OpenFire
ZA La Brosse – 5 rue de la Garenne
35520 La Chapelle des Fougeretz
France

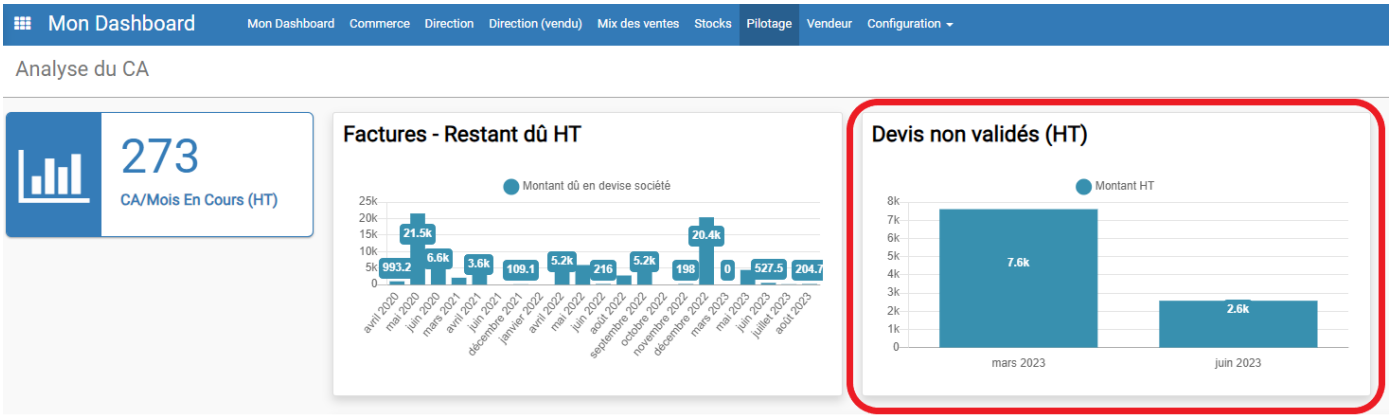
Notion de modèle, domaine et opérateur

Précédent
Sommaire

Il est possible d'optimiser et personnaliser les Widgets en utilisant différents champs techniques à disposition.

- d'une part, en précisant le modèle le plus pertinent a utiliser,
- puis en utilisant des filtres et des conditions pour vos données à afficher (via la notion de domaine).

Pour l'ensemble de cet article, nous allons partir d'un cas pratique: *je souhaite obtenir un graphique représentant le montant des commandes non validées, regroupées par Mois :*



Définition du Modèle

Pour réaliser ce tableau, je dois commencer par définir le modèle qui m'intéresse.

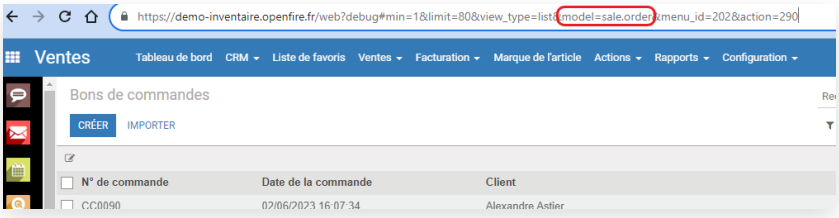
Dans OpenFire, un modèle définit la structure de données d'un type spécifique d'objet, comme les commandes de vente, la comptabilité ou la gestion des stocks. Il s'agit d'une classe Python qui spécifie les champs, les méthodes et les caractéristiques associés à cet objet.

Par exemple, dans la gestion des ventes, le modèle "Bon de commande" (sale_order) représente les commandes passées par les clients. Il contient des détails tels que les produits, les quantités, les prix, les clients, etc.

Pour obtenir un graphique représentant mes commandes non validées, je vais donc devoir utiliser le modèle bon de commande (ou sale_order).

La méthode la plus simple pour retrouver le nom d'un modèle, est de l'identifier directement depuis le menu qui nous intéresse. En effet, le nom du modèle est récupérable depuis l'URL visible en haut de votre navigateur, dans la partie " model= ".

