

PM

MODULE PROJECT MANAGEMENT

Spécification des Vues & Formulaires

VIEWS_FORMS_SPEC.md

HR_ONIAN · Spécification Technique · Février 2026

Spécifications des Vues et Formulaires - Module GAC

Version: 1.0

Date: 01/02/2026

Projet: HR_ONIAN

Module: Gestion des Achats & Commandes

Vue d'ensemble

Ce document spécifie toutes les vues (views) et formulaires (forms) du module GAC, organisés par fonctionnalité.

Organisation des vues

- **Class-Based Views (CBV)** pour les opérations CRUD standards
- **Function-Based Views (FBV)** pour les workflows spécifiques
- **Mixins personnalisés** pour la réutilisation du code
- **Formulaires Django** avec validation métier

1. URLs et routing

1.1 Structure des URLs

Fichier: `gestion_achats/urls.py`

```

from django.urls import path, include
from gestion_achats.views import (
    demande_views,
    bon_commande_views,
    fournisseur_views,
    reception_views,
    catalogue_views,
    budget_views,
    dashboard_views
)

app_name = 'gestion_achats'

urlpatterns = [
    # Dashboard
    path('', dashboard_views.DashboardView.as_view(), name='dashboard'),

    # Demandes d'achat
    path('demandes/', include([
        path('', demande_views.DemandeListView.as_view(), name='demande_list'),
        path('creer/', demande_views.DemandeCreateView.as_view(), name='demande_create'),
        path('<uuid:pk>/', demande_views.DemandeDetailView.as_view(), name='demande_detail'),
        path('<uuid:pk>/modifier/', demande_views.DemandeUpdateView.as_view(), name='demande_update'),
        path('<uuid:pk>/supprimer/', demande_views.DemandeDeleteView.as_view(), name='demande_delete'),
        path('<uuid:pk>/soumettre/', demande_views.soumettre_demande, name='demande_soumettre'),
        path('<uuid:pk>/valider-n1/', demande_views.valider_demande_n1, name='demande_valider_n1'),
        path('<uuid:pk>/valider-n2/', demande_views.valider_demande_n2, name='demande_valider_n2'),
        path('<uuid:pk>/refuser/', demande_views.refuser_demande, name='demande_refuser'),
        path('<uuid:pk>/annuler/', demande_views.annuler_demande, name='demande_annuler'),
        path('<uuid:pk>/convertir-bc/', demande_views.convertir_en_bc, name='demande_convertir_bc'),
        path('<uuid:pk>/ajouter-ligne/', demande_views.ajouter_ligne, name='demande_ajouter_ligne'),
        path('ligne/<uuid:pk>/modifier/', demande_views.modifier_ligne, name='demande_modifier_ligne'),
        path('ligne/<uuid:pk>/supprimer/', demande_views.supprimer_ligne, name='demande_supprimer_ligne'),
        path('mes-demandes/', demande_views.MesDemandesView.as_view(), name='mes_demandes'),
        path('a-valider/', demande_views.DemandesAValiderView.as_view(), name='demandes_a_valider'),
    ])),

    # Bons de commande
    path('bons-commande/', include([
        path('', bon_commande_views.BonCommandeListView.as_view(), name='bon_commande_list'),
        path('creer/', bon_commande_views.BonCommandeCreateView.as_view(), name='bon_commande_create'),
        path('<uuid:pk>/', bon_commande_views.BonCommandeDetailView.as_view(), name='bon_commande_detail'),
        path('<uuid:pk>/modifier/', bon_commande_views.BonCommandeUpdateView.as_view(), name='bon_commande_update'),
        path('<uuid:pk>/emettre/', bon_commande_views.emettre_bon_commande, name='bon_commande_emettre'),
        path('<uuid:pk>/envoyer/', bon_commande_views.envoyer_bon_commande, name='bon_commande_envoyer'),
        path('<uuid:pk>/confirmer/', bon_commande_views.confirmer_bon_commande, name='bon_commande_confirmer'),
        path('<uuid:pk>/annuler/', bon_commande_views.annuler_bon_commande, name='bon_commande_annuler'),
        path('<uuid:pk>/pdf/', bon_commande_views.telecharger_pdf, name='bon_commande_pdf'),
        path('<uuid:pk>/ajouter-ligne/', bon_commande_views.ajouter_ligne, name='bon_commande_ajouter_ligne'),
        path('ligne/<uuid:pk>/modifier/', bon_commande_views.modifier_ligne, name='bon_commande_modifier_ligne'),
        path('ligne/<uuid:pk>/supprimer/', bon_commande_views.supprimer_ligne, name='bon_commande_supprimer_ligne'),
    ])),

    # Fournisseurs
    path('fournisseurs/', include([
        path('', fournisseur_views.FournisseurListView.as_view(), name='fournisseur_list'),
        path('creer/', fournisseur_views.FournisseurCreateView.as_view(), name='fournisseur_create'),
        path('<uuid:pk>/', fournisseur_views.FournisseurDetailView.as_view(), name='fournisseur_detail'),
        path('<uuid:pk>/modifier/', fournisseur_views.FournisseurUpdateView.as_view(), name='fournisseur_update'),
        path('<uuid:pk>/desactiver/', fournisseur_views.desactiver_fournisseur, name='fournisseur_desactiver'),
        path('<uuid:pk>/evaluer/', fournisseur_views.evaluer_fournisseur, name='fournisseur_evaluer'),
    ])),

    # Réceptions
    path('receptions/', include([
        path('', reception_views.ReceptionListView.as_view(), name='reception_list'),
        path('creer/<uuid:bc_pk>/', reception_views.ReceptionCreateView.as_view(), name='reception_create'),
        path('<uuid:pk>/', reception_views.ReceptionDetailView.as_view(), name='reception_detail'),
        path('<uuid:pk>/modifier/', reception_views.ReceptionUpdateView.as_view(), name='reception_update'),
        path('<uuid:pk>/valider/', reception_views.valider_reception, name='reception_valider'),
        path('<uuid:pk>/annuler/', reception_views.annuler_reception, name='reception_annuler'),
    ])),

    # Catalogue
    path('catalogue/', include([
        path('articles/', catalogue_views.ArticleListView.as_view(), name='article_list'),
        path('articles/creer/', catalogue_views.ArticleCreateView.as_view(), name='article_create'),
        path('articles/<uuid:pk>/', catalogue_views.ArticleDetailView.as_view(), name='article_detail'),
        path('articles/<uuid:pk>/modifier/', catalogue_views.ArticleUpdateView.as_view(), name='article_update'),
        path('categories/', catalogue_views.CategorieListView.as_view(), name='categorie_list'),
        path('categories/creer/', catalogue_views.CategorieCreateView.as_view(), name='categorie_create'),
    ])),
]

