om5 Documentation

Version 1.2.0a1

om5

août 07, 2024

Table des matières

om5_no_code a pour objet de créer des applications sans avoir besoin d'écrire une seule ligne de code. En effet, om5_no_code est une sur-couche du framework openMairie et propose de générer (sans compétences en langage informatique) une application qui aurait autrefois nécessité l'intervention d'un développeur spécialisé. Autrement dit, de créer du code, sans coder !

om5_no_code propose au niveau d'abstraction de l'analyse, la génération de code (formulaire, édition) et l'utilisation des ressources de la base de données postgresql (triggers, procédures, vues).

Note : La partie « triggers, procédures, vues » est en cours de développement.

L'interface graphique augmente la productivité des développeurs mais la contrepartie de cet avantage est la généralité et avec om5_no_code, une interface générique ; mais il est possible de revenir sur l'utilisation du framework openMairie. (surcharge des classes métier générées, css, javascript . . .).

Enfin, l'utilisation des techniques déclaratives visuelles au lieu de la programmation permet aux experts métier de diriger ou de participer à la fourniture de la solution.

Ce document a pour but de guider les utilisateurs et les développeurs dans la prise en main du projet om5-no-code.

Il est composer de 2 modules :

- un installateur du framework,
- un outil de développement rapide qui permet de créer les tables et de générer des formulaires et des éditions.

La partie installateur décrit le fonctionnement du module d'installation du framework openMairie.

La partie developpement rapide, décrit le fonctionnement de ce module autour d'un exemple de bibliothèque.

La partie téchnique propose de décrire l'integration du projet om5_rad dans le framework openMairie.

Bonne lecture et n'hésitez pas à nous faire part de vos remarques à l'adresse suivante : contact@openmairie.org !

Les sources et le téléchargement du projet sont sur la forge de l'ADULLACT au lien suivant :

http://adullact.net/projects/om5-no-code/

CHAPITRE 1

Installateur

Ce chapître décrit l'installateur du projet.

1.1 Pré requis d'installation :

Le framework openMairie 4.10.0 nécessite l'installation
- du serveur web APACHE,
- du module php <= 8.0
- de la base de données POSTGRES

Sources : (debian 9 / juin 2020)

https://wiki.arles-linux.org/doku.php?id=install_debian

1.1.1 Installation des composants sur debian 11

installation serveur apache et module php

```
apt install -y apache2
apt install -y php
apt install -y php-pgsql
apt install -y php-mbstring
apt install -y php-xml
# redémarrer apache
systemctl restart apache2
```

installation de la base de données postgresql et de la cartouche géographique postgis

```
apt install -y postgresql
apt install -y postgresql-contrib
apt install -y postgis
```

création d'un utilisateur deb pour postgres (avec droits de créer une base de données, schéma, tables . . .)

Sources :

https://wiki.arles-linux.org/doku.php?id=securite_postgres

```
su postgres
createuser -P deb -U postgres
        Saisir le mot de passe pour le nouveau rôle : deb
        Le saisir de nouveau :  deb
psql -U postgres
postgres=#
        alter role deb with superuser;
        alter role deb with createdb;
        alter role deb with createrole;
```

créer une base de données om5-no-code

```
createdb om5-no-code
```

télécharger le fichier sur la forge de l'adullact dans /var/www/html : décompresser om5-no-code_[version].zip

Voir explication sur openmairie.org

https://openmairie.readthedocs.io/projects/omframework/fr/4.9/installation/index.html#telecharger-l-archive-zip

créer le répertoire var :

```
/var/www/html/om5-no-code $ mkdir var
```

www-data (utilisateur apache) doit avoir les droits d'écriture sur dyn, gen et var

```
/var/www/html/om5-no-code $ chown -R www-data:www-data var
/var/www/html/om5-no-code $ chown -R www-data:www-data dyn
/var/www/html/om5-no-code $ chown -R www-data:www-data gen
```

dans le navigateur, aller sur le répertoire de l'application app/om_setup_config.php ou directement dans l'application /var/www/html/om5-no-code

dans la configuration standard, le database.inc.php se configure par défaut de la manière suivante : # base = om5_rad ; schema : om5 ; user = deb ; password = deb ;

1.1.2 Cas de la debian 12 et de compatibilité 8.0

Le framework 4.10.0 n'est pas compatible avec la version php 8.2 fournie par debian 12.

