Vues

- Modifier une vue
- Empêcher qu'une vue soit écrasée lors d'une mise à jour
- Ajouter un champs - Odoo v16

Modifier une vue

Ce guide explique comment modifier un rapport ou une vue backend via l'interface de Odoo.
Pour comprendre le mécanisme plus en profondeur, je vous invite à lire la section "Extending Views" du chapître 4 du livre de dev Odoo.

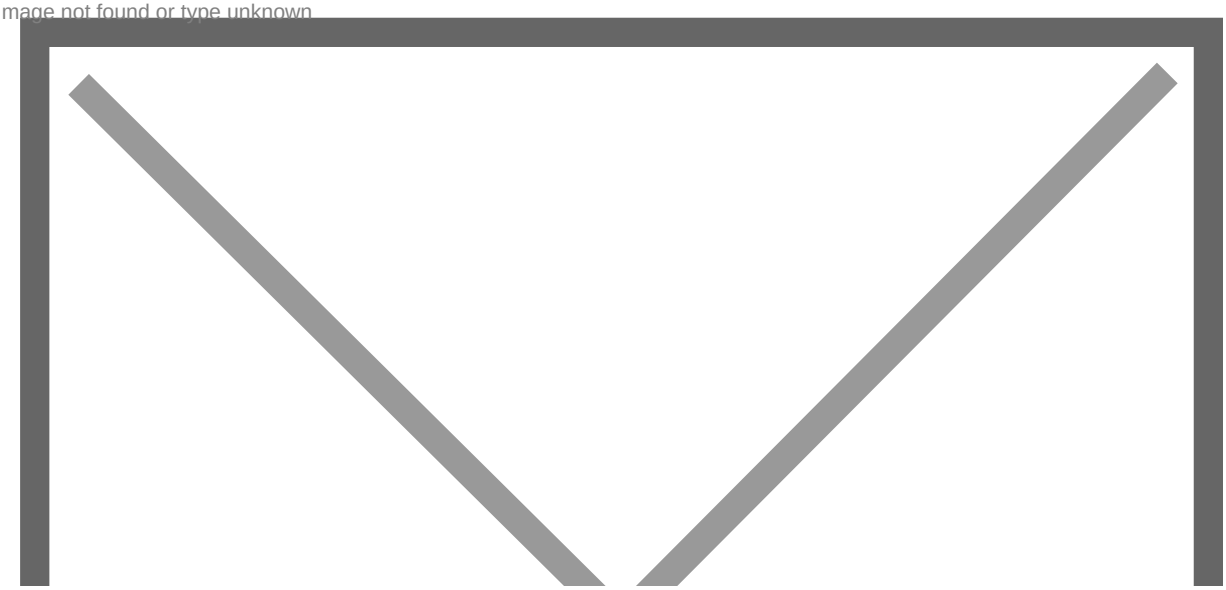
Petite clarification du vocabulaire:

Une vue est une page XML qui touche à l'interface de Odoo.
Un template (ou vue Qweb) est une vue qui touche aux rapports pdf et aux pages du frontend. Ce guide s'applique aux rapports mais marche aussi pour les vue front-end, et dans une certaines mesures, toutes les autres vues.

Cas utilisé

Nous allons prendre un exemple lié à une tâche récente pour foodhub.
La demande était de rajouter une ligne dans les notes de crédits avec la phrase ci-dessous.

Image not found or type unknown



Trouver le rapport à modifier

La première chose à faire dans ce cas-là est d'aller voir le code du rapport en question. Pour faire cela :
On note le nom du modèle dont il est question. On peut le voir ici dans l'url de la vue liste des notes de crédit : https://wholesale.test.coopiteasy.be/web#action=776&model=account.invoice&view_type=list&menu_id=117. On sait donc qu'on a affaire au modèle account.invoice
Ensuite, on applique le mode dev et on va dans Settings>Technical>Reports.
Ceci est une liste des "actions" qui ouvrent les rapports (pas les template des rapports eux-même). On fait une recherche sur le nom du modèle dans la liste. S'affichent alors tous les rapports que l'on peut imprimer à partir de la vue "formulaire" de notre modèle.

Image not found or type unknown

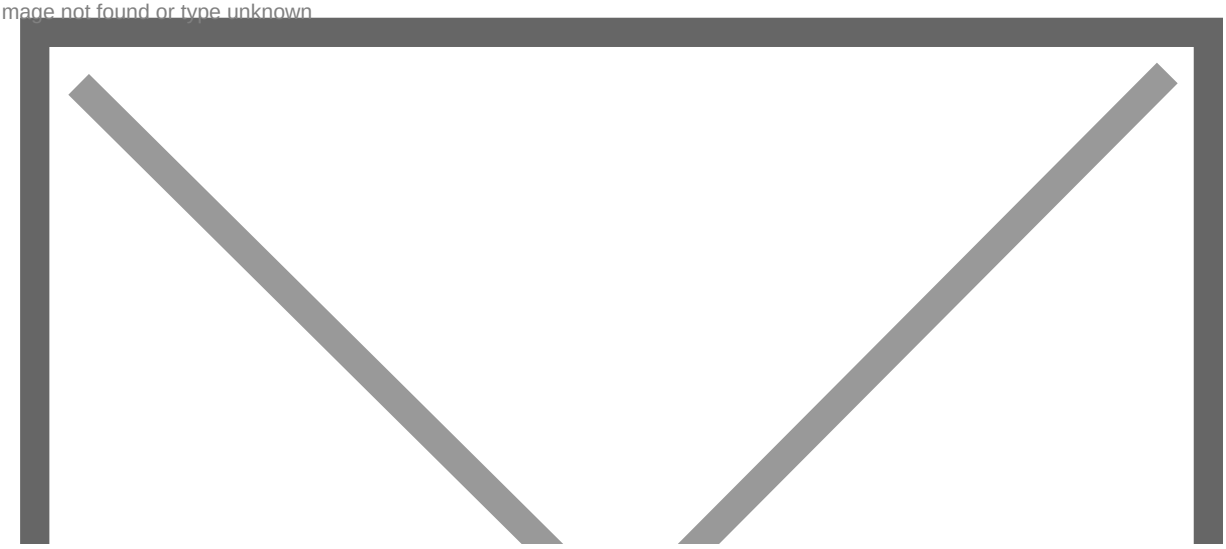
On ouvre le premier (car c’est celui qu’on veut modifier). On appuie sur le smart buttons Qweb View pour accéder au template du rapport lui-même.