```

```

# Budgets
path('budgets/', include([
    path('', budget_views.BudgetListView.as_view(), name='budget_list'),
    path('creer/', budget_views.BudgetCreateView.as_view(), name='budget_create'),
    path('<uuid:pk>/', budget_views.BudgetDetailView.as_view(), name='budget_detail'),
    path('<uuid:pk>/modifier/', budget_views.BudgetUpdateView.as_view(), name='budget_update'),
    path('<uuid:pk>/historique/', budget_views.BudgetHistoriqueView.as_view(), name='budget_historique'),
])),

# API AJAX
path('api/', include([
    path('articles/recherche/', catalogue_views.recherche_articles_ajax, name='api_recherche_articles'),
    path('fournisseurs/pour-article/<uuid:article_pk>/', fournisseur_views.fournisseurs_pour_article_ajax,
name='api_fournisseurs_article'),
])),
]
)

```

2. Vues des Demandes d'Achat

Fichier: `gestion_achats/views/demande_views.py`

2.1 DemandeListView

Type : ListView (CBV)

URL : `/gestion-achats/demandes/`

Template : `gestion_achats/demande/demande_list.html`

Permission : `@login_required`

Description : Liste toutes les demandes d'achat visibles par l'utilisateur

Contexte :

```
{
    'object_list': QuerySet[GACDemandeAchat],
    'statut_filter': str, # Filtre de statut appliqué
    'date_debut': date, # Filtre date début
    'date_fin': date, # Filtre date fin
    'stats': {
        'total': int,
        'par_statut': dict,
        'montant_total': Decimal
    }
}
```

Filtres disponibles :

- Par statut
- Par date de création
- Par demandeur
- Par budget
- Par montant (min/max)

Code :

```

class DemandeListView(LoginRequiredMixin, ListView):
    model = GACDemandeAchat
    template_name = 'gestion_achats/demande/demande_list.html'
    context_object_name = 'demandes'
    paginate_by = 20

    def get_queryset(self):
        qs = GACDemandeAchat.objects.select_related(
            'demandeur', 'validateur_n1', 'validateur_n2', 'budget'
        ).prefetch_related('lignes_article')

        # Filtrer selon les permissions
        user = self.request.user

        if not (user.has_role('ACHETEUR') or user.has_role('ADMIN_GAC')):
            # L'utilisateur ne voit que ses demandes et celles à valider
            qs = qs.filter(
                Q(demandeur=user) |
                Q(validateur_n1=user) |
                Q(validateur_n2=user)
            )

        # Filtres
        statut = self.request.GET.get('statut')
        if statut:
            qs = qs.filter(statut=statut)

        date_debut = self.request.GET.get('date_debut')
        if date_debut:
            qs = qs.filter(date_creation__gte=date_debut)

        date_fin = self.request.GET.get('date_fin')
        if date_fin:
            qs = qs.filter(date_creation__lte=date_fin)

        return qs.order_by('-date_creation')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        # Statistiques
        qs = self.get_queryset()
        context['stats'] = {
            'total': qs.count(),
            'par_statut': qs.values('statut').annotate(count=Count('id')),
            'montant_total': qs.aggregate(Sum('montant_total_ttc'))['montant_total_ttc__sum'] or 0
        }

        return context