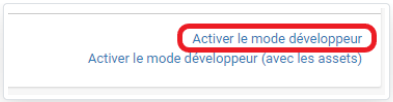
Par exemple, le domaine du menu **Ventes > Bon de commande** est sale.order :



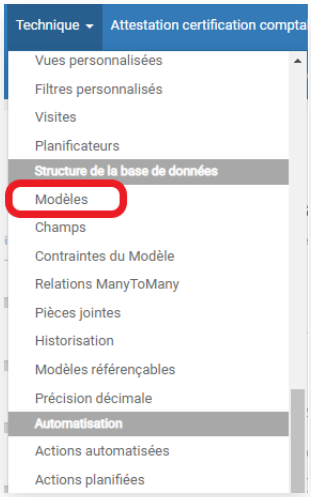
Attention: Dans l'url, le modèle est séparé par des Points, alors que dans les Widgets, le séparateur a utiliser est l'underscore (_). Ainsi le domaine sale.order deviendra sale_order

Ces noms de modèles sont également disponibles dans le menu configuration mais cela nécessite l'activation du mode Développeur.

Pour cela, rendez-vous dans l'onglet **Configuration** puis cliquez sur l'option Activer le mode développeur à droite :

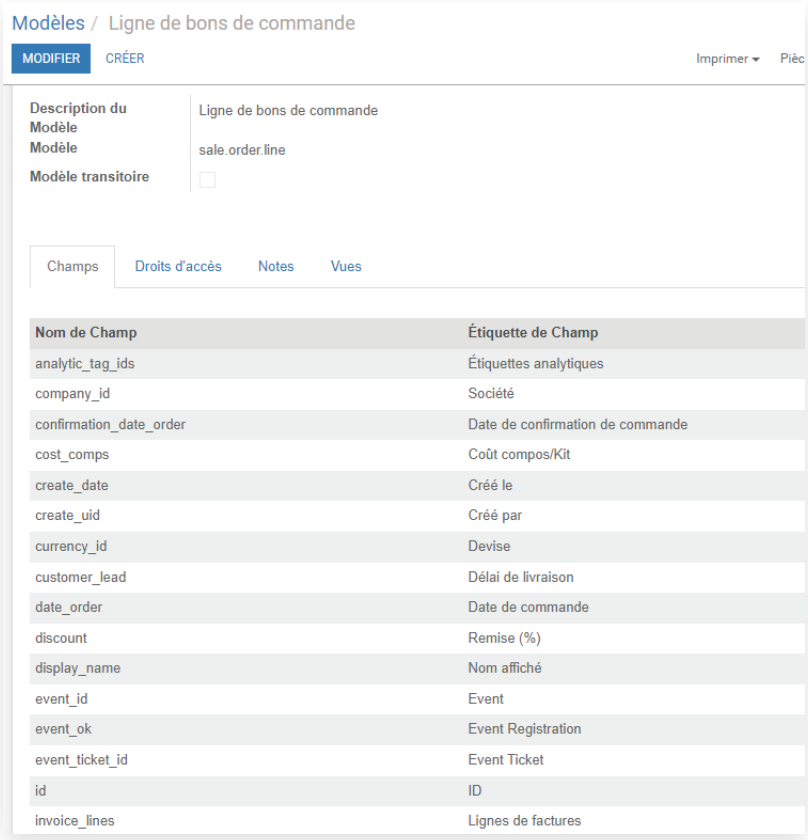


De nouveaux menus apparaitront alors. Toujours dans le menu Configuration, cliquez alors sur **Technique > Structure de la base de données > Modèles**:



Depuis ce menu, je peux retrouver l'ensemble des modèles d'OpenFire. Par exemple, si je cherche le domaine des lignes de commande je trouve la valeur *sale.order.line*.

