Le suivi de carrière interne, les reclassements et la mobilité interne et externe (départs en retraites, licenciements, décès...)". Cette fonctionnalité permettra de suivre l'évolution de carrière des employés au sein de l'entreprise et les différents mouvements de personnel.

Voici le code détaillé :

1. Modèles (ajoutez ceci à models.py) :

from django.db import models

from employe.models import Employe

class Poste(models.Model):

titre = models.CharField(max\_length=100)

description = models.TextField()

def \_\_str\_\_(self):

return self.titre

class EvolutionCarriere(models.Model):

TYPE\_EVOLUTION = [

('PROMOTION', 'Promotion'),

('MUTATION', 'Mutation'),

('RECLASSEMENT', 'Reclassement'),

]

employe = models.ForeignKey(Employe, on\_delete=models.CASCADE, related\_name='evolutions\_carriere')

date\_changement = models.DateField()

type\_evolution = models.CharField(max\_length=20, choices=TYPE\_EVOLUTION)

ancien\_poste = models.ForeignKey(Poste, on\_delete=models.SET\_NULL, null=True, related\_name='anciens\_postes')

nouveau\_poste = models.ForeignKey(Poste, on\_delete=models.SET\_NULL, null=True, related\_name='nouveaux\_postes')

commentaire = models.TextField(blank=True)

def \_\_str\_\_(self):

return f"{self.employe.nom} - {self.type\_evolution} - {self.date\_changement}"

class MouvementPersonnel(models.Model):

TYPE\_MOUVEMENT = [

('DEPART\_RETRAITE', 'Départ en retraite'),

('LICENCIEMENT', 'Licenciement'),

('DEMISSION', 'Démission'),

('DECES', 'Décès'),

('AUTRE', 'Autre'),

]

employe = models.ForeignKey(Employe, on\_delete=models.CASCADE, related\_name='mouvements')

date\_mouvement = models.DateField()

type\_mouvement = models.CharField(max\_length=20, choices=TYPE\_MOUVEMENT)

raison = models.TextField()

def \_\_str\_\_(self):

return f"{self.employe.nom} - {self.get\_type\_mouvement\_display()} - {self.date\_mouvement}"

1. Formulaires (ajoutez ceci à forms.py) :

from django import forms

from .models import EvolutionCarriere, MouvementPersonnel

class EvolutionCarriereForm(forms.ModelForm):

class Meta:

model = EvolutionCarriere

fields = ['employe', 'date\_changement', 'type\_evolution', 'ancien\_poste', 'nouveau\_poste', 'commentaire']

widgets = {

'date\_changement': forms.DateInput(attrs={'type': 'date'}),

}

class MouvementPersonnelForm(forms.ModelForm):

class Meta:

model = MouvementPersonnel

fields = ['employe', 'date\_mouvement', 'type\_mouvement', 'raison']

widgets = {

'date\_mouvement': forms.DateInput(attrs={'type': 'date'}),

}

1. Vues (ajoutez ceci à views.py) :

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib import messages

from .models import Employe, EvolutionCarriere, MouvementPersonnel

from .forms import EvolutionCarriereForm, MouvementPersonnelForm

def suivi\_carriere(request):

evolutions = EvolutionCarriere.objects.all().order\_by('-date\_changement')

mouvements = MouvementPersonnel.objects.all().order\_by('-date\_mouvement')

context = {

'evolutions': evolutions,

'mouvements': mouvements,

}

return render(request, 'gpec/suivi\_carriere.html', context)

def ajouter\_evolution(request):

if request.method == 'POST':

form = EvolutionCarriereForm(request.POST)

if form.is\_valid():

form.save()

messages.success(request, "Évolution de carrière ajoutée avec succès.")

return redirect('suivi\_carriere')

else:

form = EvolutionCarriereForm()

return render(request, 'gpec/ajouter\_evolution.html', {'form': form})

def ajouter\_mouvement(request):

if request.method == 'POST':

form = MouvementPersonnelForm(request.POST)

if form.is\_valid():

form.save()

messages.success(request, "Mouvement de personnel ajouté avec succès.")

return redirect('suivi\_carriere')

else:

form = MouvementPersonnelForm()

return render(request, 'gpec/ajouter\_mouvement.html', {'form': form})

def detail\_employe\_carriere(request, employe\_id):

employe = get\_object\_or\_404(Employe, id=employe\_id)

evolutions = employe.evolutions\_carriere.all().order\_by('-date\_changement')

mouvements = employe.mouvements.all().order\_by('-date\_mouvement')

context = {

'employe': employe,

'evolutions': evolutions,

'mouvements': mouvements,

}

return render(request, 'gpec/detail\_employe\_carriere.html', context)