Il faut mettre php 7.4 ou 8.0 au lieu de la version de base 8.2

Sources :

https://www.linuxtricks.fr/wiki/debian-installer-une-version-plus-recente-de-php

https://sys-admin.fr/installation-php-7-4-sur-debian/

https://www.interserver.net/tips/kb/change-php-version-apache-ubuntu/

Pour installer php 7.4, il faut récupérer les pakages sur https://packages.sury.org/php/

#liste des clés dans usr/share/keyrings

```
apt-key list
```

ajouter la clé deb.sury.org-php.jpg

```
wget https://packages.sury.org/php/apt.gpg -O /usr/share/keyrings/deb.sury.org-php.gpg

# ajout de la ligne "deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg]
# https://packages.sury.org/php/ bookworm main" dans /etc/apt/sources.list.d/php-sury.
↪list

echo "deb [signed-by=/usr/share/keyrings/deb.sury.org-php.gpg] https://packages.sury.
↪org/php/ $(lsb_release -sc) main" > /etc/apt/sources.list.d/php-sury.list
```

installer la version de php 7.4

```
apt update

# install php7.4 -> création du repertoire etc/php/7.4

apt install php7.4
apt install php7.4-pgsql
apt install php7.4-mbstring
apt install php7.4-xml
```

changer le module php dans apache

```
# desactiver la version 8.2 de php
sudo a2dismod php8.2

# activer la version 7.4 de php
sudo a2enmod php7.4
```

1.2 Installateur :

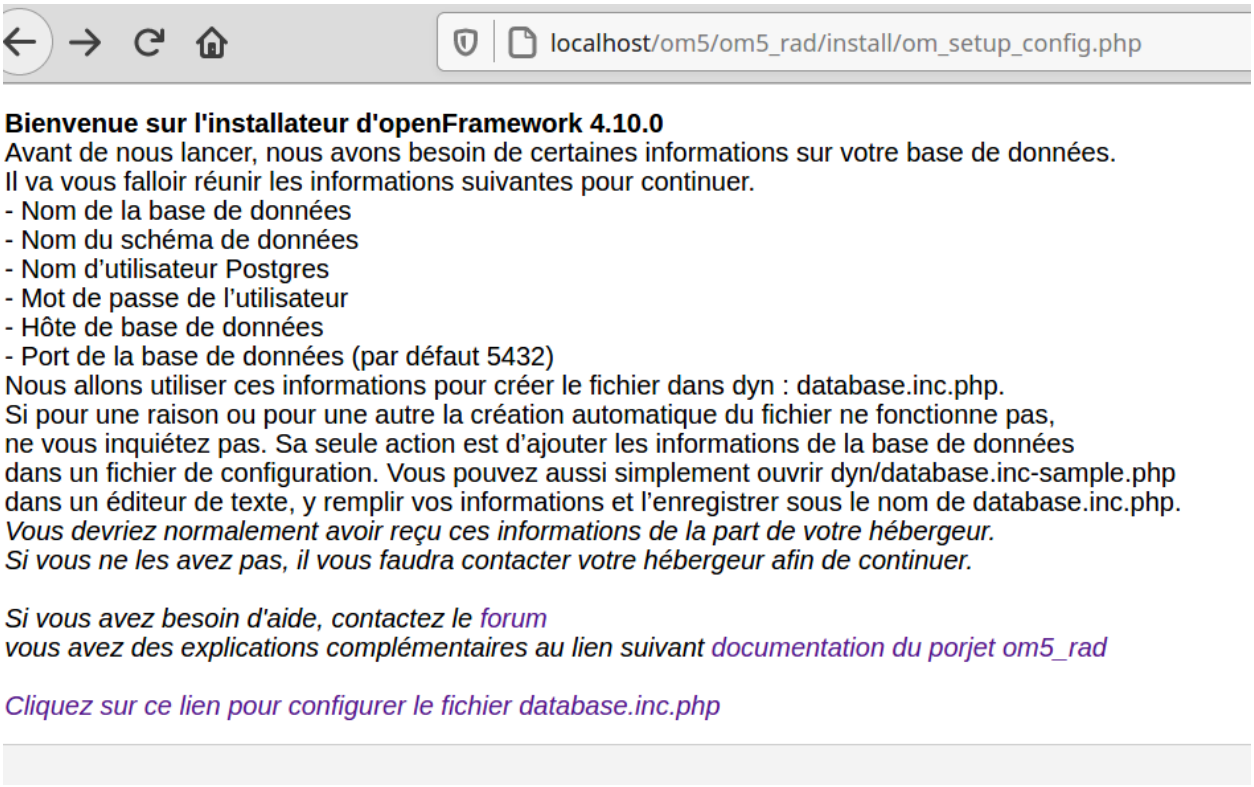
note :

```
Attention, la procédure d'installation détruit le schéma existant d'om5.
```