```

2.2 DemandeCreateView

Type : CreateView (CBV)

URL : /gestion-achats/demandes/creer/

Template : gestion_achats/demande/demande_form.html

Permission : @login_required

Formulaire : DemandeCreateForm

Description : Créer une nouvelle demande d'achat en brouillon

Processus :

1. Afficher le formulaire de création
2. Sélectionner le budget (optionnel)
3. Lier au projet si applicable
4. Créer la demande en statut BROUILLON
5. Rediriger vers la page de détail pour ajouter des lignes

Code :

```

class DemandeCreateView(LoginRequiredMixin, CreateView):
    model = GACDemandeAchat
    form_class = DemandeCreateForm
    template_name = 'gestion_achats/demande/demande_form.html'

    def form_valid(self, form):
        # Créer la demande via le service
        demande = DemandeService.creer_demande_brouillon(
            demandeur=self.request.user,
            objet=form.cleaned_data['objet'],
            justification=form.cleaned_data['justification'],
            departement=form.cleaned_data.get('departement'),
            projet=form.cleaned_data.get('projet'),
            budget=form.cleaned_data.get('budget')
        )

        messages.success(self.request, f"Demande {demande.numero} créée en brouillon")

        return redirect('gestion_achats:demande_detail', pk=demande.pk)

```

2.3 DemandeDetailView

Type : DetailView (CBV)

URL : /gestion-achats/demandes/<uuid>/

Template : gestion_achats/demande/demande_detail.html

Permission : @login_required + vérification visibilité

Description : Affiche le détail d'une demande avec toutes ses lignes et son historique

Contexte :

```
{
    'object': GACDemandeAchat,
    'lignes': QuerySet[GACLigneDemandeAchat],
    'historique': QuerySet[GACHistorique],
    'peut_modifier': bool,
    'peut soumettre': bool,
    'peut_valider_n1': bool,
    'peut_valider_n2': bool,
    'peut_refuser': bool,
    'peut_convertir_bc': bool,
    'bon_commande': GACBonCommande (si existe)
}
```

Code :

```

class DemandeDetailView(LoginRequiredMixin, DetailView):
    model = GACDemandeAchat
    template_name = 'gestion_achats/demande/demande_detail.html'
    context_object_name = 'demande'

    def get_queryset(self):
        qs = super().get_queryset()
        user = self.request.user

        # Filtrer selon visibilité
        if not (user.has_role('ACHETEUR') or user.has_role('ADMIN_GAC')):
            qs = qs.filter(
                Q(demandeur=user) |
                Q(validateur_n1=user) |
                Q(validateur_n2=user)
            )

        return qs.select_related(
            'demandeur', 'validateur_n1', 'validateur_n2', 'budget', 'bon_commande'
        ).prefetch_related('lignes_article')

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        demande = self.object
        user = self.request.user

        # Permissions
        context['peut_modifier'] = (
            demande.statut == 'BROUILLON' and
            demande.demandeur == user
        )

        context['peut_soumettre'] = (
            demande.statut == 'BROUILLON' and
            demande.demandeur == user and
            demande.lignes.exists()
        )

        context['peut_valider_n1'] = (
            demande.statut == 'SOUMISE' and
            demande.validateur_n1 == user
        )

        context['peut_valider_n2'] = (
            demande.statut == 'VALIDEE_N1' and
            (demande.validateur_n2 == user or user.has_role('ACHETEUR'))
        )

        context['peut_refuser'] = (
            context['peut_valider_n1'] or context['peut_valider_n2']
        )

        context['peut_convertir_bc'] = (
            demande.statut == 'VALIDEE_N2' and
            user.has_role('ACHETEUR')
        )

        # Historique
        context['historique'] = HistoriqueService.get_historique(demande)

        return context

```

2.4 soumettre_demande (FBV)

