Document Complet - Solution Gestion Budgétaire et Workflow d'Achat

**Document de Référence pour Agents IA - Version Spécialisée Gestion Budget & Achats**

1. CONTEXTE ET EXPRESSION DE BESOIN MÉTIER

**Objectif Stratégique**

Digitaliser intégralement le processus de gestion budgétaire et workflow d'achat organisationnel avec une plateforme web responsive. Centraliser la gestion des budgets par service, les demandes de devis, les validations multi-niveaux, le suivi des commandes et livraisons avec traçabilité complète conforme aux exigences de contrôle financier.

**Processus Métier Détaillé**

**Étape 1 - Préparation Budget Annuel :**

* Services proposent leurs budgets avec 12 colonnes détaillées
* Responsable budget valide les enveloppes par service
* Cloisonnement strict : chaque service voit uniquement son budget
* Responsable budget a vision globale tous services

**Étape 2 - Demande de Devis :**

* Service X saisit : dénomination, référence, prix, quantité, justification
* Sélection ligne budgétaire d'imputation
* Vérification automatique budget disponible
* Soumission à responsable budget

**Étape 3 - Validation Budgétaire :**

* Responsable budget vérifie cohérence avec planification
* Actions : Approuver/Rejeter/Demander infos/Réaffecter ligne
* Validation conditionnelle selon consommation globale

**Étape 4 - Validation Achat :**

* Service achat analyse fournisseur et conditions commerciales
* Actions : Valider + commander/Valider + déléguer commande/Rejeter
* Optimisation : regroupement commandes, négociation

**Étape 5 - Commande et Suivi :**

* Commande passée par service achat ou service demandeur
* Suivi livraison avec planning prévisionnel
* Relances automatiques fournisseurs

**Étape 6 - Réception et Contrôle :**

* Upload bon de livraison (PDF/image)
* Vérification complétude/état par service demandeur
* Gestion litiges : relances jusqu'à livraison conforme
* Mise à jour budget avec dépense réelle

**Structure Budget par Service**

Chaque ligne budget contient :

* Date prévisionnelle d'achat
* Intitulé de la dépense
* Nature (abonnement, matériel, infrastructure...)
* Fournisseur prévu
* Base de calcul (estimation/prix ferme)
* Quantité
* Montant HT prévu
* Montant TTC prévu
* Suivi dépassement (prévu - dépensé)
* Catégorie (software/hardware/mobilier...)
* Type dépense (CAPEX/OPEX)
* Commentaire service
* Commentaire suivi budget
* Statut validation (oui/non/potentiellement)

**Contraintes Techniques Critiques**

* Hébergement local obligatoire (pas de cloud)
* Application web responsive tous navigateurs
* Cloisonnement strict données par service
* Temps réponse < 2 secondes
* Conformité RGPD et sécurité renforcée
* Interface intuitive (max 3 clics par action)
* Traçabilité complète toutes opérations

2. ARCHITECTURE UI/UX COMPLÈTE

**Arborescence Générale par Profil**

text

📱 Application Gestion Budget & Achats

├── 🏠 Dashboard (Personnalisé par profil)

├── 💰 Gestion Budgétaire

│ ├── Mon Budget (Service Demandeur)

│ ├── Tous les Budgets (Responsable Budget)

│ ├── Préparation Budget Annuel

│ ├── Suivi Consommation Temps Réel

│ ├── Analyses & Projections

│ └── Réallocations Inter-Services

├── 🛒 Demandes & Achats

│ ├── Nouvelle Demande de Devis

│ ├── Mes Demandes (Service Demandeur)

│ ├── Demandes à Valider Budget

│ ├── Demandes à Valider Achat

│ ├── Commandes en Cours

│ ├── Planning Livraisons

│ └── Historique Complet

├── 📦 Livraisons & Réception

│ ├── Livraisons Attendues

│ ├── Vérifications Pendantes

│ ├── Relances Fournisseurs

│ ├── Gestion Litiges

│ └── Bons de Livraison

├── 📊 Reporting & Analytics

│ ├── Tableaux de Bord Exécutifs

│ ├── Analyses Budgétaires

│ ├── Performance Fournisseurs

│ ├── KPIs Achats

│ └── Exports & Rapports

├── 👤 Profil & Notifications

│ ├── Mon Profil

│ ├── Délégations

│ ├── Paramètres Notifications

│ └── Historique Activité

└── ⚙️ Administration

├── Configuration Services

├── Gestion Utilisateurs & Rôles

├── Paramètres Workflow

├── Configuration Budget

└── Maintenance Système

**Dashboards Spécialisés par Rôle**

**Dashboard Service Demandeur :**

* Widgets : Budget restant (jauge visuelle), Demandes en cours, Livraisons attendues
* Actions rapides : Nouvelle demande, Consulter budget, Vérifier livraison
* Alertes : Dépassements potentiels, Validations requises
* Graphiques : Consommation mensuelle vs budget, Répartition par catégorie