Lors du lancement dans le répertoire du framework , si le fichier dyn/database.inc.php n'existe pas, il est lancé l'installateur qui est dans install/om_setup_config.php

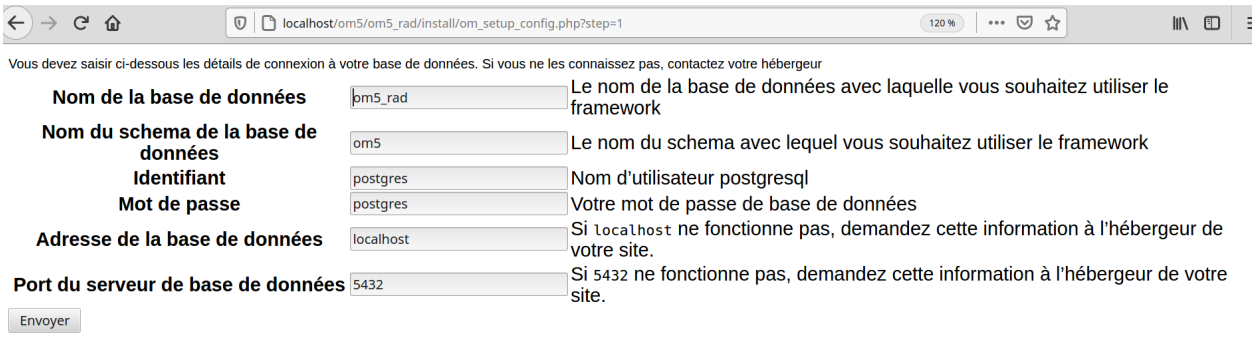
Dans tout les cas, il faut que database.inc.php soit inexistant sinon le framework démarre sur les paramètres de ce fichier.

Démarrer en cliquant sur le lien : « allez c'est parti ! «

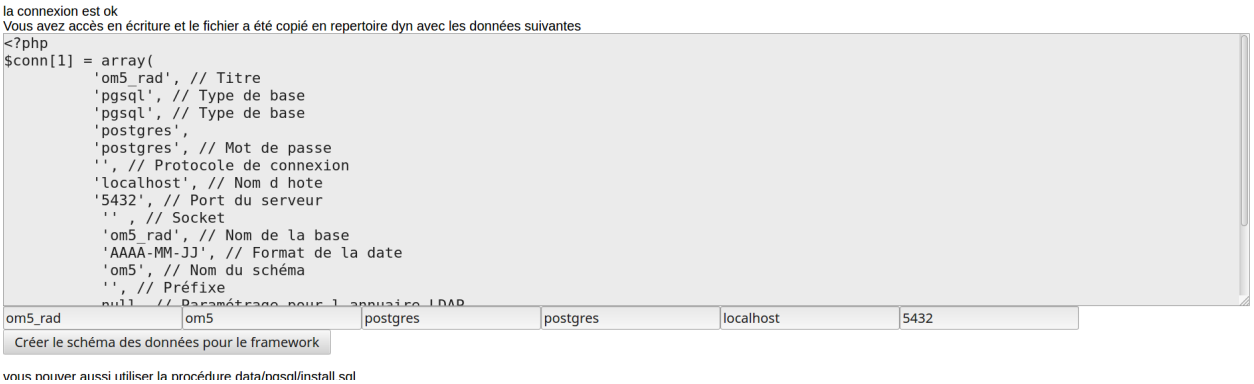


1.2.1 Création du database.inc.php :

La première étape consiste à créer le lien entre l'application et postgres en créant le fichier dyn/database.inc.php