Type : Function-Based View

URL : /gestion-achats/demandes/<uuid>/soumettre/

Méthode HTTP : POST

Permission : Demandeur uniquement

Description : Soumet une demande pour validation N1

Code :

```

@login_required
@require_POST
def soumettre_demande(request, pk):
    demande = get_object_or_404(GACDemandeAchat, pk=pk)

    # Vérifier que l'utilisateur est le demandeur
    if demande.demandeur != request.user:
        messages.error(request, "Vous n'êtes pas le demandeur de cette demande")
        return redirect('gestion_achats:demande_detail', pk=pk)

    # Vérifier le statut
    if demande.statut != 'BROUILLON':
        messages.error(request, "Cette demande ne peut pas être soumise")
        return redirect('gestion_achats:demande_detail', pk=pk)

    try:
        DemandeService.soumettre_demande(demande, request.user)
        messages.success(request, f"Demande {demande.numero} soumise pour validation")
    except ValidationError as e:
        messages.error(request, str(e))

    return redirect('gestion_achats:demande_detail', pk=pk)

```

2.5 valider_demande_n1 (FBV)

Type : Function-Based View

URL : /gestion-achats/demandes/<uuid>/valider-n1/

Méthode HTTP : POST

Permission : Validateur N1 uniquement

Formulaire : ValidationN1Form (avec commentaire optionnel)

Description : Valide une demande au niveau N1

Code :

```

@login_required
@require_http_methods(["GET", "POST"])
def valider_demande_n1(request, pk):
    demande = get_object_or_404(GACDemandeAchat, pk=pk)

    # Vérifier permissions
    if demande.validateur_n1 != request.user:
        messages.error(request, "Vous n'êtes pas le validateur N1 de cette demande")
        return redirect('gestion_achats:demande_detail', pk=pk)

    if request.method == 'POST':
        form = ValidationN1Form(request.POST)
        if form.is_valid():
            try:
                DemandeService.valider_n1(
                    demande,
                    request.user,
                    form.cleaned_data.get('commentaire')
                )
                messages.success(request, f"Demande {demande.numero} validée (N1)")
                return redirect('gestion_achats:demandes_a_valider')
            except (ValidationError, PermissionDenied) as e:
                messages.error(request, str(e))
        else:
            form = ValidationN1Form()

    return render(request, 'gestion_achats/demande/valider_n1.html', {
        'demande': demande,
        'form': form
    })

```

2.6 valider_demande_n2 (FBV)

Type : Function-Based View

URL : /gestion-achats/demandes/<uuid>/valider-n2/

Méthode HTTP : POST**Permission :** Validateur N2 ou rôle ACHETEUR**Formulaire :** ValidationN2Form**Description :** Valide une demande au niveau N2**Code similaire à valider_n1** en appelant DemandeService.valider_n2()**2.7 refuser_demande (FBV)****Type :** Function-Based View**URL :** /gestion-achats/demandes/<uuid>/refuser/**Méthode HTTP :** POST**Permission :** Validateur N1 ou N2 selon l'état**Formulaire :** RefusDemandeForm (avec motif obligatoire)**Description :** Refuse une demande**Code :**

```

@login_required
@require_http_methods(["GET", "POST"])
def refuser_demande(request, pk):
    demande = get_object_or_404(GACDemandeAchat, pk=pk)

    if request.method == 'POST':
        form = RefusDemandeForm(request.POST)
        if form.is_valid():
            try:
                DemandeService.refuser_demande(
                    demande,
                    request.user,
                    form.cleaned_data['motif_refus']
                )
                messages.success(request, f"Demande {demande.numero} refusée")
                return redirect('gestion_achats:demandes_a_valider')
            except (ValidationError, PermissionDenied) as e:
                messages.error(request, str(e))
        else:
            form = RefusDemandeForm()

    return render(request, 'gestion_achats/demande/refuser.html', {
        'demande': demande,
        'form': form
    })

```

2.8 convertir_en_bc (FBV)**Type :** Function-Based View**URL :** /gestion-achats/demandes/<uuid>/convertir-bc/**Méthode HTTP :** POST**Permission :** Rôle ACHETEUR uniquement**Formulaire :** ConvertirBCForm (sélection fournisseur, date livraison)**Description :** Convertit une demande validée en bon de commande**Code :**

```

@login_required
@require_http_methods(["GET", "POST"])
@user_has_role('ACHETEUR')
def convertir_en_bc(request, pk):
    demande = get_object_or_404(GACDemandeAchat, pk=pk, statut='VALIDEE_N2')

    if request.method == 'POST':
        form = ConvertirBCForm(request.POST)
        if form.is_valid():
            try:
                bc = DemandeService.convertir_en_bon_commande(
                    demande,
                    request.user,
                    form.cleaned_data['fournisseur'],
                    form.cleaned_data.get('date_livraison_souhaitee')
                )
                messages.success(request, f"BC {bc.numero} créé depuis la demande {demande.numero}")
                return redirect('gestion_achats:bon_commande_detail', pk=bc.pk)
            except (ValidationError, PermissionDenied) as e:
                messages.error(request, str(e))
        else:
            form = ConvertirBCForm()

    return render(request, 'gestion_achats/demande/convertir_bc.html', {
        'demande': demande,
        'form': form
    })