**Dashboard Responsable Budget :**

* Vue consolidée tous services avec codes couleur urgence
* Demandes à valider priorisées par montant et délai
* Projections financières et alertes dépassement
* Outils réallocation inter-services
* Statistiques : taux validation, délais moyens

**Dashboard Service Achat :**

* Queue demandes à traiter avec analyse fournisseur
* Commandes en cours et suivi livraisons
* Relances requises avec escalade automatique
* Performance fournisseurs (délais, qualité, prix)
* Planning charge de travail

3. SPÉCIFICATIONS TECHNIQUES DÉTAILLÉES

**Stack Technique Recommandée**

**Backend :**

text

PHP 8.1+

Laravel 10+ avec packages :

- ringlesoft/laravel-process-approval (workflow)

- filament/filament (interface admin)

- spatie/laravel-medialibrary (fichiers)

- maatwebsite/excel (exports)

- spatie/laravel-permission (rôles)

- livewire/livewire (composants réactifs)

**Base de Données :**

text

PostgreSQL 15+ (performance, JSON support)

Redis (cache, queues, sessions)

**Infrastructure :**

text

Docker + docker-compose

Nginx (reverse proxy)

SSL/TLS obligatoire

Backup automatique quotidien

**Modèles de Données Spécialisés**

**Modèle Budget Principal :**

php

**class** BudgetLigne **extends** Model

{

**protected** $fillable = [

'service\_id', 'date\_prevue', 'intitule', 'nature',

'fournisseur\_prevu', 'base\_calcul', 'quantite',

'montant\_ht\_prevu', 'montant\_ttc\_prevu', 'depassement',

'categorie', 'type\_depense', 'commentaire\_service',

'commentaire\_budget', 'valide\_budget'

];

**protected** $casts = [

'date\_prevue' => 'date',

'montant\_ht\_prevu' => 'decimal:2',

'montant\_ttc\_prevu' => 'decimal:2',

'valide\_budget' => 'string' *// oui/non/potentiellement*

];

}

**Modèle Demande de Devis :**

php

**class** DemandeDevis **extends** Model **implements** Approvable

{

**use** HasApprovals, HasMediaConversions;

**protected** $fillable = [

'service\_demandeur\_id', 'budget\_ligne\_id',

'denomination', 'reference\_produit', 'description',

'quantite', 'prix\_unitaire\_ht', 'prix\_total\_ttc',

'fournisseur\_propose', 'justification\_besoin',

'urgence', 'date\_besoin', 'statut'

];

*// Configuration workflow 3 niveaux*

**public** **function** getApprovalFlowAttribute()

{

**return** 'demande-devis-workflow';

}

}

**Modèle Commande :**

php

**class** Commande **extends** Model

{

**protected** $fillable = [

'demande\_devis\_id', 'numero\_commande',

'date\_commande', 'commanditaire', 'statut',

'date\_livraison\_prevue', 'montant\_reel'

];

**protected** $casts = [

'date\_commande' => 'date',

'date\_livraison\_prevue' => 'date'

];

}

**Modèle Livraison :**

php

**class** Livraison **extends** Model

{

**use** HasMediaConversions; *// Pour bons de livraison*

**protected** $fillable = [

'commande\_id', 'date\_livraison', 'statut\_reception',

'commentaire\_reception', 'verifie\_par',

'conforme', 'actions\_requises'

];

**protected** $casts = [

'date\_livraison' => 'date',

'conforme' => 'boolean'

];

}

**Configuration Workflow Spécialisé**

**Workflow Demande de Devis (3 niveaux) :**

php

ApprovalFlow::create([

'name' => 'demande-devis-workflow',

'steps' => [

[

'role' => 'responsable-budget',

'action' => 'approve',

'conditions' => ['budget\_disponible' => true]

],

[

'role' => 'service-achat',

'action' => 'approve',

'conditions' => ['fournisseur\_valide' => true]

],

[

'role' => 'service-demandeur',

'action' => 'verify\_delivery',

'auto\_trigger' => 'on\_delivery'