Image not found or type unknown

On ouvre le premier élément de la vue “liste”. A partir de là, il faut jouer au chat et à la souris pour trouver le “bon template”. **Les vues et les templates dans Odoo sont organisées en arborescence.** Il y a deux mécanisme dans cet arborescence:

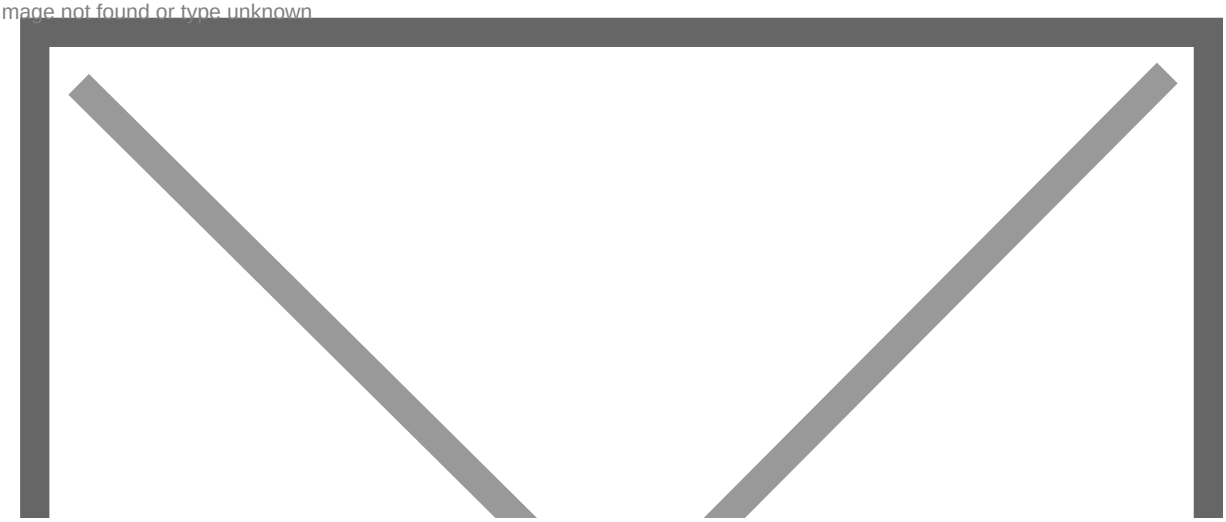
- **Héritage de vue** (une vue peut-être parent ou enfant d’autres vues). On peut naviguer dans cette arborescence via le champ ‘Inherited View’ (vue parente) et l’onglets ‘Inherited Views’ (vue infantes). Voir les flèches bleues/
- Un template peut aussi en **appeler une autre avec l’attribut `t-call`** (voir flèche rouge). L’appel se fait via le XMLID (nomdumodule.nomdutemplate). Cela imbrique un autre template dans le template courant. (Note: cela ne marche qu'avec les template, pas les autres types de vues).

Image not found or type unknown



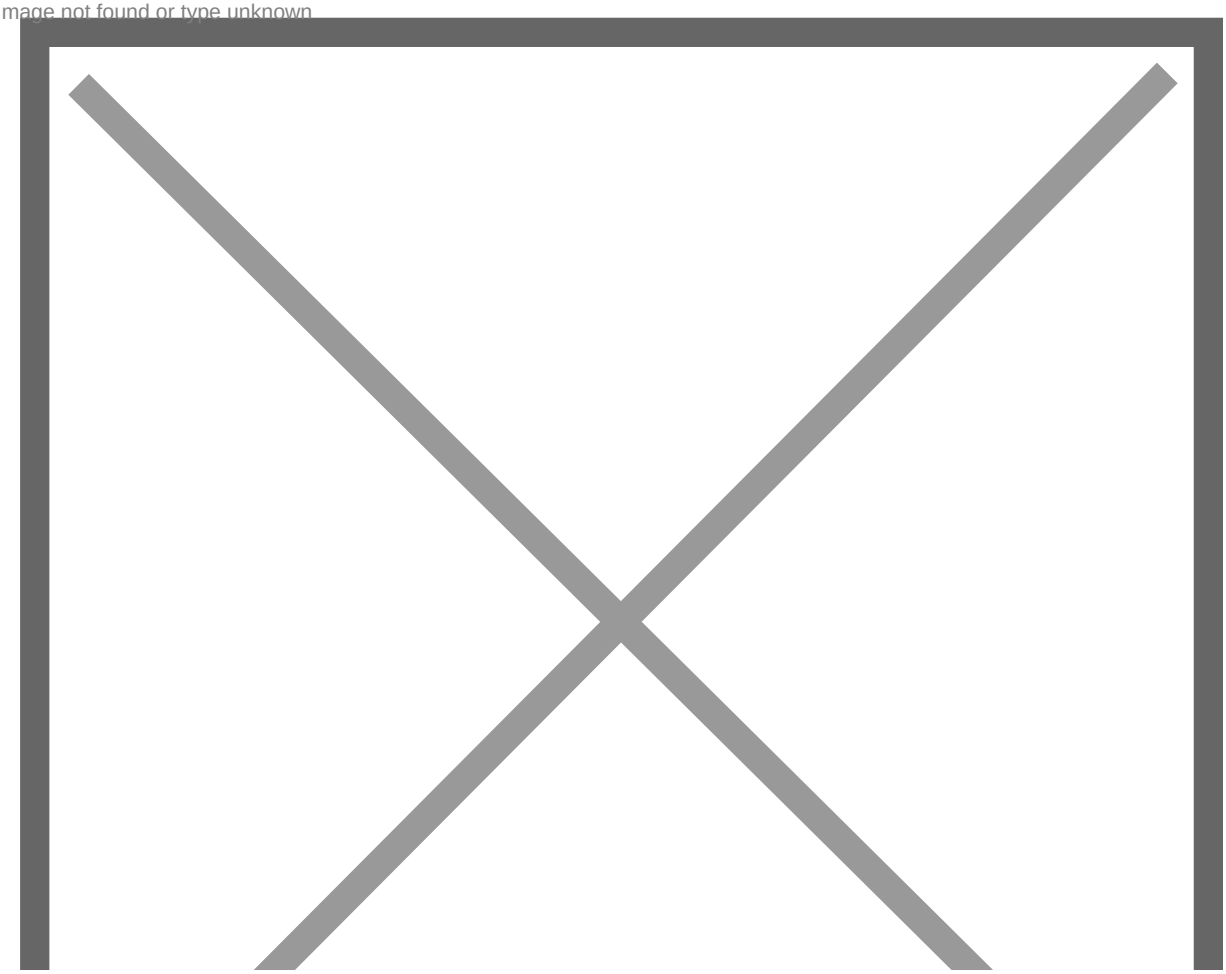
On voit que le template ci-dessus est trop court, ce n’est pas celui là qu’on veut. Dans ce cas-ci, il n’y a pas de vue infante ni parente. On voit par contre un appel ‘t-call’ (voir flèche rouge).
On cherche donc quel template est appelé via ‘t-call’. Pour cela on va dans les vues (Settings>Technical>Views) et on cherche le nom du template appelé (on prend le nom du template, donc ce qui est après le point : report_invoice_document_with_payments).
On trouve un résultat. On l’ouvre:

Image not found or type unknown



On observe ici que le template est toujours très court, ce ne doit pas être le bon. On constate qu'il y a une vue parente. On clique dessus et tada ! Un bon template bien long ! En parcourant le code on voit des champs qui semblent correspondre à ce qu'on voit dans le pdf de la facture. Parfait, on a trouvé le bon rapport, c'est report_invoice_document !

Image not found or type unknown



Modifier le rapport

Pour modifier le rapport, on ne modifie pas le code du template lui-même, mais on crée un template qui hérite de lui et le modifie. Cela permet que le changement ne disparaisse pas lors d'une mise à jour du module.

On crée donc une nouvelle vue en appuyant sur Create.

On remplit le formulaire. Seuls les champs suivants sont importants:

- View Name : on choisit par convention le même nom que la vue parente + un suffixe qui vient préciser ce qu'on modifie. Par exemple, report_invoice_document_out_refund.
- Inherited View (le champs, pas l'onglet) : c'est là qu'on précise que l'on vient hériter de report_invoice_document
- View inheritance mode : il faut choisir "Extension view"

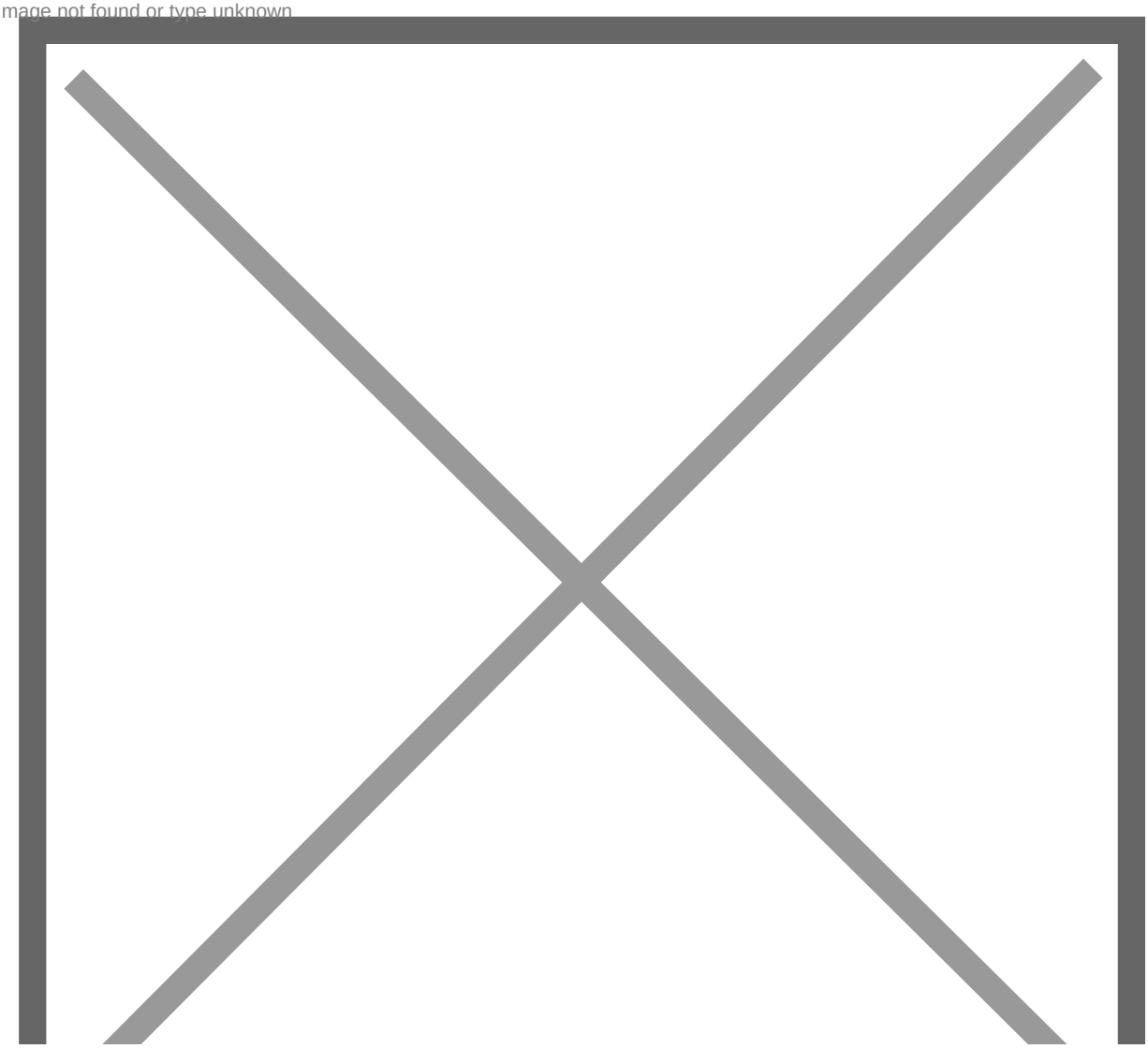
Ensuite, il faut écrire le code xml qui va modifier la vue parente. Voici le code qui a été écrit dans notre exemple:

```
<?xml version="1.0"?>
        <data inherit_id="account.report_invoice_document">
                <xpath expr="//div[@id='total']/.." position="after">
                        <p t-if="o.type == 'out_refund'">Please deduct this credit note from a future invoice or ask a refund.</p>
                </xpath>
        </data>
```