```

2.9 ajouter_ligne (FBV)

Type : Function-Based View + AJAX

URL : /gestion-achats/demandes/<uuid>/ajouter-ligne/

Méthode HTTP : POST

Permission : Demandeur, demande en BROUILLON

Formulaire : LigneDemandeForm

Description : Ajoute une ligne à une demande

Code :

```

@login_required
@require_POST
def ajouter_ligne(request, pk):
    demande = get_object_or_404(GACDemandeAchat, pk=pk)

    # Vérifications
    if demande.demandeur != request.user:
        return JsonResponse({'error': 'Permission refusée'}, status=403)

    if demande.statut != 'BROUILLON':
        return JsonResponse({'error': 'Demande non modifiable'}, status=400)

    form = LigneDemandeForm(request.POST)
    if form.is_valid():
        try:
            ligne = DemandeService.ajouter_ligne(
                demande,
                form.cleaned_data['article'],
                form.cleaned_data['quantite'],
                form.cleaned_data['prix_unitaire'],
                form.cleaned_data.get('commentaire')
            )
        except ValidationError as e:
            return JsonResponse({'error': str(e)}, status=400)
        else:
            return JsonResponse({
                'success': True,
                'ligne': {
                    'id': str(ligne.pk),
                    'article': ligne.article.designation,
                    'quantite': float(ligne.quantite),
                    'prix_unitaire': float(ligne.prix_unitaire),
                    'montant': float(ligne.montant)
                },
                'total_ttc': float(demande.montant_total_ttc)
            })
    else:
        return JsonResponse({'errors': form.errors}, status=400)

```

2.10 MesDemandesView

Type : ListView (CBV)

URL : /gestion-achats/demandes/mes-demandes/

Template : gestion_achats/demande/mes_demandes.html

Permission : @login_required

Description : Liste des demandes créées par l'utilisateur connecté

2.11 DemandesAValiderView

Type : ListView (CBV)

URL : /gestion-achats/demandes/a-valider/

Template : gestion_achats/demande/a_valider.html

Permission : @login_required

Description : Liste des demandes en attente de validation par l'utilisateur

Code :

```

class DemandesAValiderView(LoginRequiredMixin, ListView):
    model = GACDemandeAchat
    template_name = 'gestion_achats/demande/a_valider.html'
    context_object_name = 'demandes'

    def get_queryset(self):
        return DemandeService.get_demandes_en_attente_validation(self.request.user)

```

3. Vues des Bons de Commande

Fichier: `gestion_achats/views/bon_commande_views.py`

3.1 BonCommandeListView

Type : ListView (CBV)

URL : `/gestion-achats/bons-commande/`

Template : `gestion_achats/bon_commande/bon_commande_list.html`

Permission : Rôle ACHETEUR ou ADMIN_GAC

Description : Liste de tous les bons de commande

Filtres :

- Par statut
- Par fournisseur
- Par date d'émission
- Par acheteur
- Par montant

3.2 BonCommandeDetailView

Type : DetailView (CBV)

URL : `/gestion-achats/bons-commande/<uuid>/`

Template : `gestion_achats/bon_commande/bon_commande_detail.html`

Permission : ACHETEUR ou demandeur de la DA liée

Contexte :

```
{
    'object': GACBonCommande,
    'lignes': QuerySet[GACLigneBonCommande],
    'receptions': QuerySet[GACReception],
    'historique': QuerySet[GACHistorique],
    'peut_emettre': bool,
    'peut_envoyer': bool,
    'peut_confirmer': bool,
    'peut_annuler': bool,
    'taux_reception': float # % de marchandises reçues
}
```

3.3 emettre_bon_commande (FBV)

Type : Function-Based View

URL : `/gestion-achats/bons-commande/<uuid>/emettre/`

Méthode HTTP : POST

Permission : Rôle ACHETEUR

Description : Finalise et émet un BC (génère le PDF)

Code :

```

@login_required
@require_POST
@user_has_role('ACHETEUR')
def emettre_bon_commande(request, pk):
    bc = get_object_or_404(GACBonCommande, pk=pk)

    try:
        BonCommandeService.emettre_bon_commande(bc, request.user)
        messages.success(request, f"Bon de commande {bc.numero} émis avec succès")
    except ValidationError as e:
        messages.error(request, str(e))

    return redirect('gestion_achats:bon_commande_detail', pk=pk)