]

]

]);

4. INTERFACES FILAMENT SPÉCIALISÉES

**Resource Budget Management :**

php

**class** BudgetLigneResource **extends** Resource

{

**protected** **static** ?**string** $model = BudgetLigne::**class**;

**public** **static** **function** form(Form $form): Form

{

**return** $form->schema([

DatePicker::make('date\_prevue')->required(),

TextInput::make('intitule')->required(),

Select::make('nature')->options([

'abonnement' => 'Abonnement annuel',

'materiel' => 'Matériel',

'infrastructure' => 'Infrastructure'

]),

TextInput::make('montant\_ht\_prevu')->numeric()->required(),

Select::make('type\_depense')->options([

'CAPEX' => 'CAPEX',

'OPEX' => 'OPEX'

]),

Select::make('valide\_budget')->options([

'oui' => 'Validé',

'non' => 'Non validé',

'potentiellement' => 'Potentiellement'

])

]);

}

**public** **static** **function** table(Table $table): Table

{

**return** $table->columns([

TextColumn::make('intitule')->searchable(),

TextColumn::make('montant\_ht\_prevu')->money('EUR'),

BadgeColumn::make('valide\_budget')->colors([

'success' => 'oui',

'danger' => 'non',

'warning' => 'potentiellement'

]),

TextColumn::make('depassement')->getStateUsing(

**fn** ($record) => $record->montant\_ht\_prevu - $record->montant\_depense

)

])->filters([

SelectFilter::make('categorie'),

SelectFilter::make('type\_depense'),

SelectFilter::make('valide\_budget')

]);

}

}

**Resource Demande de Devis :**

php

**class** DemandeDevisResource **extends** Resource

{

**protected** **static** ?**string** $model = DemandeDevis::**class**;

**public** **static** **function** form(Form $form): Form

{

**return** $form->schema([

Section::make('Identification Produit')->schema([

TextInput::make('denomination')->required(),

TextInput::make('reference\_produit'),

Textarea::make('description')->rows(3)

]),

Section::make('Aspects Financiers')->schema([

Select::make('budget\_ligne\_id')

->relationship('budgetLigne', 'intitule')

->searchable()

->required(),

TextInput::make('quantite')->numeric()->required(),

TextInput::make('prix\_unitaire\_ht')->numeric()->required(),

TextInput::make('prix\_total\_ttc')->numeric()->required()

]),

Section::make('Justification')->schema([

RichEditor::make('justification\_besoin')->required(),

Select::make('urgence')->options([

'normale' => 'Normale',

'urgente' => 'Urgente',

'critique' => 'Critique'

]),

DatePicker::make('date\_besoin')

]),

Section::make('Fournisseur')->schema([

TextInput::make('fournisseur\_propose')->required(),

FileUpload::make('devis\_pdf')

->acceptedFileTypes(['application/pdf'])

->directory('devis')

])

]);

}

}

5. COMPOSANTS LIVEWIRE SPÉCIALISÉS

**Dashboard Budget en Temps Réel :**

php

**class** BudgetDashboard **extends** Component

{

**public** $service\_id;

**public** $periode = 'current\_year';

**public** **function** render()

{

$budget\_total = BudgetLigne::where('service\_id', $this->service\_id)

->sum('montant\_ht\_prevu');

$consomme = DemandeDevis::whereHas('budgetLigne', **function**($q) {

$q->where('service\_id', $this->service\_id);

})->where('statut', 'approuve')

->sum('prix\_total\_ttc');

$disponible = $budget\_total - $consomme;

$taux\_consommation = ($consomme / $budget\_total) \* 100;

**return** view('livewire.budget-dashboard', compact(

'budget\_total', 'consomme', 'disponible', 'taux\_consommation'

));

}

}

**Timeline Workflow :**

php

**class** WorkflowTimeline **extends** Component

{

**public** DemandeDevis $demande;

**public** **function** render()

{

$activities = $this->demande->approvals()

->with('user')

->orderBy('created\_at')

->get();

**return** view('livewire.workflow-timeline', compact('activities'));

}

}

**Validation Groupée :**

php

**class** GroupValidation **extends** Component

{

**public** $selected\_demandes = [];

**public** $action = '';

**public** $commentaire = '';

**public** **function** validateSelected()

{

**foreach**($this->selected\_demandes **as** $demande\_id) {

$demande = DemandeDevis::find($demande\_id);

**if**($this->action === 'approve') {

$demande->approve(auth()->user(), $this->commentaire);

