

# 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	Rôle dans BillApp
Backend	Laravel 12	Routes, validation, transactions, Eloquent, commandes Artisan, mails.
Frontend	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	id, name, price_xof, frequency	Catalogue des offres. Seed initial:

Table	Colonnes principales	Notes metier
		Basique mensuel, Premium annuel.
<code>subscriptions</code>	<code>user_id</code> , <code>plan_id</code> , <code>billing_period</code> , <code>status</code> , <code>next_billing_at</code>	Suivi d'abonnement actif/annule + prochaine echeance.
<code>invoices</code>	<code>user_id</code> , <code>amount</code> , <code>reference</code> , <code>billed_at</code>	Trace de facturation immutable cote historique.

### 3.2 Choix importants de schema

- `price_xof` en entier non signe: evite les erreurs de flottants sur la monnaie.

- `reference` unique: garantit l'unicite de facture.

- Index `subscriptions(status, next_billing_at)`: critique pour le scan des

echeances.

- Index `invoices(billed_at)`: accelere le tri historique recent.

### 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. Verifier la coherence entre `plan.frequency` et `billing_period`.

3. Calculer `next_billing_at` avec overflow-safe: `addMonthNoOverflow()` ou

```
addYearNoOverflow().
```

4. Dans une transaction DB: creer subscription + premiere invoice.

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

d'echec).

### 4.3 Pourquoi la date de prochaine facturation etait mauvaise

Le bug initial venait d'un calcul ou d'un type date incoherent. La version stabilisee

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 arrives a echeance: `status=active` et

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

2. Pour chaque abonnement du:

3. Creer facture (`amount = plan.price_xof`, nouvelle reference `INV-XXXX`).

4. Avancer `next_billing_at` a partir de l'ancienne echeance (pas a partir de `now`).

5.Tenter l'envoi email de facture (echec journalise, sans rollback metier).

### 5.3 Planification recommandee

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

En production, lancer le scheduler systeme (cron) pour declencher 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 detaille (user\_id, reference, message).

- Le metier ne casse pas si SMTP tombe.

### 6.1 En local vs production

En local, les utilisateurs ne recoivent 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 monétaire 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

Pages auth (login/register)

Autres pages

(welcome/dashboard)

### Variant logo

Logo blanc (variant="white")

Logo degrade rouge

(variant="gradient")

## 9. Profil Utilisateur

- Edition du nom et email via page settings profile.

- Acces depuis bouton MON PROFIL dans le dashboard.

- Bouton deconnexion conserve dans le header (sans nom utilisateur affiche a

cote).

## | 10. Routes metier 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 + periode, puis confirme abonnement.

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

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

5.Scheduler/commande facture automatiquement aux echeances 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 meme quand `activeSubscription = null`.

## Extrait de strategie

- Tests Feature HTTP pour SubscriptionController

- Tests Unit pour generateReference et calcul recurrence

- Tests Console pour app:run-billing

- Mail::fake() pour verifier emission des emails

## 14. Recommandations de production

- Utiliser un provider SMTP fiable + domaine verifie (SPF, DKIM, DMARC).

- Passer l'envoi email en queue async (`ShouldQueue`) pour performance.

- Ajouter idempotence et verrouillage pour eviter double facturation en

concurrency.

- 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.