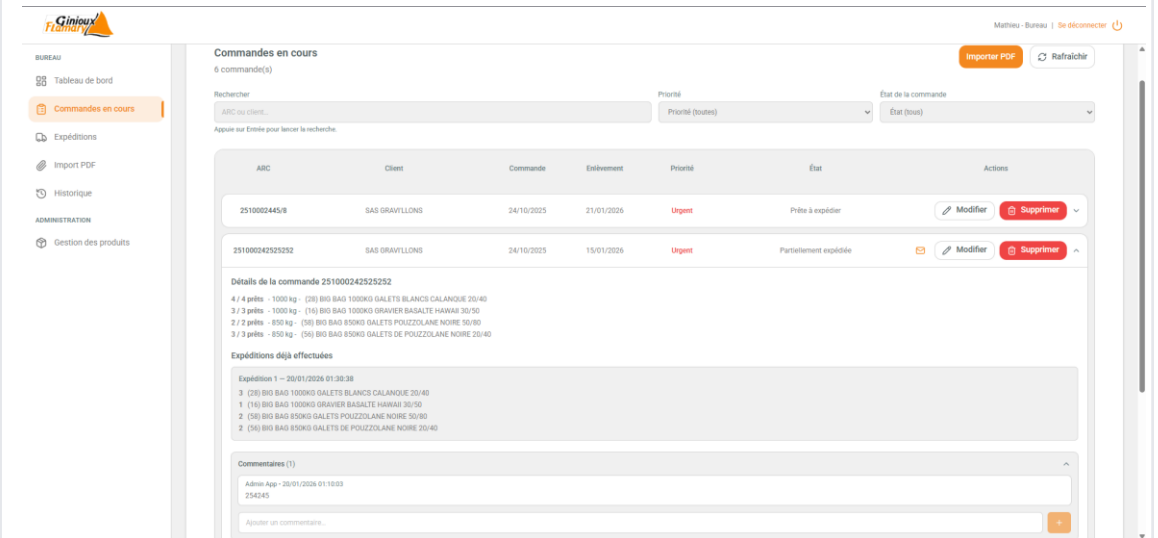


## Bureau : suivi & accès rapide



## Points clés

- Navigation par rôle (routes protégées)
- Listes + filtres (ARC, client, priorité, état)
- Détail commande : lignes, statuts, commentaires
- Actions métier : mise à jour des quantités & statuts via API
- Appels API centralisés + cookies envoyés (refresh token)



Exemple : Commande ouverte (lignes + expéditions + commentaires)

## API Express + règles métier

- Production : quantité prête (quantity\_ready) + recalcul statut
- Expéditions : quantité chargée (quantity\_loaded) + départ camion
- Bureau : accusé de réception (ACK) + historique
- Contrôles : bornage des quantités + erreurs HTTP cohérentes

```
Transaction + verrous FOR UPDATE  
Création du départ (shipments + shipment_lines)  
Maj quantités + recalcul statut + commit
```

# Base de données : modèle relationnel



## Tables principales

- orders (en-tête + statuts)
- order\_products (lignes + quantités)
- shipments / shipment\_lines
- users + refresh\_tokens
- order\_comments + reads

## Garanties

- Intégrité référentielle (clés étrangères)
- Cohérence des quantités (ready/loaded/shipped)
- Historique des expéditions par commande

## Authentication & session

- JWT (access token) pour les requêtes API
- Refresh token en cookie httpOnly
- Rotation + stockage en base (refresh\_tokens)
- Logout : révocation côté serveur

## Autorisation & API

- RBAC : ADMIN / BUREAU / PRODUCTION
- Routes protégées + contrôle des actions
- Validation des entrées (quantités, IDs)
- Erreurs HTTP cohérentes (400/401/403/422)