} **elseif**($this->action === 'reject') {

$demande->reject(auth()->user(), $this->commentaire);

}

}

$this->emit('validation-completed');

}

}

6. SYSTÈME DE NOTIFICATIONS AVANCÉ

**Events et Listeners :**

php

**class** BudgetAlertEvent

{

**public** **function** \_\_construct(

**public** BudgetLigne $ligne,

**public** **float** $seuil\_depassement

) {}

}

**class** BudgetAlertListener

{

**public** **function** handle(BudgetAlertEvent $event)

{

*// Email au responsable service*

Mail::to($event->ligne->service->responsable)

->send(**new** BudgetDepassementAlert($event->ligne));

*// Notification responsable budget*

$responsable\_budget = User::role('responsable-budget')->first();

$responsable\_budget->notify(

**new** BudgetSeuilNotification($event->ligne)

);

*// Webhook externe si configuré*

**if**(config('budget.webhook\_url')) {

Http::post(config('budget.webhook\_url'), [

'type' => 'budget\_alert',

'service' => $event->ligne->service->nom,

'ligne' => $event->ligne->intitule,

'depassement' => $event->seuil\_depassement

]);

}

}

}

**Jobs de Relance Automatique :**

php

**class** RelanceFournisseurJob **implements** ShouldQueue

{

**public** **function** \_\_construct(

**public** Commande $commande,

**public** **int** $niveau\_relance = 1

) {}

**public** **function** handle()

{

$template = **match**($this->niveau\_relance) {

1 => 'emails.relance-fournisseur-1',

2 => 'emails.relance-fournisseur-2',

3 => 'emails.relance-fournisseur-escalade'

};

Mail::to($this->commande->fournisseur\_email)

->send(**new** RelanceFournisseur($this->commande, $template));

*// Programmer prochaine relance si pas de réponse*

**if**($this->niveau\_relance < 3) {

RelanceFournisseurJob::dispatch(

$this->commande,

$this->niveau\_relance + 1

)->delay(now()->addDays(3));

}

}

}

7. EXPORTS ET RAPPORTS SPÉCIALISÉS

**Export Budget Complet :**

php

**class** BudgetExport **implements** FromCollection, WithHeadings, WithStyles

{

**public** **function** \_\_construct(

**public** **int** $service\_id,

**public** **string** $periode

) {}

**public** **function** collection()

{

**return** BudgetLigne::where('service\_id', $this->service\_id)

->with(['service', 'demandesAssociees'])

->get()

->map(**function**($ligne) {

**return** [

'date\_prevue' => $ligne->date\_prevue->format('d/m/Y'),

'intitule' => $ligne->intitule,

'nature' => $ligne->nature,

'montant\_ht\_prevu' => $ligne->montant\_ht\_prevu,

'montant\_depense' => $ligne->demandesAssociees->sum('prix\_total\_ttc'),

'reste\_disponible' => $ligne->montant\_ht\_prevu - $ligne->demandesAssociees->sum('prix\_total\_ttc'),

'taux\_consommation' => ($ligne->demandesAssociees->sum('prix\_total\_ttc') / $ligne->montant\_ht\_prevu) \* 100,

'statut\_validation' => $ligne->valide\_budget

];

});

}

**public** **function** headings(): **array**

{

**return** [

'Date Prévue', 'Intitulé', 'Nature', 'Budget HT',

'Dépensé', 'Disponible', 'Taux (%)', 'Statut'

];

}

}

**Rapport Analyse Fournisseurs :**

php

**class** AnalyseFournisseursExport **implements** FromView

{

**public** **function** view(): View

{

$fournisseurs\_stats = DB::table('commandes')

->join('demande\_devis', 'commandes.demande\_devis\_id', '=', 'demande\_devis.id')

->select([

'demande\_devis.fournisseur\_propose',

DB::raw('COUNT(\*) as nb\_commandes'),

DB::raw('AVG(DATEDIFF(livraisons.date\_livraison, commandes.date\_livraison\_prevue)) as retard\_moyen'),

DB::raw('SUM(CASE WHEN livraisons.conforme = true THEN 1 ELSE 0 END) / COUNT(\*) \* 100 as taux\_conformite'),

DB::raw('SUM(commandes.montant\_reel) as ca\_total')

])

->leftJoin('livraisons', 'commandes.id', '=', 'livraisons.commande\_id')

->groupBy('demande\_devis.fournisseur\_propose')

->orderBy('ca\_total', 'desc')