Explication du code :

- `<?xml version="1.0"?>` : il faut mettre ça dans tous les code XML
- `<data inherit_id="account.report_invoice_document"></data>` Ici on reprécise le XMLID qui est hérité : on hérite de report_invoice_document qui est dans le module “account”. Je ne suis pas sûr à 100% que ce soit nécessaire mais dans le doute on le met.
- `<xpath expr="//div[@id='total']/.." position="after"></xpath>` C’est ici qu’on dit où on veut faire la modification. Pour les rapports, le seul moyen est le “xpath”. (note : pour les autres vues il existe un moyen plus simple, cf le livre que j’ai cité en introduction). Il faut aller voir le code de la vue parente, et situer l’endroit dans le code où l’on veut intervenir. Dans notre cas, on veut ajouter une ligne en dessous du tableau qui finit par “Totaal”. Dans le code on peut deviner que c’est dans cette zone:

Image not found or type unknown



On a un élément ‘Total’, qui est imbriqué dans plusieurs balises “td”, “tr”, “table” (qui sont des balises liées au tableau), et (le slash “/” veut dire que c’est des balises de fermeture).

On décide d’intervenir au-dessus du “<p>” qui a un “name=comment”, mais après la dernière balise “div”, voir la flèche bleue.

On veut donc placer l’élément après toute la grosse balise “div” qui a “class=clearfix”. On remplit donc l’attribut “expr” de xpath avec : "//div[@id='total']/..". Ce qui se traduit en : la balise parente (“/..") de la balise div qui a pour id ‘total’ (//div[@id='total']").

Dans l’attribut “position”, on précise qu’on veut ajouter quelque chose après l’élément qu’on a sélectionné (position="after").

Note : il peut y avoir plusieurs manières de sélectionner un élément avec xpath. Les principaux critères de choix sont:

- Faire au plus précis (on préfère cibler un “id” ou un “name”, qui est spécifique à un élément plutôt qu’une “class”).

- Faire au plus simple

Par exemple on aurait aussi pu avoir <xpath expr="//p[@name='comment']" position="before"></xpath>. Ça aurait été plus simple, même si on préfère “id” plutôt que “name” car il y a un contrôle sur l’unicité des id, pas sur le name.

- Please deduct this credit note from a future invoice or ask a refund. . Enfin, on code l’élément qu’on ajoute. Ici c’est un texte (d'où la balise pour “paragraphe”). On affiche le texte seulement si le champs type de l’objet (une facture car la vue est liée au modèle account.invoice) est ‘out_refund’, c’est) dire une note de crédit (t-if="o.type == 'out_refund'")