En outre, si je clique sur le modèle, j'obtiens la liste des différents champs qui sont disponibles pour ce modèle:



Nom de Champ	Étiquette de Champ
analytic_tag_ids	Étiquettes analytiques
company_id	Société
confirmation_date_order	Date de confirmation de commande
cost_comps	Coût compos/Kit
create_date	Créé le
create_uid	Créé par
currency_id	Devise
customer_lead	Délai de livraison
date_order	Date de commande
discount	Remise (%)
display_name	Nom affiché
event_id	Event
event_ok	Event Registration
event_ticket_id	Event Ticket
id	ID
invoice_lines	Lignes de factures

Notion de domaine

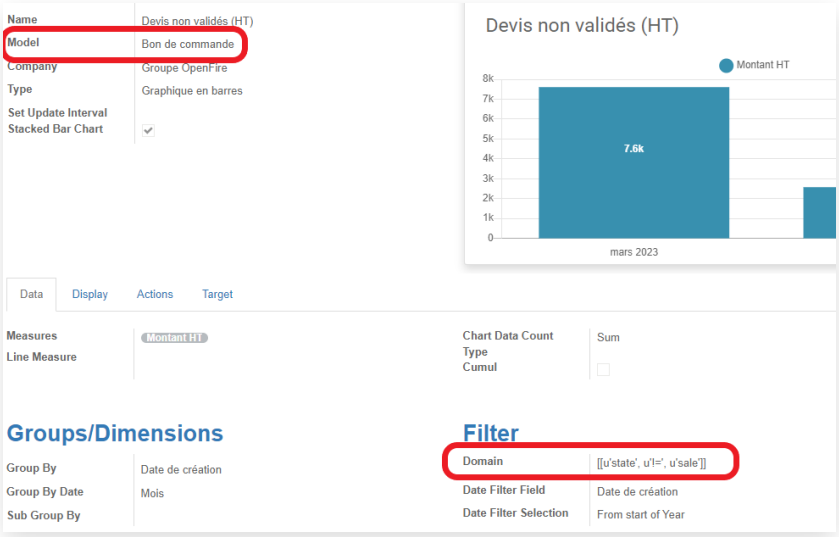
Dans OpenFire, un domaine représente un ensemble de critères de filtrage appliqués aux données. Il permet de restreindre les enregistrements visibles dans une application.

Les domaines sont des règles qui définissent quels enregistrements un utilisateur peut voir en fonction de critères spécifiques. Cela optimise l'interface en filtrant les données selon les besoins individuels, améliorant ainsi l'efficacité et la pertinence.

Le domaine est une liste python, définie dans des crochets [ ], et constituée de filtres en parenthèses et de connecteurs logiques ('&', '|'). Les filtres entre parenthèses ont pour structure ('nom_du_champ', 'opérateur', valeur)

Pour obtenir mon graphique représentant le montant des commandes non validées, je vais donc devoir :

- utiliser le modèle bon de commande (ou sale_order).
- puis filtrer mes résultats dans le domaine de recherche. Pour commencer, je vais utiliser le domaine [[u'state', u'!=', u'sale']]



Explications:

Nous avons vu que dans le contexte spécifique d'OpenFire, le modèle "Bon de commande" (sale_order) est lié à la gestion des commandes de vente.

L'expression [[u'state', u'!=', u'sale']] est utilisée comme filtre dans le domaine de recherche pour sélectionner un sous-ensemble de commandes de vente en fonction de leur état (state).

Décomposons l'expression [[u'state', u'!=', u'sale']] :

- u'state' fait référence au champ "state" (état) du modèle "Bon de commande" (sale_order). Cela représente l'état actuel de la commande de vente ("Estimation", "Devis", "Bon de Commande", etc.).