# Jeu d'essai — scénario de démonstration

Cycle complet : Import PDF → Bureau → Production → Expéditions → Historique

- 1

Import PDF (création commande)
- 2

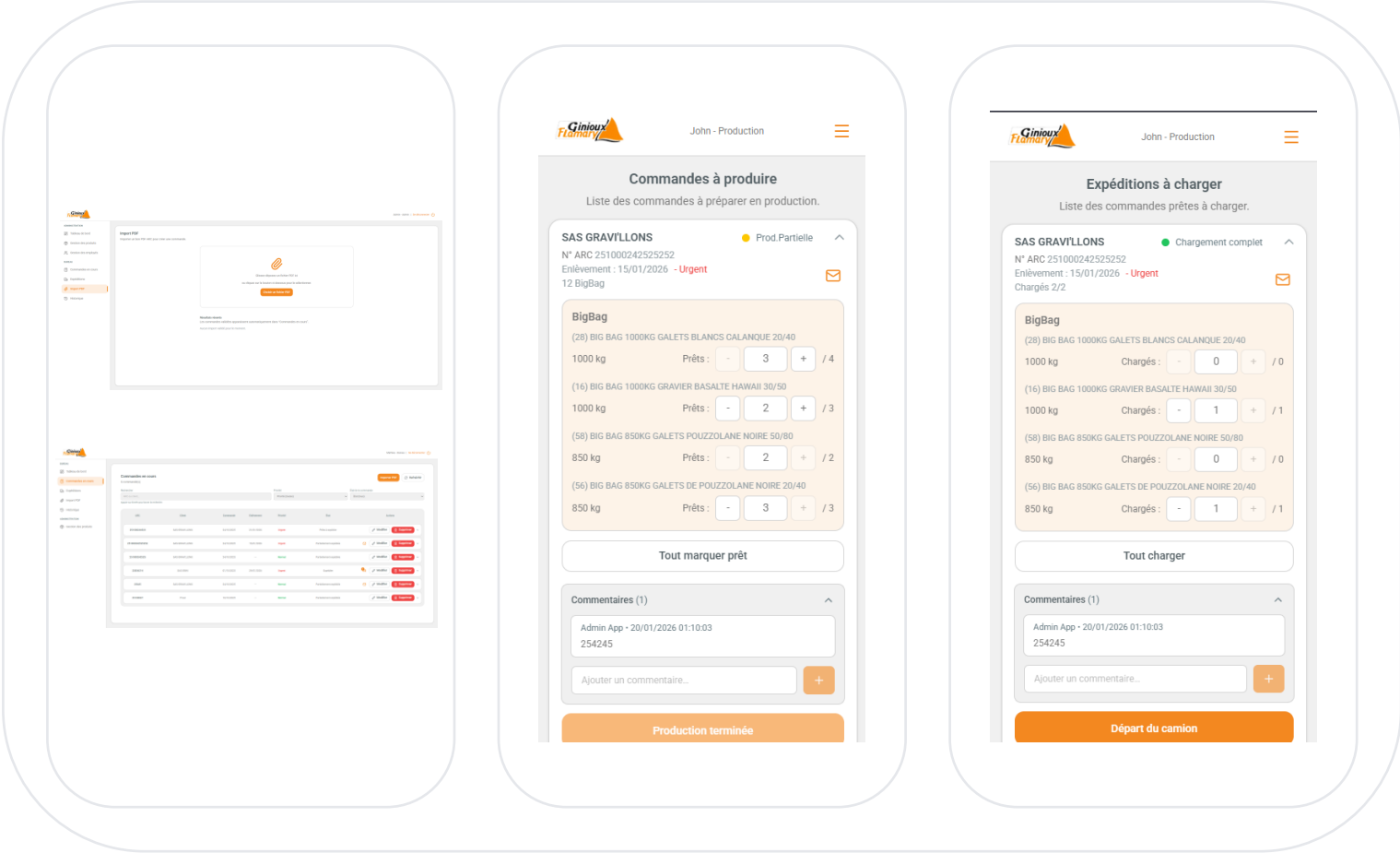
Bureau : vérification commande
- 3

Production : quantités prêtes
- 4

Production : chargement + départ
- 5

Bureau : ACK expédition
- 6

Bureau : Historique



Résultat : données cohérentes entre UI ↔ API ↔ BDD (quantités + statuts).