```

3.4 envoyer_bon_commande (FBV)

Type : Function-Based View

URL : /gestion-achats/bons-commande/<uuid>/envoyer/

Méthode HTTP : POST

Permission : Rôle ACHETEUR

Formulaire : EnvoiBCForm (email destinataire optionnel)

Description : Envoie le BC par email au fournisseur

3.5 confirmer_bon_commande (FBV)

Type : Function-Based View

URL : /gestion-achats/bons-commande/<uuid>/confirmer/

Méthode HTTP : POST

Permission : Rôle ACHETEUR

Formulaire : ConfirmationBCForm (numéro confirmation, date livraison)

Description : Enregistre la confirmation du fournisseur

3.6 telecharger_pdf (FBV)

Type : Function-Based View

URL : /gestion-achats/bons-commande/<uuid>/pdf/

Méthode HTTP : GET

Permission : ACHETEUR ou demandeur de la DA liée

Description : Télécharge le PDF du BC

Code :

```

@login_required
def telecharger_pdf(request, pk):
    bc = get_object_or_404(GACBonCommande, pk=pk)

    # Vérifier permissions
    user = request.user
    if not (user.has_role('ACHETEUR') or
            (bc.demande_achat and bc.demande_achat.demandeur == user)):
        raise PermissionDenied

    # Vérifier que le PDF existe
    if not bc.fichier_pdf:
        messages.error(request, "Aucun PDF généré pour ce BC")
        return redirect('gestion_achats:bon_commande_detail', pk=pk)

    # Retourner le fichier
    response = FileResponse(bc.fichier_pdf.open('rb'), content_type='application/pdf')
    response['Content-Disposition'] = f'attachment; filename="BC_{bc.numero}.pdf"'
    return response

```

4. Vues des Fournisseurs

Fichier: `gestion_achats/views/fournisseur_views.py`

4.1 FournisseurListView

Type : ListView (CBV)

URL : `/gestion-achats/fournisseurs/`

Template : `gestion_achats/fournisseur/fournisseur_list.html`

Permission : ACHETEUR ou ADMIN_GAC

Filtres :

- Par statut (actif/inactif)
- Par catégorie
- Par note d'évaluation
- Recherche par nom/SIRET

4.2 FournisseurDetailView

Type : DetailView (CBV)

URL : `/gestion-achats/fournisseurs/<uuid>/`

Template : `gestion_achats/fournisseur/fournisseur_detail.html`

Permission : ACHETEUR ou ADMIN_GAC

Contexte :

```

{
    'object': GACFournisseur,
    'bons_commande': QuerySet[GACBonCommande],
    'evaluations': QuerySet[GACEvaluationFournisseur],
    'statistiques': {
        'nombre_commandes': int,
        'montant_total': Decimal,
        'taux_livraison_temps': float,
        'evaluation_moyenne': float
    }
}

```

4.3 evaluer_fournisseur (FBV)

Type : Function-Based View

URL : `/gestion-achats/fournisseurs/<uuid>/evaluer/`

Méthode HTTP : POST**Permission :** Rôle ACHETEUR**Formulaire :** EvaluationFournisseurForm**Description :** Évalue un fournisseur après une commande**Champs du formulaire :**

- note_qualite (1-5)
- note_delai (1-5)
- note_prix (1-5)
- commentaire (optionnel)

5. Vues des Réceptions

Fichier: gestion_achats/views/reception_views.py

5.1 ReceptionCreateView

Type : CreateView (CBV)**URL :** /gestion-achats/receptions/creer/<bc_uuid>/**Template :** gestion_achats/reception/reception_form.html**Permission :** Rôle RECEPTIONNAIRE ou ACHETEUR**Description :** Crée une réception pour un BC**Processus :**

1. Afficher le formulaire avec les lignes du BC pré-remplies
2. Saisir les quantités reçues/acceptées/refusées pour chaque ligne
3. Marquer la conformité de chaque ligne
4. Créer la réception

5.2 valider_reception (FBV)

Type : Function-Based View**URL :** /gestion-achats/receptions/<uuid>/valider/**Méthode HTTP :** POST**Permission :** Rôle RECEPTIONNAIRE ou ACHETEUR**Description :** Valide une réception et met à jour le BC et le budget

6. Vues du Catalogue

Fichier: gestion_achats/views/catalogue_views.py

6.1 ArticleListView

Type : ListView (CBV)**URL :** /gestion-achats/catalogue/articles/**Template :** gestion_achats/catalogue/article_list.html**Permission :** @login_required**Filtres :**

- Par catégorie
- Par statut (actif/inactif)
- Recherche par référence/désignation

6.2 recherche_articles_ajax (FBV)

