# Système de Gestion des Feedbacks de Recrutement

## Intégration ERP (Odoo) et Portail Web (Django) via XML-RPC

Réalisé par :
Noje Alessian & Opre Nicoleta

# Méthodologie de Développement : Pair Programming

**1** Ordinateur pour
**2** développeurs.

## Répartition des Rôles :

**Pilote (Driver)** : Écrit le code (Focus sur la syntaxe).

**Copilote (Navigator)** : Vérifie la logique et anticipe les erreurs.

## Avantages :

- Revue de code instantanée (Moins de bugs).
- Partage total de la connaissance (100% du code maîtrisé par le binôme).

# Architecture du Système

## Backend avec Odoo

- Cœur du système.
- Pour les recruteurs.

## Protocole XML-RPC

- Pont reliant les deux mondes.
- Requêtes sécurisées de Django à Odoo.
- 'Donne-moi les notes de ce candidat'.
- Odoo répond, Django affiche.

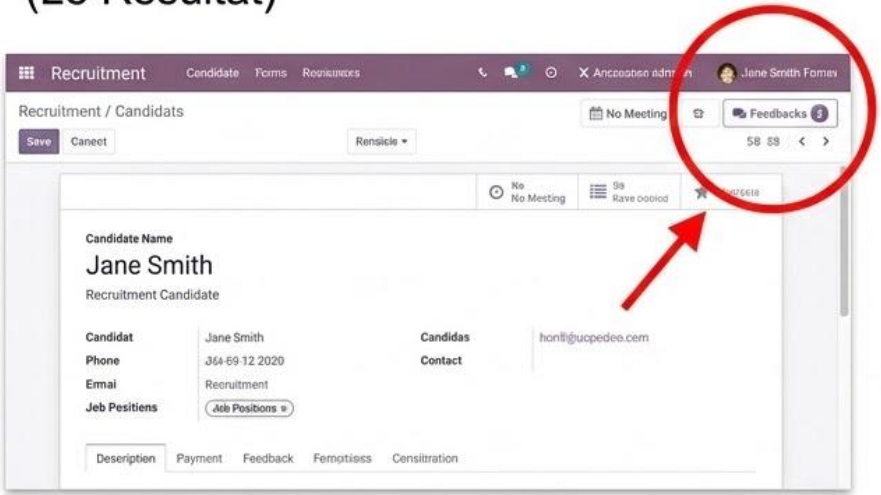
Requête / Réponse

## Frontend avec Django

- Interface web pour les candidats.
- Légère, ne stocke pas les feedbacks.
- Sert uniquement à l'affichage.

# Odoo UX : Le ‘Smart Button’ (Héritage de Vue)

(Le Résultat)




Accès direct aux feedbacks depuis la fiche Candidat

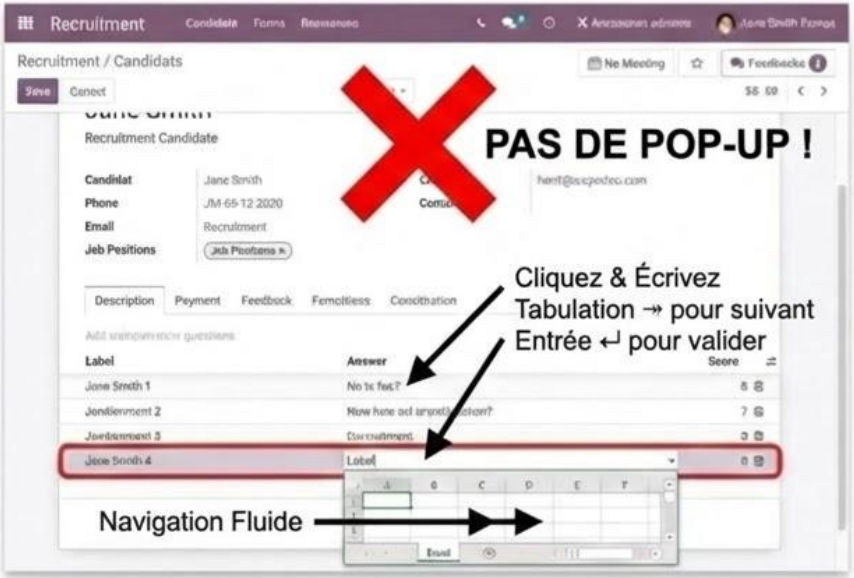
(Le Code XML)