Documenter le changement

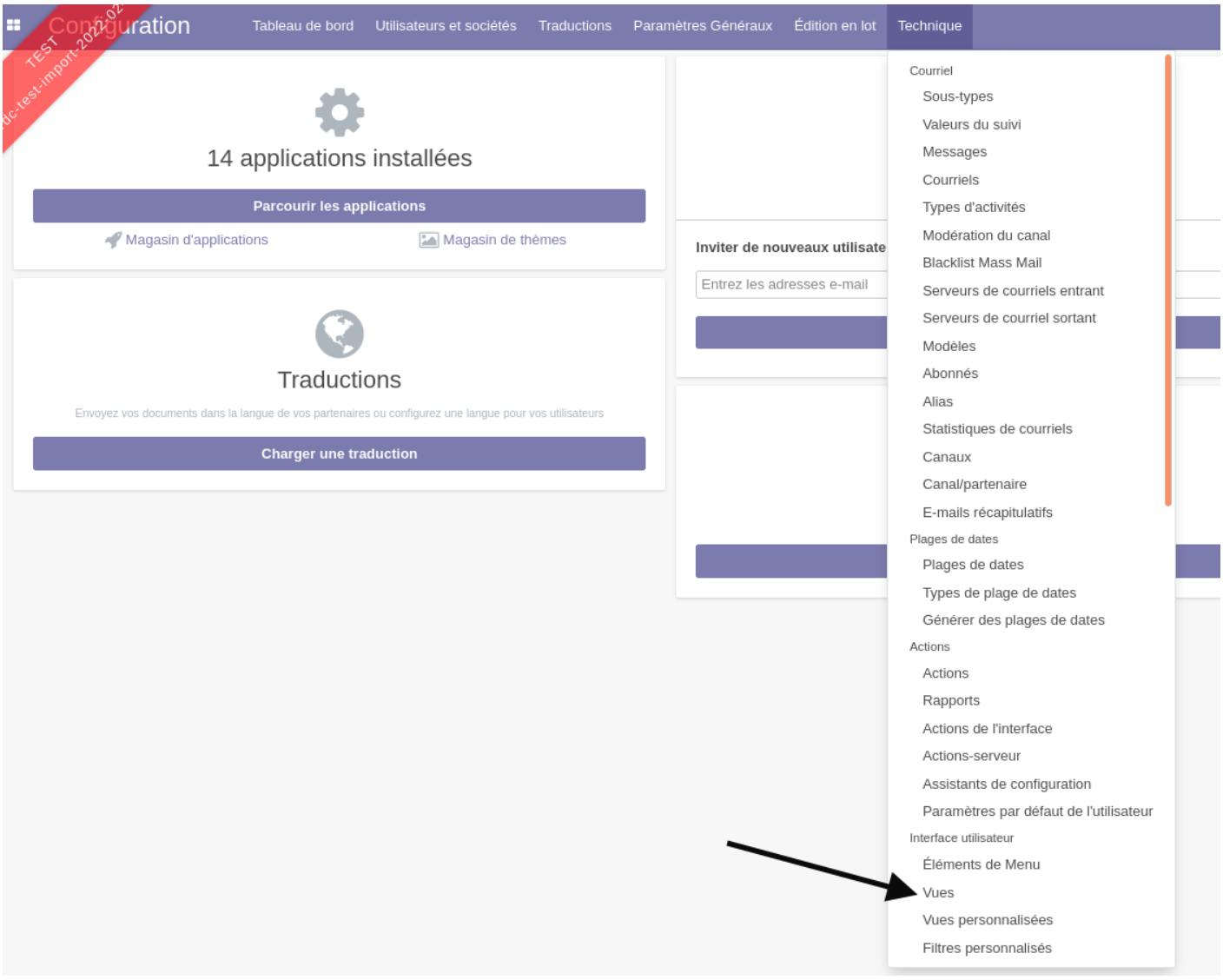
On va ensuite documenter le changement dans le module custom existant (s’il existe, sinon que faire ?). Par exemple, on va dans le repo cie_custom, puis dans le module foodhub_custom. On ouvre le fichier readme/DESCRIPTION.rst, on l’édite (en cliquant sur le crayon en haut à droite). On édite le fichier et on écrit un message de commit (par exemple : “[ADD] report modification doc”). On choisit l’option “Create a new branch” pour ouvrir une pull request. On renseigne ensuite la pull request en taggant des développeurs pour qu’ils valident le changement.

Empêcher qu'une vue soit écrasée lors d'une mise à jour

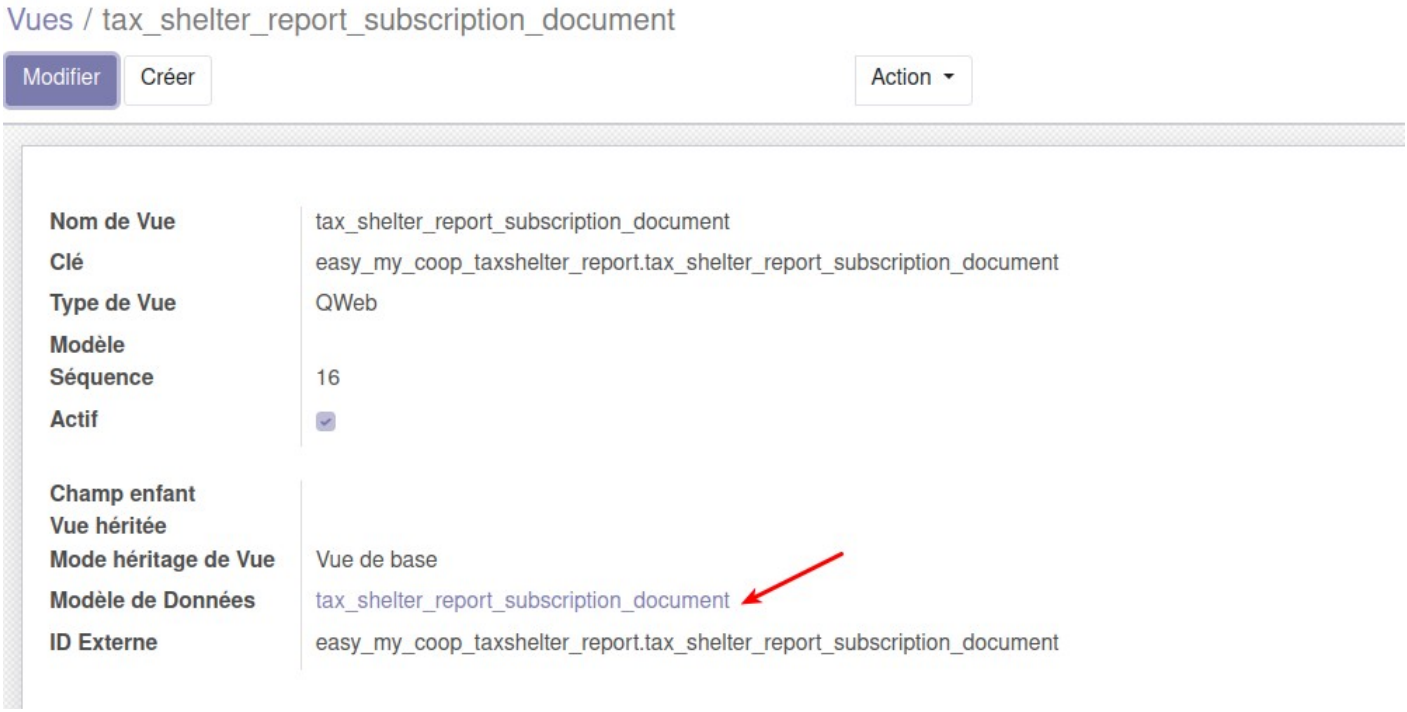
Certains objets dans Odoo sont modifiables mais pas pérennes, c'est-à-dire qu'une mise à jour peut supprimer les modifications faites.