Type : API AJAX

URL : /gestion-achats/api/articles/recherche/

Méthode HTTP : GET

Permission : @login_required

Paramètres :

- `q` : Terme de recherche

Retour JSON :

```
[  
    {  
        "id": "uuid",  
        "reference": "ART-001",  
        "designation": "Ordinateur portable Dell",  
        "prix_unitaire": 1200.00,  
        "unite": "pièce"  
    },  
    ...  
]
```

Utilisation : Autocomplétion dans les formulaires de demande/BC

7. Vues des Budgets

Fichier: gestion_achats/views/budget_views.py

7.1 BudgetListView

Type : ListView (CBV)

URL : /gestion-achats/budgets/

Template : gestion_achats/budget/budget_list.html

Permission : Rôle GESTIONNAIRE_BUDGET ou ADMIN_GAC

7.2 BudgetDetailView

Type : DetailView (CBV)

URL : /gestion-achats/budgets/<uuid>/

Template : gestion_achats/budget/budget_detail.html

Permission : Gestionnaire du budget ou ADMIN_GAC

Contexte :

```
{  
    'object': GACBudget,  
    'demandes': QuerySet[GACDemandeAchat],  
    'bons_commande': QuerySet[GACBonCommande],  
    'historique_mouvements': list,  
    'graphique_consommation': dict # Données pour Chart.js  
}
```

8. Dashboard

Fichier: gestion_achats/views/dashboard_views.py

8.1 DashboardView

Type : TemplateView (CBV)

URL : /gestion-achats/

Template : gestion_achats/dashboard/dashboard.html

Permission : @login_required

Contexte :

```
{
    'stats_demandes': {
        'total': int,
        'en_attente': int,
        'validees': int,
        'montant_total': Decimal
    },
    'stats_bons_commande': {
        'total': int,
        'en_cours': int,
        'montant_total': Decimal
    },
    'stats_budgets': {
        'total_disponible': Decimal,
        'taux_consommation': float,
        'budgets_en_alerte': int
    },
    'demandes_a_valider': QuerySet[GACDemandeAchat],
    'bons_commande_recents': QuerySet[GACBonCommande],
    'alertes_budgetaires': QuerySet[GACBudget]
}
```

9. Formulaires

Fichier: gestion_achats/forms/demande_forms.py

9.1 DemandeCreateForm

```
class DemandeCreateForm(forms.ModelForm):
    class Meta:
        model = GACDemandeAchat
        fields = ['objet', 'justification', 'priorite', 'departement', 'projet', 'budget']
        widgets = {
            'justification': forms.Textarea(attrs={'rows': 4}),
        }

    def clean(self):
        cleaned_data = super().clean()

        # Validation personnalisée
        if not cleaned_data.get('objet'):
            raise forms.ValidationError("L'objet de la demande est obligatoire")

        return cleaned_data
```

9.2 LigneDemandeForm

```
class LigneDemandeForm(forms.ModelForm):
    article = forms.ModelChoiceField(
        queryset=GACArticle.objects.filter(statut='ACTIF'),
        widget=forms.Select(attrs={'class': 'select2'})
    )

    class Meta:
        model = GACLigneDemandeAchat
        fields = ['article', 'quantite', 'prix_unitaire', 'commentaire']
        widgets = {
            'commentaire': forms.Textarea(attrs={'rows': 2}),
        }

    def clean_quantite(self):
        quantite = self.cleaned_data.get('quantite')
        if quantite <= 0:
            raise forms.ValidationError("La quantité doit être supérieure à 0")
        return quantite

    def clean_prix_unitaire(self):
        prix = self.cleaned_data.get('prix_unitaire')
        if prix <= 0:
            raise forms.ValidationError("Le prix doit être supérieur à 0")
        return prix
```

9.3 ValidationN1Form

```
class ValidationN1Form(forms.Form):
    commentaire = forms.CharField(
        required=False,
        widget=forms.Textarea(attrs={'rows': 3, 'placeholder': 'Commentaire optionnel'}),
        label="Commentaire"
    )
```

9.4 RefusDemandeForm

```
class RefusDemandeForm(forms.Form):
    motif_refus = forms.CharField(
        required=True,
        widget=forms.Textarea(attrs={'rows': 4, 'placeholder': 'Indiquez le motif du refus'}),
        label="Motif du refus"
    )

    def clean_motif_refus(self):
        motif = self.cleaned_data.get('motif_refus')
        if len(motif) < 10:
            raise forms.ValidationError("Le motif doit contenir au moins 10 caractères")
        return motif
```

