

# **BillApp - Guide Complet (Base de Donnees a l'Interface)**

Document d'apprentissage personnel: architecture, logique metier, flux de facturation recurrente, Inertia/React TypeScript, theming Netflix Dark, mailing et bonnes pratiques de production.

Laravel 12

Inertia.js

React + TypeScript

Tailwind CSS

MySQL

## **1. Vision Produit**

**BillApp** est un SaaS de facturation recurrente inspire d'une ergonomie

streaming moderne.

•Inscription / connexion utilisateur.

•Souscription a un plan avec recurrence mensuelle ou annuelle.

•Generation immediate d'une premiere facture.

• Facturation automatique par commande planifiee (`app:run-billing`).

- Historique de factures visible depuis le dashboard, même sans abonnement

actif.

- Envoi email HTML Dark Mode de chaque facture.

## 2. Stack Technique et Responsabilités

Couche	Technologie	Role dans BillApp
Backen d	Laravel 12	Routes, validation, transactions, Eloquent, commandes Artisan, mails.
Fronten d	Inertia + React + TypeScript	Pages SPA server-driven, formulaires, rendu des états d'abonnement/factures.
UI	Tailwind CSS	Theme Netflix Dark: fond #050505, cartes #141414, accent #E50914.
DB	MySQL/SQLite	Persistance plans, subscriptions, invoices, users.
Email	Laravel Mail + Symfony Mailer	Envoi des factures HTML avec référence et montant FCFA.

## 3. Architecture Base de Données

### 3.1 Tables métier

Table	Colonnes principales	Notes métier
plans	<code>id, name, price_xof, frequency</code>	Catalogue des offres. Seed initial:

Table	Colonnes principales	Notes metier
		Basique mensuel, Premium annuel.
subscription s	user_id, plan_id, billing_period, status, next_billing_at	Suivi d'abonnement actif/annule + prochaine échéance.
invoices	user_id, amount, reference, billed_at	Trace de facturation immutable côte historique.

### 3.2 Choix importants de schema

- `price_xof` en entier non signé: évite les erreurs de flottants sur la monnaie.
- `reference` unique: garantit l'unicité de facture.
- Index `subscriptions(status, next_billing_at)`: critique pour le scan des échéances.
- Index `invoices(billed_at)`: accélère le tri historique récent.

### 3.3 Relations Eloquent

```

Plan hasMany Subscription
User hasOne activeSubscription(where status=active)
User hasMany subscriptions
User hasMany invoices
Subscription belongsTo User

```

Subscription belongsTo Plan

Invoice belongsTo User

## 4. Logique de Souscription (SubscriptionController)

### 4.1 Validation d'entree

plan\_id: required|integer|exists:plans,id

billing\_period: required|in:month,year

### 4.2 Regles metier appliquees

1. Interdire une nouvelle souscription si un abonnement actif existe deja.

2. Vérifier la cohérence entre plan.frequency et billing\_period.

3. Calculer next\_billing\_at avec overflow-safe: addMonthNoOverflow() ou

addYearNoOverflow().

4. Dans une transaction DB: créer subscription + première invoice.

5. Envoyer l'email de facture après transaction (try/catch + log en cas

d'échec).

## 4.3 Pourquoi la date de prochaine facturation était mauvaise

Le bug initial venait d'un calcul ou d'un type date incohérent. La version stabilisée

renvoie CarbonInterface et calcule une date future avec addMonthNoOverflow() ou

addYearNoOverflow(), puis stocke explicitement next\_billing\_at.

## 5. Automatisation de Facturation (Commande Artisan)

### 5.1 Signature

```
php artisan app:run-billing
```

### 5.2 Algorithme de run

1. Charger les abonnements actifs arrivés à échéance: status=active et

```
next_billing_at <= now().
```

2. Pour chaque abonnement du:

3. Créer facture (amount = plan.price\_xof, nouvelle référence INV-XXXX).

- 
4. Avancer `next_billing_at` a partir de l'ancienne échéance (pas à partir de now).
- 

5. Tenter l'envoi email de facture (échec journalisé, sans rollback métier).

---

### 5.3 Planification recommandée

```
// routes/console.php ou bootstrap scheduler  
Schedule::command('app:run-billing')->everyMinute();
```

En production, lancer le scheduler système (cron) pour déclencher Laravel Scheduler.

## 6. Emails de Facture (InvoiceMail)

### Contenu

- Objet: Votre facture INV-XXXX - BillApp

- Vue HTML: resources/views/emails/invoice.blade.php

• Montant formate en FCFA

- Reference facture

- Date locale de facturation

## Robustesse

- Try/catch autour de `Mail::send.`

- Log d'erreur détaillé (`user_id, reference, message`).

- Le métier ne casse pas si SMTP tombe.