Pour empêhcer les mises à jour au niveau des vues (ce qui concerne principalement les rapports PDF), il faut être en mode débug puis

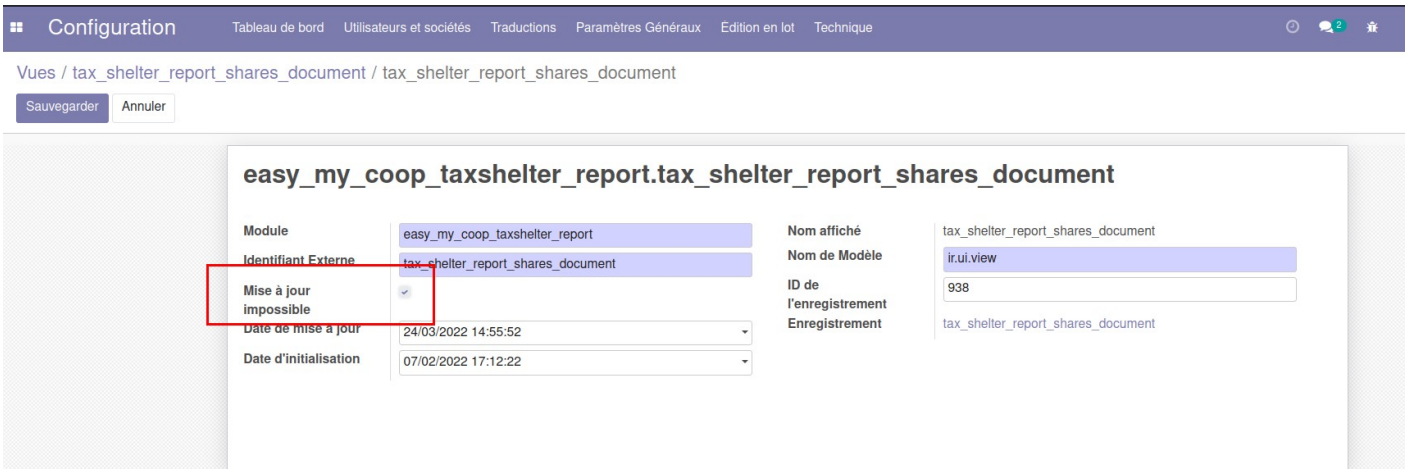
- aller dans la configuration générale < technique <vue



- aller sur la View du rapport,
- cliquer sur Modèle de données,



- dans la vue qui s'ouvre, cliquer sur la case "Mise à jour impossible"