9.5 ConvertirBCForm

```
class ConvertirBCForm(forms.Form):
    fournisseur = forms.ModelChoiceField(
        queryset=GACFournisseur.objects.filter(statut='ACTIF'),
        label="Fournisseur"
    )

    date_livraison_souhaitee = forms.DateField(
        required=False,
        widget=forms.DateInput(attrs={'type': 'date'}),
        label="Date de livraison souhaitée"
    )

    def clean_date_livraison_souhaitee(self):
        date_livraison = self.cleaned_data.get('date_livraison_souhaitee')
        if date_livraison and date_livraison < timezone.now().date():
            raise forms.ValidationError("La date de livraison ne peut pas être dans le passé")
        return date_livraison
```

9.6 ReceptionForm

```

class ReceptionForm(forms.ModelForm):
    class Meta:
        model = GACReception
        fields = ['date_reception', 'receptionnaire', 'commentaire']
        widgets = {
            'date_reception': forms.DateInput(attrs={'type': 'date'}),
            'commentaire': forms.Textarea(attrs={'rows': 3}),
        }

class LigneReceptionForm(forms.ModelForm):
    class Meta:
        model = GACLigneReception
        fields = ['quantite_recue', 'quantite_accepsee', 'quantite_refusee', 'conforme', 'commentaire_reception']
        widgets = {
            'commentaire_reception': forms.Textarea(attrs={'rows': 2}),
        }

    def clean(self):
        cleaned_data = super().clean()
        quantite_recue = cleaned_data.get('quantite_recue')
        quantite_accepsee = cleaned_data.get('quantite_accepsee')
        quantite_refusee = cleaned_data.get('quantite_refusee')

        if quantite_accepsee + quantite_refusee != quantite_recue:
            raise forms.ValidationError(
                "La somme des quantités acceptée et refusée doit égaler la quantité reçue"
            )

        return cleaned_data

```

9.7 EvaluationFournisseurForm

```

class EvaluationFournisseurForm(forms.Form):
    note_qualite = forms.IntegerField(
        min_value=1,
        max_value=5,
        widget=forms.RadioSelect(choices=[(i, i) for i in range(1, 6)]),
        label="Qualité des produits"
    )

    note_delai = forms.IntegerField(
        min_value=1,
        max_value=5,
        widget=forms.RadioSelect(choices=[(i, i) for i in range(1, 6)]),
        label="Respect des délais"
    )

    note_prix = forms.IntegerField(
        min_value=1,
        max_value=5,
        widget=forms.RadioSelect(choices=[(i, i) for i in range(1, 6)]),
        label="Compétitivité des prix"
    )

    commentaire = forms.CharField(
        required=False,
        widget=forms.Textarea(attrs={'rows': 3}),
        label="Commentaire"
    )

```

10. Mixins personnalisés

Fichier: `gestion_achats/mixins.py`

10.1 RoleRequiredMixin

```
class RoleRequiredMixin:
    """Mixin pour vérifier qu'un utilisateur a un rôle spécifique."""
    required_role = None

    def dispatch(self, request, *args, **kwargs):
        if not request.user.has_role(self.required_role):
            messages.error(request, "Vous n'avez pas les permissions nécessaires")
            return redirect('gestion_achats:dashboard')
        return super().dispatch(request, *args, **kwargs)

# Exemple d'utilisation
class BonCommandeCreateView(RoleRequiredMixin, CreateView):
    required_role = 'ACHETEUR'
    model = GACBonCommande
    ...
```

10.2 DemandeAccessMixin

```
class DemandeAccessMixin:
    """Mixin pour vérifier l'accès à une demande."""

    def dispatch(self, request, *args, **kwargs):
        demande = self.get_object()
        user = request.user

        # Accès autorisé si :
        # - Admin GAC
        # - Acheteur
        # - Demandeur
        # - Validateur N1/N2
        if not (user.has_role('ADMIN_GAC') or
                user.has_role('ACHETEUR') or
                demande.demandeur == user or
                demande.validateur_n1 == user or
                demande.validateur_n2 == user):
            raise PermissionDenied

        return super().dispatch(request, *args, **kwargs)
```

11. Décorateurs personnalisés

Fichier: `gestion_achats/decorators.py`

11.1 user_has_role

```
from functools import wraps
from django.core.exceptions import PermissionDenied

def user_has_role(role):
    """Décorateur pour vérifier qu'un utilisateur a un rôle spécifique."""
    def decorator(view_func):
        @wraps(view_func)
        def _wrapped_view(request, *args, **kwargs):
            if not request.user.has_role(role):
                raise PermissionDenied
            return view_func(request, *args, **kwargs)
        return _wrapped_view
    return decorator

# Exemple d'utilisation
@login_required
@user_has_role('ACHETEUR')
def emettre_bon_commande(request, pk):
    ...
```

Fin des spécifications des vues et formulaires