->get();

**return** view('exports.analyse-fournisseurs', compact('fournisseurs\_stats'));

}

}

8. SÉCURITÉ ET CLOISONNEMENT

**Middleware Cloisonnement Service :**

php

**class** ServiceAccessMiddleware

{

**public** **function** handle($request, Closure $next, $resource = null)

{

$user = auth()->user();

*// Responsable budget : accès global*

**if**($user->hasRole('responsable-budget')) {

**return** $next($request);

}

*// Service demandeur : accès restreint à son service*

**if**($user->hasRole('service-demandeur')) {

$service\_id = $user->service\_id;

*// Restriction sur les routes budget*

**if**(str\_contains($request->route()->getName(), 'budget')) {

$request->merge(['service\_filter' => $service\_id]);

}

}

**return** $next($request);

}

}

**Policy Granulaire :**

php

**class** BudgetLignePolicy

{

**public** **function** viewAny(User $user)

{

**return** $user->hasAnyRole(['responsable-budget', 'service-demandeur']);

}

**public** **function** view(User $user, BudgetLigne $ligne)

{

**return** $user->hasRole('responsable-budget') ||

$user->service\_id === $ligne->service\_id;

}

**public** **function** update(User $user, BudgetLigne $ligne)

{

*// Seul responsable budget peut modifier*

**return** $user->hasRole('responsable-budget');

}

}

9. CONFIGURATION DOCKER POUR HÉBERGEMENT LOCAL

**docker-compose.yml Complet :**

text

version: '3.8'

services:

app:

build:

context: .

dockerfile: Dockerfile

ports:

- "80:80"

- "443:443"

environment:

- APP\_ENV=production

- DB\_HOST=postgres

- REDIS\_HOST=redis

- MAIL\_HOST=mailhog

volumes:

- ./storage:/var/www/html/storage

- ./bootstrap/cache:/var/www/html/bootstrap/cache

depends\_on:

- postgres

- redis

postgres:

image: postgres:15

environment:

POSTGRES\_DB: budget\_workflow

POSTGRES\_USER: workflow\_user

POSTGRES\_PASSWORD: ${DB\_PASSWORD}

volumes:

- postgres\_data:/var/lib/postgresql/data

- ./database/backups:/backups

ports:

- "5432:5432"

redis:

image: redis:7-alpine

command: redis-server --appendonly yes

volumes:

- redis\_data:/data

nginx:

image: nginx:alpine

ports:

- "8080:80"

volumes:

- ./nginx.conf:/etc/nginx/nginx.conf

- ./ssl:/etc/nginx/ssl

depends\_on:

- app

mailhog:

image: mailhog/mailhog

ports:

- "8025:8025"

volumes:

postgres\_data:

redis\_data:

**Script de Déploiement Automatisé :**

bash

#!/bin/bash

*# deploy.sh - Déploiement automatisé*

echo "🚀 Déploiement Application Budget & Workflow Achat"

*# Variables*

APP\_DIR="/var/www/budget-workflow"

BACKUP\_DIR="/backups/$(date +%Y%m%d\_%H%M%S)"

*# Sauvegarde avant déploiement*

echo "📦 Sauvegarde base de données..."

mkdir -p $BACKUP\_DIR

docker-compose exec postgres pg\_dump -U workflow\_user budget\_workflow > $BACKUP\_DIR/database.sql

*# Arrêt des services*

echo "⏹️ Arrêt des services..."

docker-compose down

*# Mise à jour code*

echo "📥 Mise à jour code source..."

git pull origin main

*# Build containers*

echo "🔨 Build containers..."

docker-compose build --no-cache

*# Démarrage services*

echo "▶️ Démarrage services..."

docker-compose up -d

*# Attente stabilisation*

sleep 30

*# Migrations et optimisations*

echo "🗄️ Migrations base de données..."

docker-compose exec app php artisan migrate --force

echo "⚡ Optimisations Laravel..."

docker-compose exec app php artisan config:cache

docker-compose exec app php artisan route:cache

docker-compose exec app php artisan view:cache

docker-compose exec app php artisan queue:restart

*# Permissions*

echo "🔐 Configuration permissions..."

docker-compose exec app chown -R www-data:www-data storage bootstrap/cache

docker-compose exec app chmod -R 775 storage bootstrap/cache

*# Tests post-déploiement*

echo "🧪 Tests post-déploiement..."

docker-compose exec app php artisan test --parallel

*# Notification fin déploiement*

echo "✅ Déploiement terminé avec succès !"