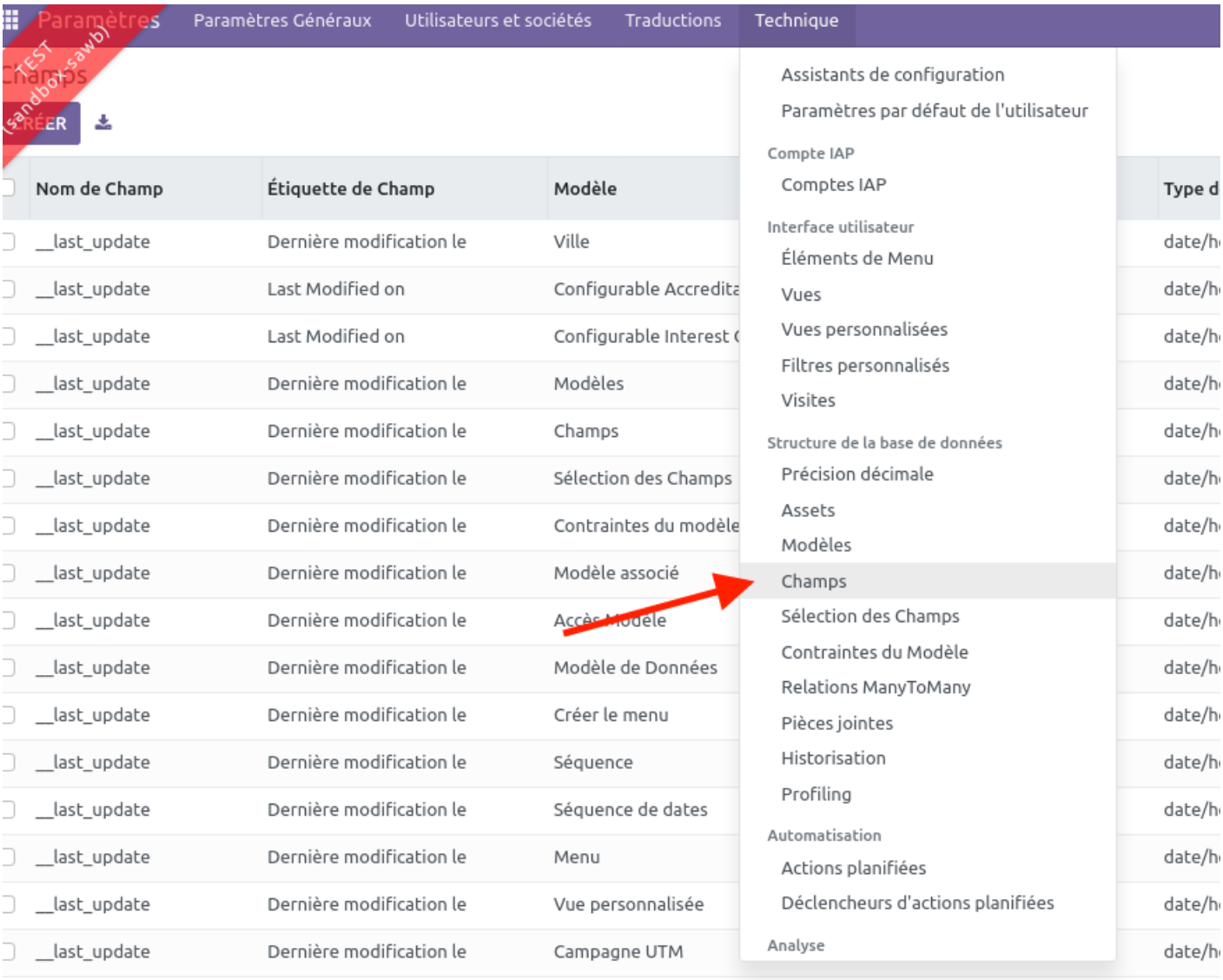


Ajouter un champs - Odoo v16

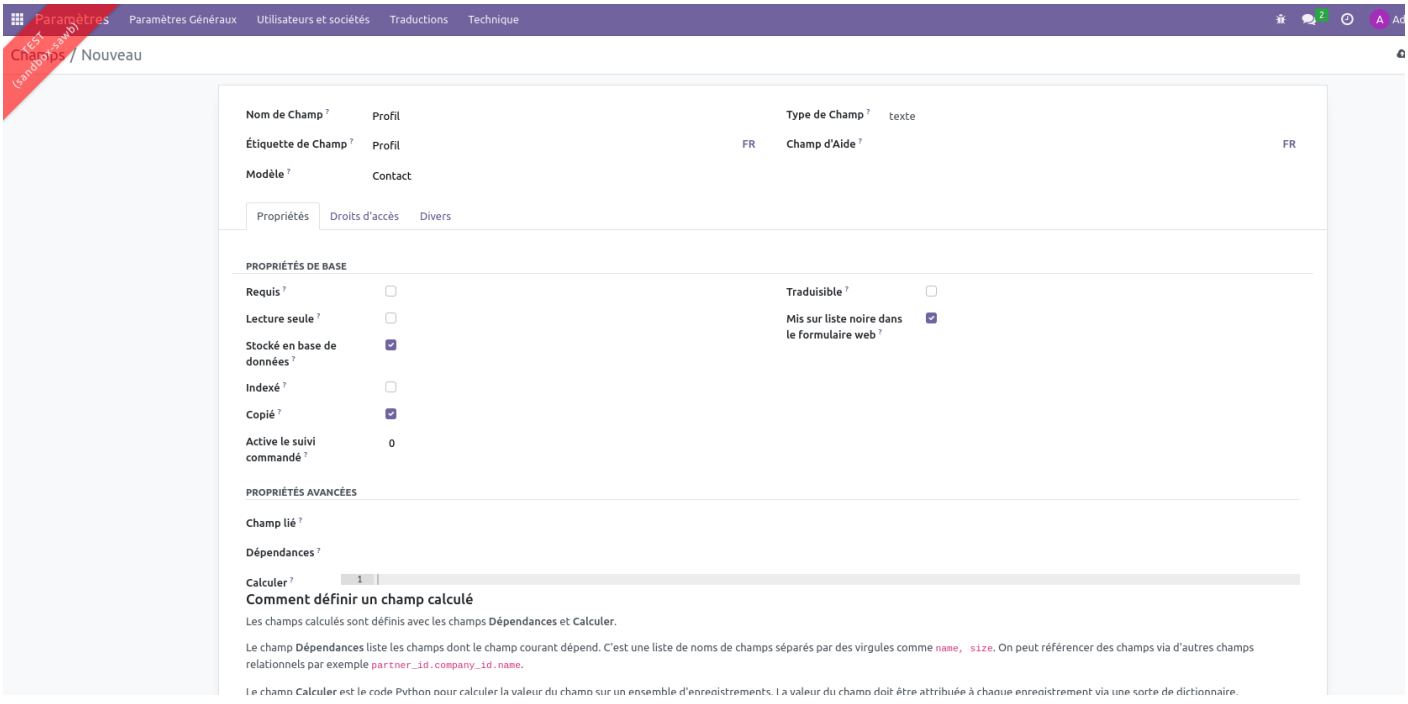
Odoo permet d’ajouter des nouveaux champs par l’interface.

> Attention que
> - les changements peuvent être invisibles pour les programmeurs, car les éléments personnalisés ne figurent nulle part dans le code. Le débogage devient plus difficile en conséquence
> - les migrations deviennent beaucoup plus difficiles car il faut retrouver toutes les modifications personnalisées et les prendre en compte
> - former les gens à l'utiliser correctement est plutôt difficile.
> - les modifications personnalisées peuvent être remplacées par des mises à jour de modules, ce qui annule le travail.