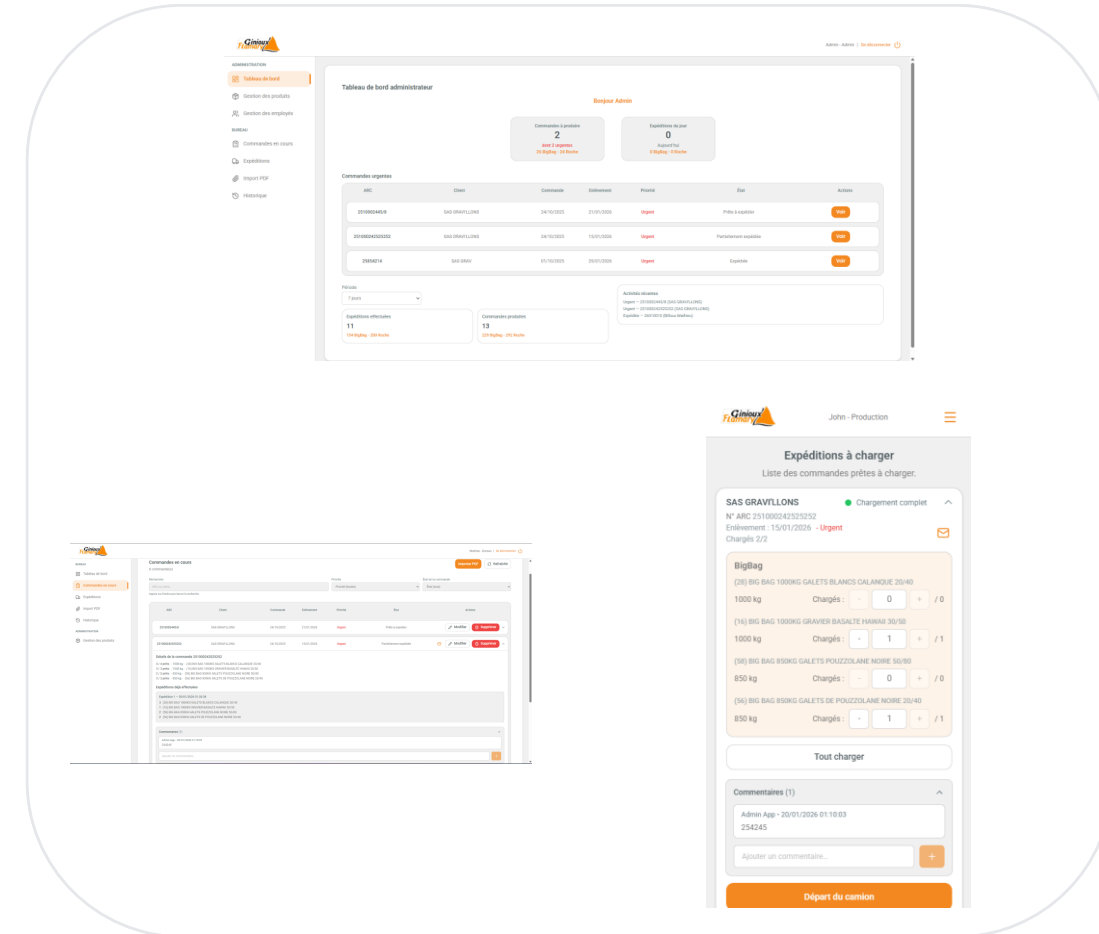
# Conclusion & améliorations

## Bilan

- Application fonctionnelle et sécurisée
- Flux complet : import → suivi → production → expéditions → historique
- Architecture séparée (Front React / API Express / BDD MySQL)

## Pistes d'amélioration

- Notifications (production terminée, expédition validée)
- Tableaux de bord avancés (stats, filtres temporels)
- Journal d'audit
- Tests automatisés (API + front)



## Dépôt

[github.com/Luxibol/ginioux\\_flamary\\_app](https://github.com/Luxibol/ginioux_flamary_app)

## Démarrage local (résumé)

- 1) Import SQL : backend/db/ginioux\_flamary.sql + donnees\_ginioux\_flamary.sql
- 2) backend/.env + frontend/.env
- 3) npm install + npm run dev (backend puis frontend)

## Captures