echo "📊 Application accessible : https://budget-workflow.local"

echo "📧 Interface mail : http://localhost:8025"

*# Nettoyage anciens backups (garde 30 jours)*

find /backups -type d -mtime +30 -exec rm -rf {} \;

10. ESTIMATION TEMPS ET PLANNING AVEC JULES

**Capacités de Jules pour ce Projet**

Jules (agent IA Gemini 2.5 Pro) peut automatiser 75-85% du développement grâce à :

* Analyse complète codebase Laravel sans limitations sandbox
* Génération automatique modèles Eloquent avec relations
* Création resources Filament avec interfaces complètes
* Configuration workflows Laravel Process Approval
* Composants Livewire réactifs pour dashboards
* Tests automatisés Pest pour validation fonctionnelle

**Planning Optimisé : 4-5 jours**

**Jour 1 - Foundation (0.5j humain, 90% Jules)**

* Setup Laravel 10 + packages spécialisés
* Configuration Docker locale avec PostgreSQL/Redis
* Modèles de données (Budget, DemandeDevis, Commande, Livraison)
* Migrations avec contraintes d'intégrité
* Seeders données de test

**Jour 2 - Workflow & Validation (0.5j humain, 85% Jules)**

* Configuration Laravel Process Approval 3 niveaux
* Events/Listeners notifications spécialisées
* Jobs relances automatiques fournisseurs
* Policies sécurité et cloisonnement services
* Middleware restrictions accès

**Jour 3 - Interfaces Filament (1j humain, 80% Jules)**

* Resources Budget avec calculs temps réel
* Resource DemandeDevis workflow intégré
* Dashboards personnalisés par rôle
* Composants Livewire (timeline, validation groupée)
* Upload/preview bons de livraison

**Jour 4 - Dashboards & Analytics (1j humain, 70% Jules)**

* Tableaux de bord exécutifs temps réel
* Exports Excel/PDF avec templates
* Analyses budgétaires et projections
* Performance fournisseurs avec KPIs
* Interface mobile responsive

**Jour 5 - Finalisation & Tests (1j humain, 60% Jules)**

* Charte graphique personnalisée
* Tests automatisés complets (Pest)
* Documentation utilisateur
* Formation équipe
* Mise en production

**Total : 4-5 jours (vs 25-30 jours développement traditionnel)**

**Commandes Démarrage Rapide Jules**

bash

*# Initialisation projet*

composer create-project laravel/laravel budget-workflow

cd budget-workflow

*# Installation packages spécialisés*

composer require ringlesoft/laravel-process-approval

composer require filament/filament

composer require spatie/laravel-medialibrary

composer require maatwebsite/excel

composer require spatie/laravel-permission

*# Configuration initiale*

php artisan vendor:publish --provider="RingleSoft\LaravelProcessApproval\ProcessApprovalServiceProvider"

php artisan filament:install --panels

php artisan migrate

php artisan make:filament-user

*# Génération resources métier*

php artisan make:filament-resource BudgetLigne

php artisan make:filament-resource DemandeDevis

php artisan make:filament-resource Commande

php artisan make:filament-resource Livraison

11. TESTS ET VALIDATION

**Tests Fonctionnels Pest :**

php

*// tests/Feature/BudgetWorkflowTest.php*

describe('Budget Workflow', **function** () {

test('service peut créer demande devis si budget disponible', **function** () {

$service = Service::factory()->create();

$budget = BudgetLigne::factory()->create([

'service\_id' => $service->id,

'montant\_ht\_prevu' => 1000,

'valide\_budget' => 'oui'

]);

$user = User::factory()->create(['service\_id' => $service->id]);

$user->assignRole('service-demandeur');

actingAs($user)

->post('/demandes-devis', [

'budget\_ligne\_id' => $budget->id,

'denomination' => 'Test Produit',

'prix\_total\_ttc' => 500,

'justification\_besoin' => 'Besoin urgent'

])

->assertStatus(201);

expect(DemandeDevis::count())->toBe(1);

});

test('demande rejetée si budget insuffisant', **function** () {

$budget = BudgetLigne::factory()->create([

'montant\_ht\_prevu' => 100,

'valide\_budget' => 'oui'

]);

$response = actingAs(User::factory()->create())

->post('/demandes-devis', [

'budget\_ligne\_id' => $budget->id,

'prix\_total\_ttc' => 500

]);

expect($response->status())->toBe(422);

expect($response->json('errors.budget'))->toContain('Budget insuffisant');