1. URLs (ajoutez ceci à urls.py) :

from django.urls import path

from . import views

urlpatterns = [

# ... autres URLs ...

path('gpec/suivi-carriere/', views.suivi\_carriere, name='suivi\_carriere'),

path('gpec/ajouter-evolution/', views.ajouter\_evolution, name='ajouter\_evolution'),

path('gpec/ajouter-mouvement/', views.ajouter\_mouvement, name='ajouter\_mouvement'),

path('gpec/detail-employe-carriere/<int:employe\_id>/', views.detail\_employe\_carriere, name='detail\_employe\_carriere'),

]

1. Templates
   1. suivi\_carriere.html :

{% extends 'base.html' %}

{% block content %}

<h1>Suivi de Carrière et Mouvements de Personnel</h1>

<h2>Évolutions de Carrière</h2>

<a href="{% url 'ajouter\_evolution' %}" class="btn btn-primary mb-3">Ajouter une évolution</a>

<table class="table">

<thead>

<tr>

<th>Employé</th>

<th>Date</th>

<th>Type</th>

<th>Ancien Poste</th>

<th>Nouveau Poste</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{% for evolution in evolutions %}

<tr>

<td>{{ evolution.employe.nom }} {{ evolution.employe.prenom }}</td>

<td>{{ evolution.date\_changement }}</td>

<td>{{ evolution.get\_type\_evolution\_display }}</td>

<td>{{ evolution.ancien\_poste.titre }}</td>

<td>{{ evolution.nouveau\_poste.titre }}</td>

<td>

<a href="{% url 'detail\_employe\_carriere' evolution.employe.id %}" class="btn btn-info btn-sm">Détails</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<h2>Mouvements de Personnel</h2>

<a href="{% url 'ajouter\_mouvement' %}" class="btn btn-primary mb-3">Ajouter un mouvement</a>

<table class="table">

<thead>

<tr>

<th>Employé</th>

<th>Date</th>

<th>Type</th>

<th>Raison</th>

<th>Actions</th>

</tr>

</thead>

<tbody>

{% for mouvement in mouvements %}

<tr>

<td>{{ mouvement.employe.nom }} {{ mouvement.employe.prenom }}</td>

<td>{{ mouvement.date\_mouvement }}</td>

<td>{{ mouvement.get\_type\_mouvement\_display }}</td>

<td>{{ mouvement.raison|truncatewords:10 }}</td>

<td>

<a href="{% url 'detail\_employe\_carriere' mouvement.employe.id %}" class="btn btn-info btn-sm">Détails</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

{% endblock %}

* 1. ajouter\_evolution.html :

{% extends 'base.html' %}

{% block content %}

<h1>Ajouter une Évolution de Carrière</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Enregistrer</button>

</form>

{% endblock %}

* 1. ajouter\_mouvement.html :

{% extends 'base.html' %}

{% block content %}

<h1>Ajouter un Mouvement de Personnel</h1>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit" class="btn btn-primary">Enregistrer</button>

</form>

{% endblock %}

* 1. detail\_employe\_carriere.html :

{% extends 'base.html' %}

{% block content %}

<h1>Détails de Carrière - {{ employe.nom }} {{ employe.prenom }}</h1>

<h2>Évolutions de Carrière</h2>

<table class="table">

<thead>

<tr>

<th>Date</th>

<th>Type</th>

<th>Ancien Poste</th>

<th>Nouveau Poste</th>

<th>Commentaire</th>

</tr>

</thead>

<tbody>

{% for evolution in evolutions %}

<tr>

<td>{{ evolution.date\_changement }}</td>

<td>{{ evolution.get\_type\_evolution\_display }}</td>

<td>{{ evolution.ancien\_poste.titre }}</td>

<td>{{ evolution.nouveau\_poste.titre }}</td>

<td>{{ evolution.commentaire }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<h2>Mouvements de Personnel</h2>

<table class="table">

<thead>

<tr>

<th>Date</th>

<th>Type</th>

<th>Raison</th>

</tr>

</thead>

<tbody>

{% for mouvement in mouvements %}

<tr>

<td>{{ mouvement.date\_mouvement }}</td>

<td>{{ mouvement.get\_type\_mouvement\_display }}</td>

<td>{{ mouvement.raison }}</td>

</tr>

{% endfor %}

</tbody>

</table>

<a href="{% url 'suivi\_carriere' %}" class="btn btn-secondary">Retour au suivi de carrière</a>

{% endblock %}

N'oubliez pas d'effectuer les migrations après avoir ajouté ces nouveaux modèles :

python manage.py makemigrations

python manage.py migrate

Cette implémentation vous permet de :

1. Voir un résumé de toutes les évolutions de carrière et mouvements de personnel
2. Ajouter de nouvelles évolutions de carrière et mouvements de personnel
3. Voir les détails de carrière pour chaque employé

Assurez-vous d'ajouter un lien vers cette nouvelle fonctionnalité dans votre dashboard GPEC ou dans le menu de navigation principal.

Cette fonctionnalité vous donnera une vue d'ensemble claire des évolutions de carrière et des mouvements de personnel au sein de votre entreprise, ce qui est essentiel pour une gestion efficace des ressources humaines et pour la planification à long terme.