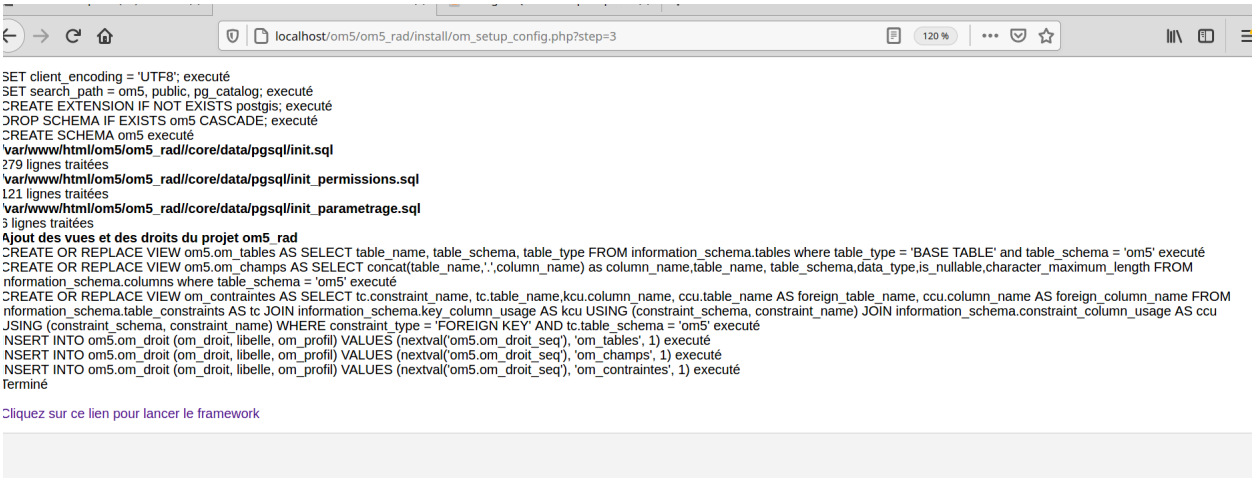


Si vous avez les droits d'écriture sur le répertoire /dyn, le fichier database.inc.php est automatiquement créé. Sinon copier dans un fichier dyn/database.inc.php, le contenu du texte affiché :



1.2.2 Installation du schéma du framework :

Les scripts nécessaires au framawork sont lancés :



En cliquant sur le lien affiché, vous lancez le framework avec le login (admin) et le mot de passe (admin).

CHAPITRE 2

Développement Rapide

Ce chapître décrit le module de développement rapide du projet om5.

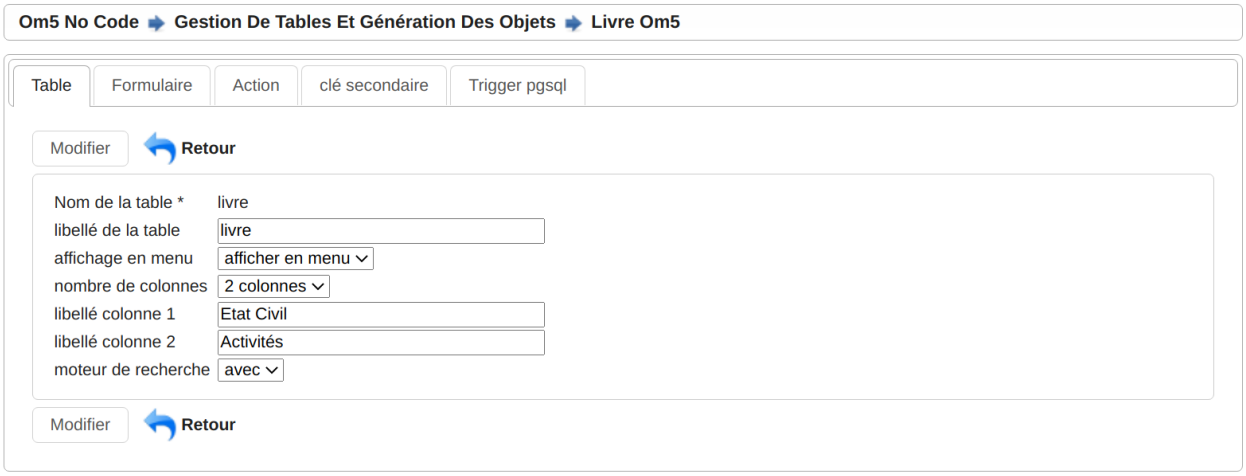
Le premier objectif est de créer les tables avec les champs et les contraintes de clé secondaire, et de générer les listes et les formulaires.

Le deuxième objectif est de créer des traitements avec les triggers ; les procédures et les vues.