### 6.1 En local vs production

En local, les utilisateurs ne reçoivent pas de vrais mails sans SMTP public. Utiliser

Mailpit/Mailhog pour visualiser localement. En production: Brevo, Mailgun,

Postmark, SES, etc.

## 7. Dashboard Inertia/React (Logique UI)

### 7.1 Donnees envoyees par DashboardController

```
{  
  plans: BillingPlan[],  
  activeSubscription: BillingSubscription | null,  
  invoices: BillingInvoice[] // 20 dernieres factures  
}
```

### 7.2 Comportements interface

- Header: retour accueil, logo unifie, bouton profil a droite, bouton

deconnexion.

- Si non abonne: formulaire de souscription (plan + periode + recap

dynamique).

- Si abonne: badge Actif, plan, prix, prochaine facturation, bouton annulation.

- Historique factures visible meme sans abonnement actif.

- Boutons de navigation retour (dont retour accueil depuis dashboard).

## 7.3 Typage TypeScript

```
// resources/js/types/billing.ts

export type PlanFrequency = 'month' | 'year';
export type SubscriptionStatus = 'active' | 'cancelled';

export interface BillingPlan {
  id: number;
  name: string;
  price_xof: number;
  frequency: PlanFrequency;
}

export interface BillingSubscription {
  id: number;
  status: SubscriptionStatus;
  billing_period: PlanFrequency;
  next_billing_at: string;
  plan: BillingPlan;
}

export interface BillingInvoice {
  id: number;
  amount: number;
  reference: string;
  billed_at: string;
}
```

## 7.4 Format monetaire FCFA

```
const formatMoney = (value: number) =>
` ${new Intl.NumberFormat('fr-FR').format(value)} FCFA`;
```

## 8. Design System "Netflix Dark" + Animations

- Fond global: #050505.

- Cartes: #141414.

- Accent principal: #E50914.

- Transitions modernes: hover elevation, scale, glow, marquee, sheen,

fadeUp.

- Navbar accueil: fixed + flottante au scroll + largeur complete.

## 8.1 Uniformisation logo

### Contexte

### Variant logo

Pages auth (login/register)

Logo blanc (`variant="white"`)

Autres pages

Logo degrade rouge

(welcome/dashboard)

(`variant="gradient"`)

## 9. Profil Utilisateur

- Edition du nom et email via page settings profile.

- Accès depuis bouton `MON PROFIL` dans le dashboard.

- Bouton deconnexion conserve dans le header (sans nom utilisateur affiche à coté).

## | 10. Routes métier principales

```
GET /           -> welcome
GET /dashboard -> DashboardController (auth+verified)
POST /subscriptions -> SubscriptionController@store
PATCH /subscriptions/cancel -> SubscriptionController@cancel
POST /logout      -> deconnexion
GET /settings/profile -> edition profil
```

## | 11. Workflow de bout en bout

1.L'utilisateur se connecte et ouvre dashboard.

2.Choisit plan + période, puis confirme abonnement.

3.Backend crée abonnement actif + facture initiale (transaction).

4.Email facture envoyé (ou log erreur sans casser le flux utilisateur).

5.Scheduler/commande facture automatiquement aux échéances suivantes.

6.Dashboard affiche et met a jour l'historique des factures.

## 12. Commandes utiles de dev

```
php artisan migrate  
php artisan serve  
npm run dev  
php artisan app:run-billing  
php artisan queue:work // si emails en file d'attente  
php artisan schedule:work // test local du scheduler
```

## 13. Tests recommandes (important)

### Cas prioritaires

•Souscription refusee si abonnement actif deja present.

•Souscription refusee si periode != frequence du plan.

•Creation transactionnelle: subscription + invoice atomiques.

• `next_billing_at` est bien dans le futur (mois/annee).

•Commande billing cree facture et avance correctement la recurrence.

- Historique visible même quand `activeSubscription = null`.

## Extrait de stratégie

- Tests Feature HTTP pour SubscriptionController
- Tests Unit pour generateReference et calcul recurrence
- Tests Console pour app:run-billing
- Mail::fake() pour vérifier émission des emails

## 14. Recommandations de production

- Utiliser un provider SMTP fiable + domaine vérifié (SPF, DKIM, DMARC).
- Passer l'envoi email en queue async (`ShouldQueue`) pour performance.
- Ajouter idempotence et verrouillage pour éviter double facturation en

concurrence.

- Ajouter webhooks paiement si integration future Stripe/Paystack/CinetPay.

- Mettre monitoring: logs, alertes erreurs mail, ratio factures creees/envoyees.

## | 15. Resume Pedagogique

Tu as construit une base SaaS solide: schema propre, logique d'abonnement

coherente, facturation recurrente automatisee, experience UI premium, typage

TypeScript clair, et mecanisme email resilient. Le socle est pret pour des

evolutions paiement reel, analytics et multi-plans plus avances.