});

});

*// tests/Feature/WorkflowApprovalTest.php*

describe('Workflow Approval', **function** () {

test('workflow 3 niveaux fonctionne correctement', **function** () {

$demande = DemandeDevis::factory()->create();

$responsable\_budget = User::factory()->create();

$responsable\_budget->assignRole('responsable-budget');

*// Niveau 1 : Responsable Budget*

$demande->approve($responsable\_budget, 'Budget validé');

expect($demande->currentApprovalStep())->toBe('service-achat');

*// Niveau 2 : Service Achat*

$service\_achat = User::factory()->create();

$service\_achat->assignRole('service-achat');

$demande->approve($service\_achat, 'Commande validée');

expect($demande->isFullyApproved())->toBeTrue();

});

});

**Tests Performance :**

php

*// tests/Performance/DashboardPerformanceTest.php*

test('dashboard charges en moins de 2 secondes', **function** () {

*// Création 1000 lignes budget + 500 demandes*

BudgetLigne::factory(1000)->create();

DemandeDevis::factory(500)->create();

$start = microtime(true);

$response = actingAs(User::factory()->create())

->get('/dashboard');

$duration = microtime(true) - $start;

expect($duration)->toBeLessThan(2.0);

expect($response->status())->toBe(200);

});

12. INSTRUCTIONS SPÉCIALISÉES POUR AGENTS IA

**Contexte Rapide**

Application de gestion budgétaire avec workflow d'achat 3 niveaux utilisant Laravel Process Approval + Filament. Processus métier : Service demande devis → Responsable budget valide → Service achat valide → Commande → Livraison contrôlée. Cloisonnement strict par service. Hébergement local obligatoire.

**Modèles Métier Prioritaires**

1. **BudgetLigne** : 12 colonnes budget par service avec cloisonnement
2. **DemandeDevis** : Workflow 3 niveaux avec contrôle budget
3. **Commande** : Suivi commande et planning livraison
4. **Livraison** : Contrôle qualité avec upload bons de livraison

**Workflows Critiques**

* **Budget → Demande** : Vérification budget disponible automatique
* **Validation Budget** : Responsable budget voit toutes demandes tous services
* **Validation Achat** : Service achat optimise fournisseurs et commandes
* **Réception** : Service demandeur vérifie conformité livraison

**Sécurité Impérative**

* Cloisonnement : Service X voit uniquement son budget
* Responsable budget : accès global tous services
* Traçabilité complète : audit trail toutes actions
* RGPD : pas de données personnelles fournisseurs

**Commandes Essentielles**

bash

*# Création modèles avec workflow*

php artisan make:model BudgetLigne -mfr

php artisan make:model DemandeDevis -mfr --implements=Approvable

*# Resources Filament avec relations*

php artisan make:filament-resource BudgetLigne --generate

*# Composants dashboard temps réel*

php artisan make:livewire BudgetDashboard

php artisan make:livewire WorkflowTimeline

*# Tests métier*

php artisan make:test BudgetWorkflowTest --pest

**Architecture Attendue**

text

app/Models/

├── BudgetLigne.php (HasMany demandeDevis)

├── DemandeDevis.php (Approvable, BelongsTo budgetLigne)

├── Commande.php (BelongsTo demandeDevis, HasOne livraison)

├── Livraison.php (BelongsTo commande, HasMedia)

└── Service.php (HasMany budgetLignes, HasMany users)

app/Filament/Resources/

├── BudgetLigneResource.php (table calculée temps réel)

├── DemandeDevisResource.php (workflow intégré)

└── CommandeResource.php (suivi livraison)

app/Http/Livewire/

├── BudgetDashboard.php (jauges consommation)

├── ValidationQueue.php (demandes à traiter)

└── LivraisonTracker.php (suivi temps réel)

**Bonnes Pratiques Spécialisées**

* **Relations Eloquent** : Eager loading pour performance dashboards
* **Scopes** : Cloisonnement service via Global Scopes
* **Events** : Notifications budget automatiques sur seuils
* **Jobs** : Relances fournisseurs asynchrones
* **Cache** : Redis pour calculs budgétaires fréquents
* **Validation** : Rules custom pour cohérence budget-demande

**Packages Critiques Usage**

php

*// Laravel Process Approval*

**use** RingleSoft\LaravelProcessApproval\Traits\Approvable;

$demande->approve($user, 'Commentaire');

*// Filament Dashboard*

Stat::make('Budget Restant', **fn**() => $this->calculateBudgetRestant())