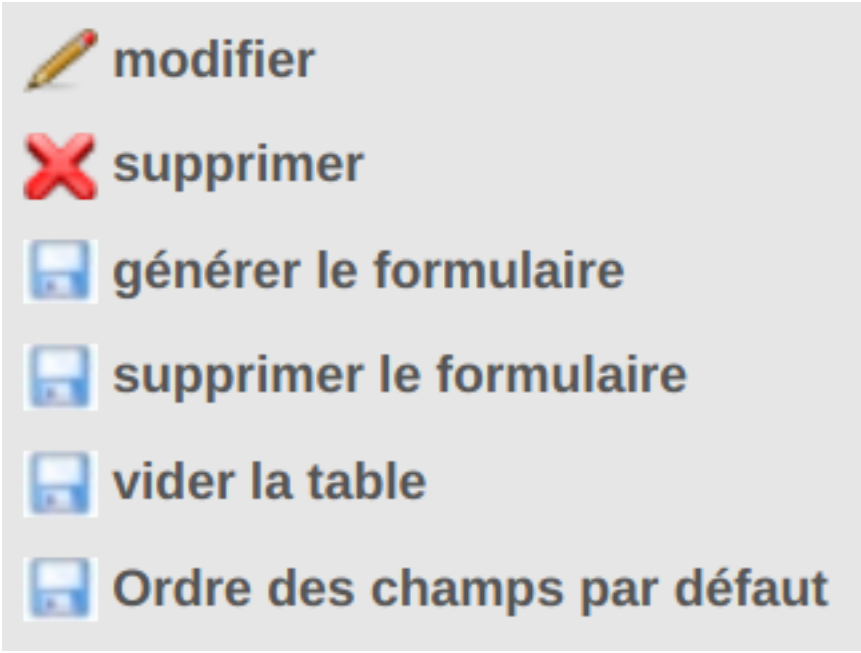
2.1 Saisir table et génération :

Il est possible de lister les tables dans le menu om5-no-code -> option Gestion de Tables

Il est possible de saisir, modifier et supprimer les tables dans le formulaire des tables :



Les actions suivantes sont possibles :



2.1.1 Action ajouter modifier et supprimer :

Lors de l'action ajouter :

- il est vérifié que la cohérence « postgresql » du nom de table ne soit pas vide et qu il ne soit pas existant
- le nom d'origine (non corrigé) est dans le champs libellé
- il est possible de saisir l'affichage en menu, le nombre de colonne du formulaire et le nom de chaque colonne
- la table est créée
- la clé primaire est créé automatiquement, elle est de type integer et elle n'accepte pas de null
- un droit est créé sur la table dans om_droit avec le profil utilisateur
- la table est insérée dans le menu option application si l'affichage est ok avec le libellé.

Note : Le nom de la table est automatiquement corrigé car il doit correspondre aux principes de nommage des tables par postgresql : pas de caractères spéciaux ou blanc, pas de majuscule, commence par une lettre ou underscore , ne

contient que des lettres ou des chiffres ou underscore. Le nom d'origine est stocké en libellé et il est modifiable.

Note : Les paramètres sont contenus dans le champs parametres de la table om_tables_parametre au format json

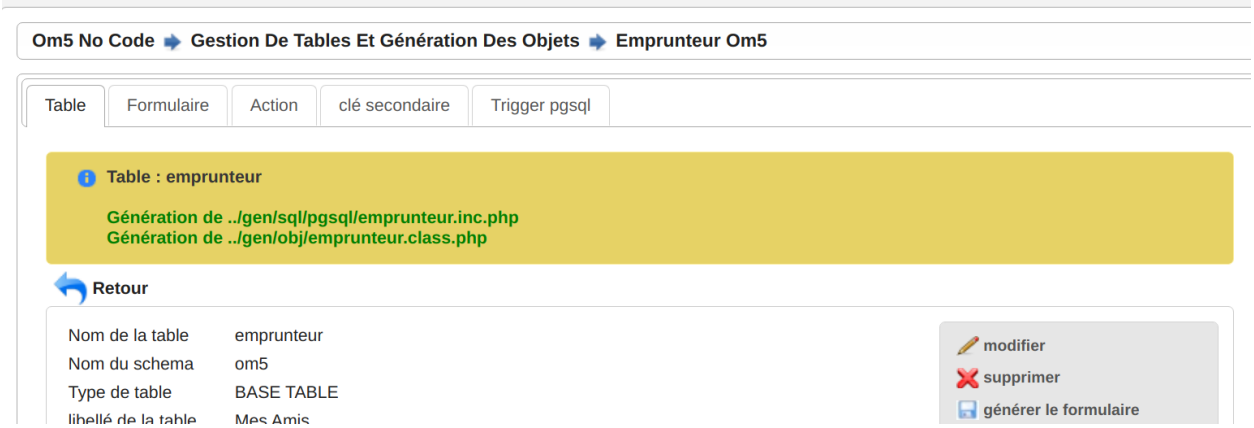
Lors de l'action modifier :
- il est possible de modifier le champs libellé
- il est possible de saisir l'affichage en menu, le nombre de colonne du formulaire et le nom de chaque colonne