Carbon.now.sh

```xml
<field name="inherit_id"
    ref="hr_recruitment.hr_applicant_view_form"/>

 <div name="button_box" position="inside">
   <button class="oe_stat_button" icon="fa-comments"
       name="action_get_attachment_tree_view"
       type="object">
      <field name="feedback_count" widget="statinfo"
          string="Feedbacks"/>
   </button>
 </div>
```

# Ergonomie : Saisie ‘Inline’ (One2Many)




Édition directe, rapide et sans interruption, comme dans un tableur.


L’attribut clé pour l’édition en ligne

# Logique Métier : Automatisation (@api.onchange)

## **Le Code** (Via Carbon.now.sh)

```python
@api.onchange('interview_type')
def _onchange_interview_type(self):
    """ Automatisation du remplissage des questions """

    # 1. Si le type est 'Technique'
    if self.interview_type == 'technical':
        # 2. Commande spéciale (5,0,0) pour vider la liste
        self.questions_ids = [(5, 0, 0)]

        # 3. Injection des questions par défaut
        self.questions_ids = [
            (0, 0, {'label': 'Python vs Java?', 'score': 0}),
            (0, 0, {'label': 'Architecture MVC?', 'score': 0}),
        ]
```

## **Le Schéma Fonctionnel**

# Champs Calculés (@api.depends)

```python
average_score = fields.Float(
    compute='_compute_average_score',
    store=True
)

@api.depends('questions_ids.score')
def _compute_average_score(self):
    """ Recalcule la moyenne dès qu'une note change """
    for record in self:
        if record.questions_ids:
            total = sum(q.score for q in record.questions_ids)
            record.average_score = total / len(record.questions_ids)
        else:
            record.average_score = 0.0
```

# Validation & Intégrité (@api.constrains)

```python
from odoo.exceptions import ValidationError

@api.constrains('state', 'questions_ids')
def _check_questions_filled(self):
    """ Interdit de valider l'entretien sans questions """
    for record in self:
        if record.state == 'done' and not record.questions_ids:
            raise ValidationError(
                "Impossible de valider un feedback sans questions \"
            )
```



- **Administration Django** : Gestion centralisée des paramètres (URL, DB, User).
- **Action Personnalisée :** Script de test intégré dans l'interface admin.
- **Validation XML-RPC** : Récupération de l'UID pour confirmer l'authentification.

# Filtrage Dynamique & Sécurité (XML-RPC)


```python
# 1. Sécurité d'abord : On filtre TOUJOURS par email du candidat
# Personne ne peut voir les données d'un autre.
search_domain = [['application_id.email_from', '=', candidate_email]]

# 2. Filtrage Dynamique : On récupère le choix du menu
filter_option = request.GET.get('filter')

if filter_option == 'high':
    # On AJOUTE une condition mathématique à la requête
    search_domain.append(['average_score', '>', 6])

elif filter_option == 'low':
    search_domain.append(['average_score', '<=', 6])
```

# Conclusion : Synthèse du Projet

- **Objectif Atteint** : Une solution Fullstack complète (Backend + Frontend).
- **Architecture Robuste** : Communication XML-RPC stable et sécurisée.
- **Expérience Recruteur** : Gain de temps grâce à l'automatisation (Smart Button, Saisie Inline).
- **Expérience Candidat** : Transparence et accès sécurisé aux données.
- **Méthodologie** : Efficacité validée du Pair Programming.