->color('success')

->chart([...]);

*// Spatie Media (bons livraison)*

$livraison->addMediaFromRequest('bon\_livraison')

->toMediaCollection('documents');

*// Laravel Excel export budget*

**return** Excel::download(**new** BudgetExport($service\_id), 'budget.xlsx');

13. ÉVOLUTIONS ET ROADMAP

**Version 1.0 - POC (4-5 jours)**

* ✅ Gestion budget 12 colonnes par service
* ✅ Workflow demande devis 3 niveaux
* ✅ Cloisonnement service strict
* ✅ Dashboard temps réel consommation
* ✅ Upload/vérification livraisons
* ✅ Exports Excel/PDF configurables

**Version 1.1 - Optimisations (+2 jours)**

* 📱 Application mobile complémentaire
* 🔔 Notifications push temps réel
* 📊 Analytics prédictifs IA
* 🔄 API REST complète
* 📈 Reporting dirigeants avancé

**Version 2.0 - Intégrations (+3-4 jours)**

* 💼 Connecteur ERP/Comptabilité
* 🏢 SSO LDAP/Active Directory
* 📋 Système GED externe
* 🤖 Chatbot assistant workflow
* 🔍 Recherche sémantique IA

**Intégrations Futures Prioritaires**

* **Sage/SAP** : Synchronisation écritures comptables
* **Teams/Slack** : Notifications workflow intégrées
* **DocuSign** : Signature électronique bons de commande
* **Power BI** : Tableaux de bord décisionnels
* **OCR IA** : Extraction automatique données factures

14. RESSOURCES ET DOCUMENTATION

**Documentation Technique**

* [Laravel Process Approval](https://github.com/ringlesoft/laravel-process-approval) - Workflow core
* [Filament](https://filamentphp.com/) - Interface admin moderne
* [Spatie Media Library](https://spatie.be/docs/laravel-medialibrary) - Gestion fichiers
* [Laravel Excel](https://laravel-excel.com/) - Exports Excel/CSV
* [Livewire](https://laravel-livewire.com/) - Composants réactifs

**Outils Développement Recommandés**

* **Jules (Agent IA)** : [https://jules.google](https://jules.google/) - Automatisation 80% code
* **Cursor + Claude** : IDE IA pour développement assisté
* **Laravel Herd** : Environnement local optimisé
* **Pest** : Framework test moderne Laravel

**Support et Communauté**

* **Laravel News** : Actualités packages et bonnes pratiques
* **Filament Discord** : Communauté support interface
* **Stack Overflow** : Tags #laravel-process-approval #filament
* **Laracasts** : Formations Laravel avancées

15. POINTS CLÉS FINAUX POUR AGENTS IA

**Résumé Exécutif**

Solution complète gestion budgétaire + workflow achat utilisant Laravel Process Approval + Filament. Processus métier spécialisé : préparation budget annuel → demande devis → validation budget → validation achat → commande → livraison contrôlée. Cloisonnement strict par service. Développement optimisé avec Jules (4-5 jours vs 25-30 jours traditionnel).

**Stack Technique Confirmée**

PHP 8.1+ | Laravel 10+ | PostgreSQL 15 | Redis | Docker | Laravel Process Approval | Filament | Spatie Media Library | Laravel Excel | Livewire | hébergement local obligatoire.

**Modèles Métier Critiques**

BudgetLigne (12 colonnes cloisonnées) → DemandeDevis (Approvable workflow 3 niveaux) → Commande (suivi livraison) → Livraison (contrôle qualité). Relations Eloquent optimisées. Sécurité granulaire par service.

**Interfaces Prioritaires**

Dashboard budgétaire temps réel avec jauges consommation. Workflow validation avec queue priorisée. Timeline visuelle avec codes couleur. Exports Excel configurables. Interface mobile responsive.

**Workflow Métier Impératif**

Service X saisit demande → Responsable Budget valide (contrôle enveloppe) → Service Achat valide (optimise fournisseur) → Commande passée → Livraison vérifiée par Service X → Budget mis à jour automatiquement.

**Document de référence complet pour agents IA - Solution Gestion Budgétaire & Workflow Achat**  
**Version 2.0 - Juin 2025**

**Usage recommandé** : Coller ce document en début de conversation avec tout agent IA pour contexte métier complet. Référencer les sections spécialisées selon la tâche (modèles, interfaces, sécurité, etc.). Mettre à jour avec évolutions et retours utilisateurs.