Lors de l'action supprimer :
- il est vérifié que la table soit vide
- il est vérifié que le formulaire n'existe plus
- la table est supprimée
- ensuite la sequence est détruite
- puis les paramètres d'om_tables_paramètres et om_forms sont détruits
- et le ou les droits sur la table sont détruits

2.1.2 Action générer le formulaire :

- la liste et le formulaire est générée
- la séquence est créée
- les paramètres des champs (om_forms) et de la table (om_tables_parametre) sont pris en compte.
- il est pris en compte les actions supplémentaires du sous formulaire om_actions

Le message suivant est affiché :



2.1.3 Action supprimer le formulaire :

la liste et le formulaire sont supprimés

2.1.4 Action vider la table :

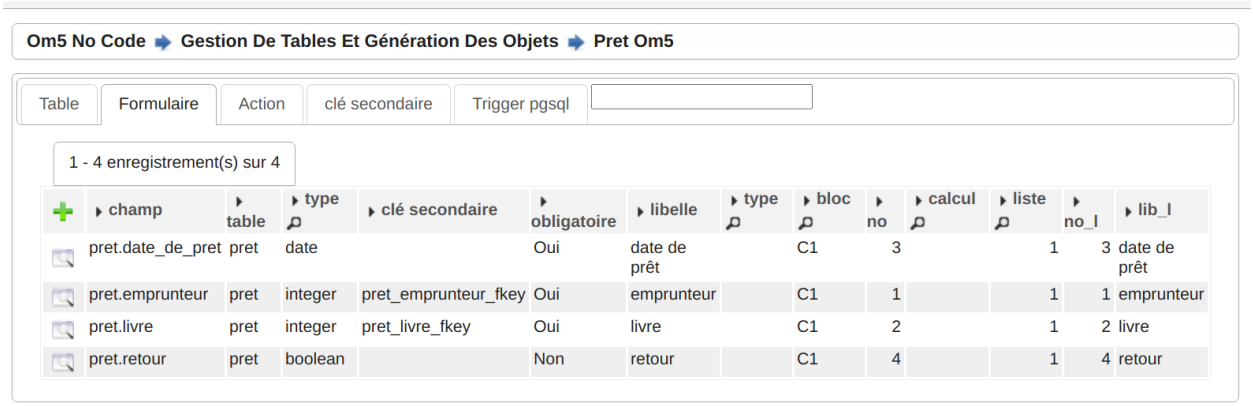
cette action vide la table des enregistrements.