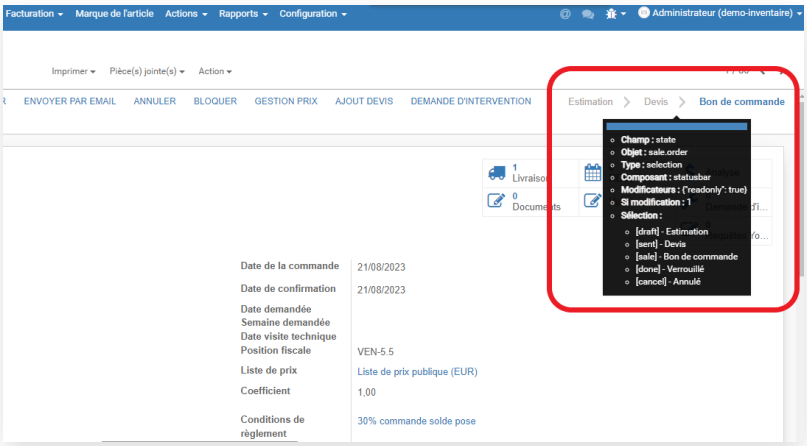
Le nom du champ est le nom technique de l'objet dans lequel on se trouve, obtenu grâce au mode développeur.

Les différents états possible pour une commande sont les suivants:

- draft : Estimation
- sent: Devis
- sale: Bon de commande

- done: Verouillé
- cancel : Annulé

A Savoir: Les différents états sont visibles via le mode Développeur, au simple survol d'un champ avec la souris:



- u'!=' est un opérateur de comparaison qui signifie "différent de".
- u'sale' est une valeur qui, dans ce contexte, représente l'état "Bon de commande".

Donc, l'expression [[u'state', u'!=', u'sale']] est un filtre pour sélectionner les commandes de vente dont l'état n'est pas Bon de Commande.

Ce résultat est fonctionnel. Néanmoins, ce domaine implique que toutes les commandes de vente qui ne sont pas encore en état de "Bon de Commande" seront incluses dans le résultat de la recherche.

En d'autres termes, les commandes dans des états tels que "Estimation", "Annulé" et 'Verrouillé" apparaitront également.

Pour la suite de cet exercice, je souhaite aller plus loin et *obtenir un graphique représentant uniquement le montant de mes devis non signés, regroupés par mois.*

Je ne souhaite donc pas voir les états "Estimation", "Annulé" et "Verrouillé".

Pour cela, je vais faire appel aux opérateurs de comparaison.

Opérateurs Python

OpenFire est basé sur l'ERP Odoo qui utilise le langage de programmation Python pour la plupart de ces fonctionnalités. Aussi, les opérateurs de comparaison que nous allons utiliser doivent utiliser ce langage.

Les opérateurs sont des symboles spéciaux en Python qui effectuent des calculs arithmétiques ou logiques.

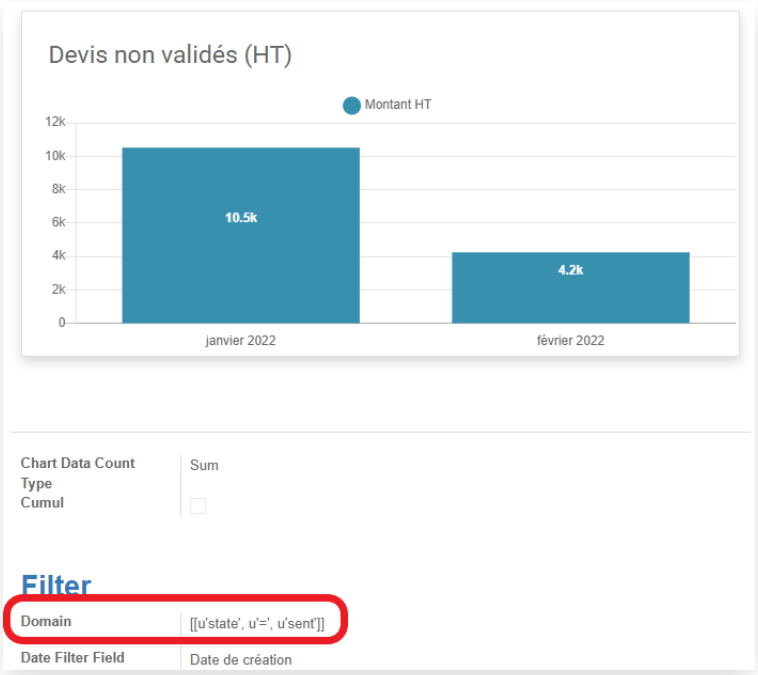
Voici quelques exemples d'opérateurs et leurs significations:

Opérateurs

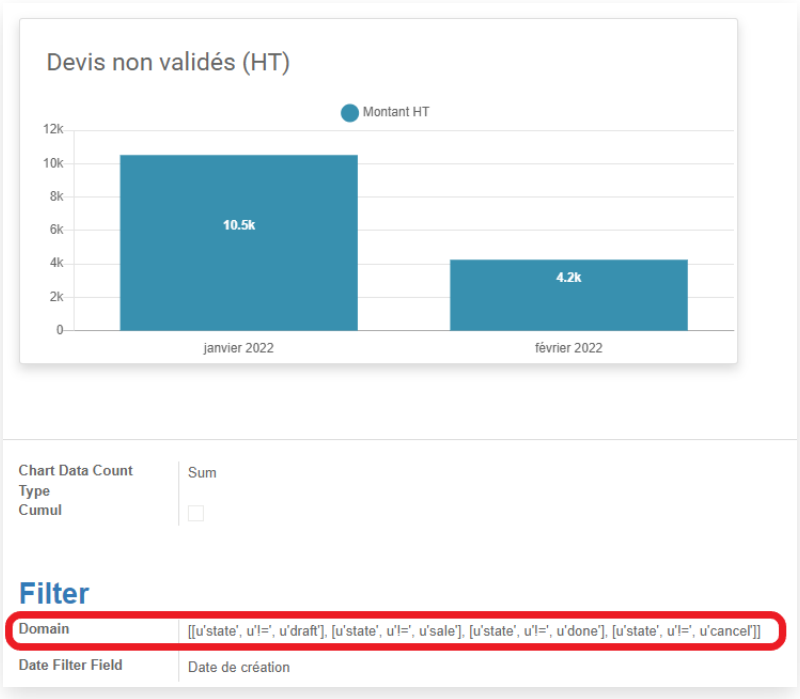
Pour notre exemple, nous allons donc modifier notre domaine initial à l'aide de ces opérateurs.

Actuellement, nous avons [[u'state', u'!=', u'sale']]. Deux approches alors sont possibles :

- Soit je filtre les commandes dont le statut est "Devis", mon domaine deviendrait alors [[u'state', u'=', u'sent']]



- Soit je filtre les commandes dont le statut n'est pas "Bon de commande", "Estimation", "Annulé" et "Verrouillé".
  Mon Domaine deviendrait donc [[u'state', u'!=', u'draft'], [u'state', u'!=', u'sale'], [u'state', u'!=', u'done'], [u'state', u'!=', u'cancel']]



Bien que les domaines soient différents, le tableau obtenu est bien identique dans les deux cas.

💡 A Savoir: Différentes fonctions permettent de filtrer par date, par exemple:

Fonction : context_today()

- Exemple 1 : date supérieure ou égale à la date du jour : ['date', u'>=', context_today().strftime('%Y-%m-%d')]
- Exemple 2 : date supérieure ou égale au premier jour du mois en cours: ['date', u'>=', context_today().strftime('%Y-%m-01')]

Fonction : relativedelta

- Exemple : date inférieure ou égale au dernier jour du mois en cours (= date strictement inférieur au premier jour du mois suivant) : ['date', u'<', (context_today() + relativedelta(day=1,months=1)).strftime('%Y-%m-%d')]

Note :

- months au pluriel = j'ajoute X mois relativement à la date saisie (dans l'exemple = context_today)
- days au pluriel = j'ajoute X jour relativement à la date saisie (dans l'exemple = context_today)
- month au singulier = récupère un mois en particulier
- day au singulier = récupère un jour en particulier
- relativedelta(day=X,months=Y) → Jour X de Y mois après le mois de référence
- relativedelta(month=X,years=Y) → mois X de Y année après le mois de référence