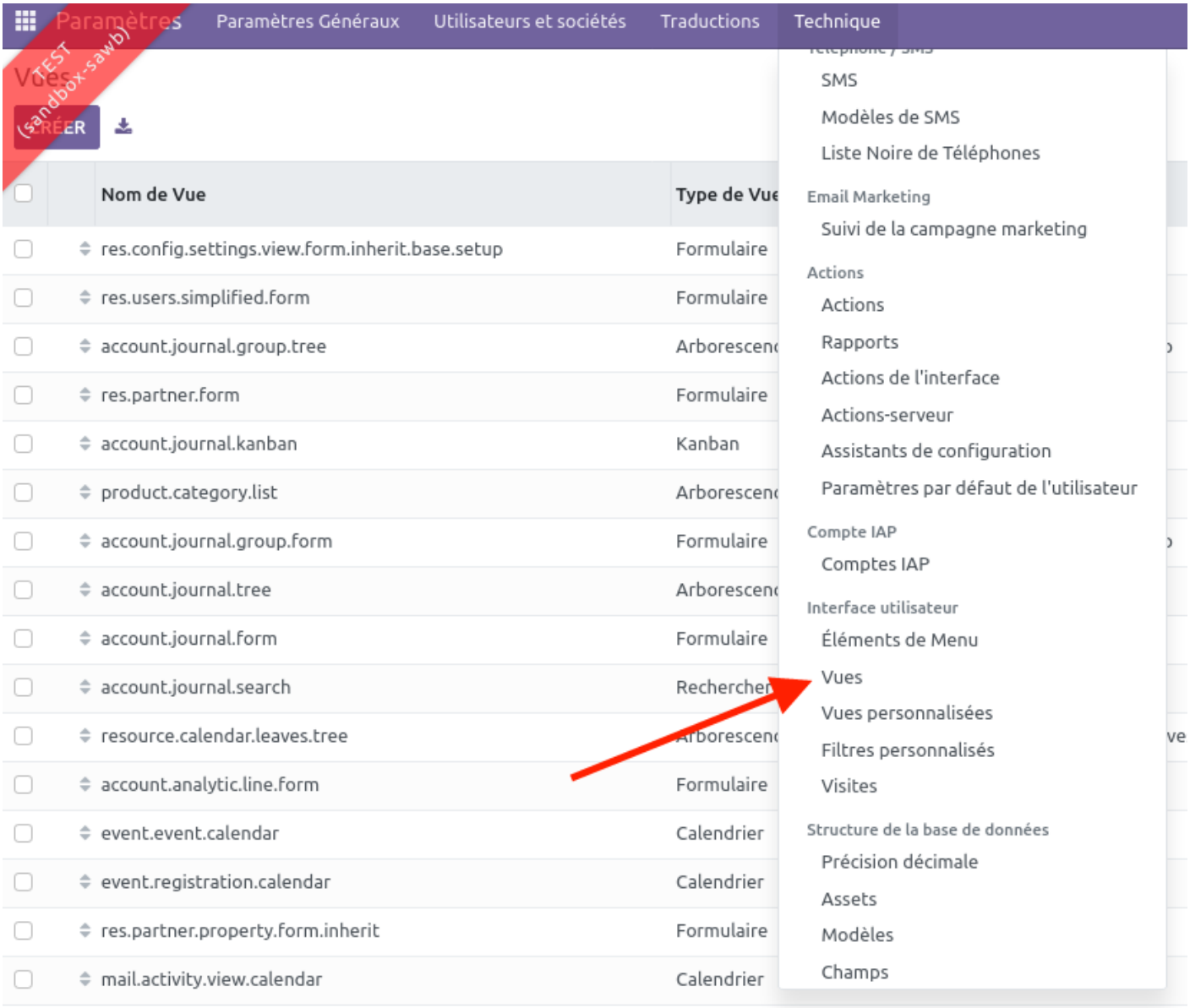
Activer le mode développeur et aller dans le menu de configuration<technique<Structure de la base de données>champs



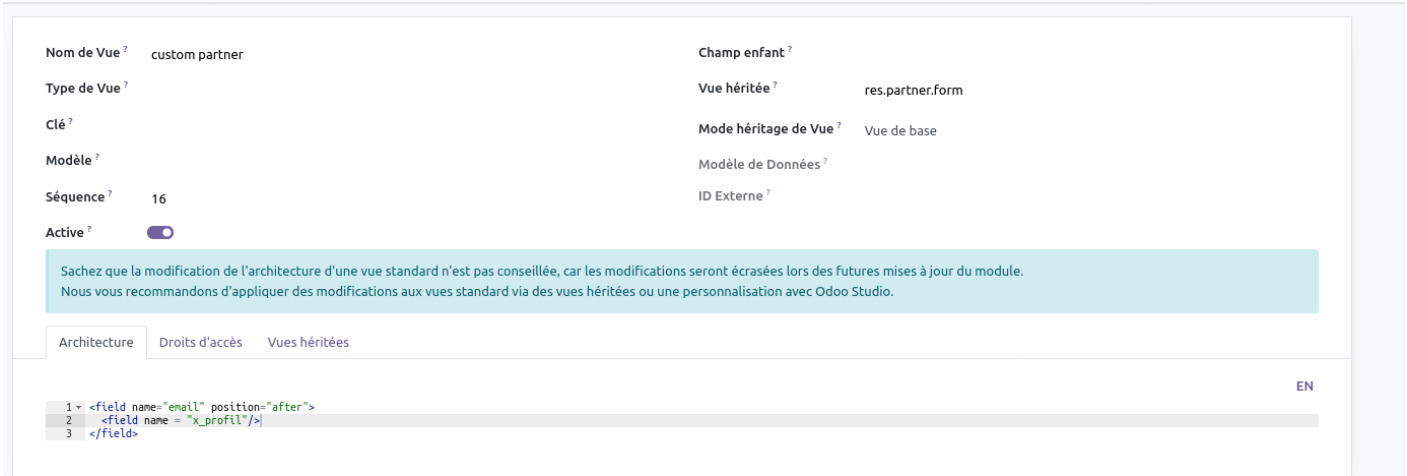
Créer ensuite un nouveau champs et entrer les paramètres adéquats. Un champ ajouté doit avoir un nom précédé d'un x_



Aller ensuite dans le menu configuration<technique<Interface utilisateur<vues



Créer une nouvelle vue qui héritera d'une vue existante. Cela permet de s'assurer que les modifications soient pérennes.



Dans l'exemple ci-dessus, le champs "x_profil", champs texte libre a été rajouté à la vue contact, après le champs "email".

```
<field name="email" position = "after">
<field name = "x_profil"/>
</field>
```