2.1.5 Action ordre des champs par défaut

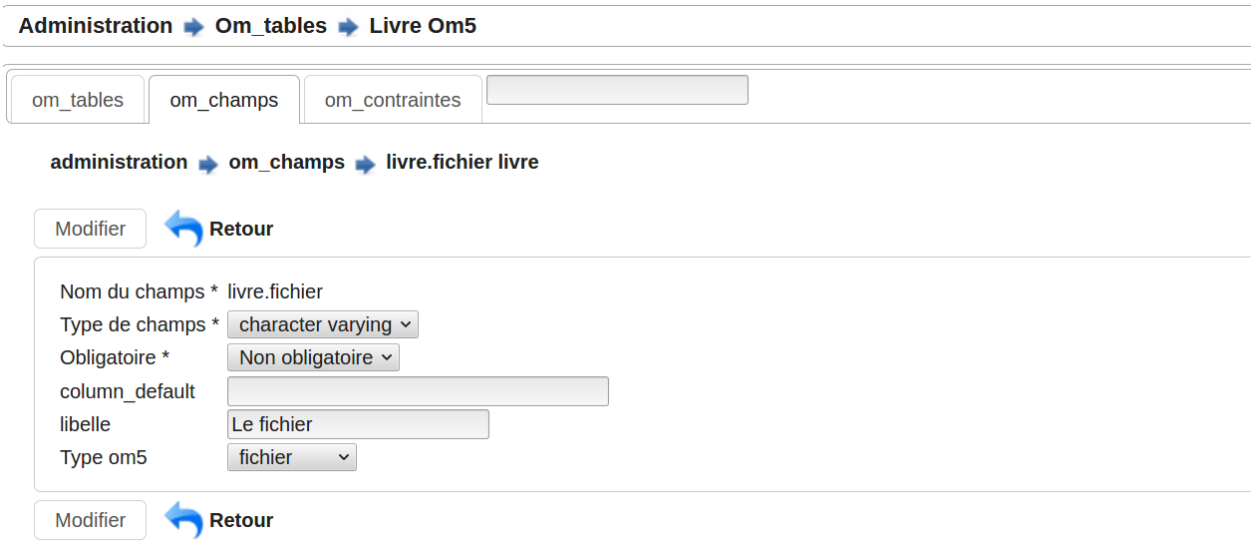
Cette action met les champs dans l'ordre de saisie pour la colonne.

2.2 Saisir des champs dans la table :

Il est possible de lister les champs dans le menu om5-no-code -> option Gestion de Tables sous formulaire de la table :



Il est possible de saisir, modifier et supprimer les champs dans le formulaire des champs :



Le nom du champ doit être unique pour la table et il doit être rempli.

Note : Le nom du champs est automatiquement corrigé car il doit correspondre aux principes de nommage des champs par postgresql : pas de caractères spéciaux ou blanc, pas de majuscule, commence par une lettre ou underscore , ne contient que des lettres ou des chiffres ou underscore. Le nom d'origine est stocké en libellé et il est modifiable.

Les paramètres sont contenus dans le champ parametres de la table om_forms au format json.

Le data_type de champ peut être :

- character varying
- entier (integer)
- décimal (numéric)
- booleen (boolean)
- date
- texte (text)
- clé secondaire

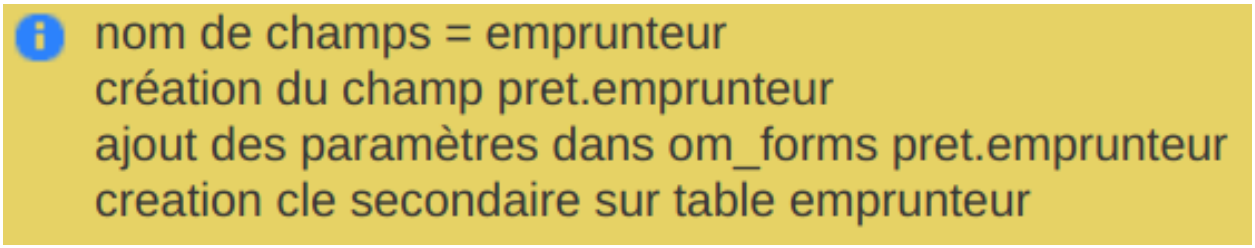
Le type (om) est par défaut celui définit dans om_champs et il est visible :

- Si il est text, il peut devenir html (éditeur html)
- si il est character varying, il peut devenir un champs fichier (file) pour télécharger des fichiers
- il peut être non visible à condition d'être non obligatoire

2.2.1 data_type clé secondaire

Le data type « clé secondaire » propose de choisir la table sur lequel sera créer la clé comme dans l'image suivante :

le message suivant s'affiche :



Note : Le cas « clé secondaire » est particulier et ne peut être utilisé que si une table portant le même nom que le champ existe.

Le champ est alors de type integer.

En cas de suppression, la clé secondaire est automatiquement supprimée.

2.2.2 Champ obligatoire

Le champ peut être obligatoire (n'accepte pas les null) ou pas (accepte les nulls)

Attention, un champ obligatoire ne doit pas être caché

2.2.3 Action sous état pour champ clé secondaire

Il est possible de générer un sous formulaire avec l'action « générer un sous formulaire ». Il faut rendre le sous état actif et l'insérer dans l'état principal.

2.2.4 Position et colonne

En création la position est automatiquement créée dans le bloc C1 (colonne 1) pour le formulaire et le libellé est créée sur la base du nom du champs

La position en liste se fait de la même manière.

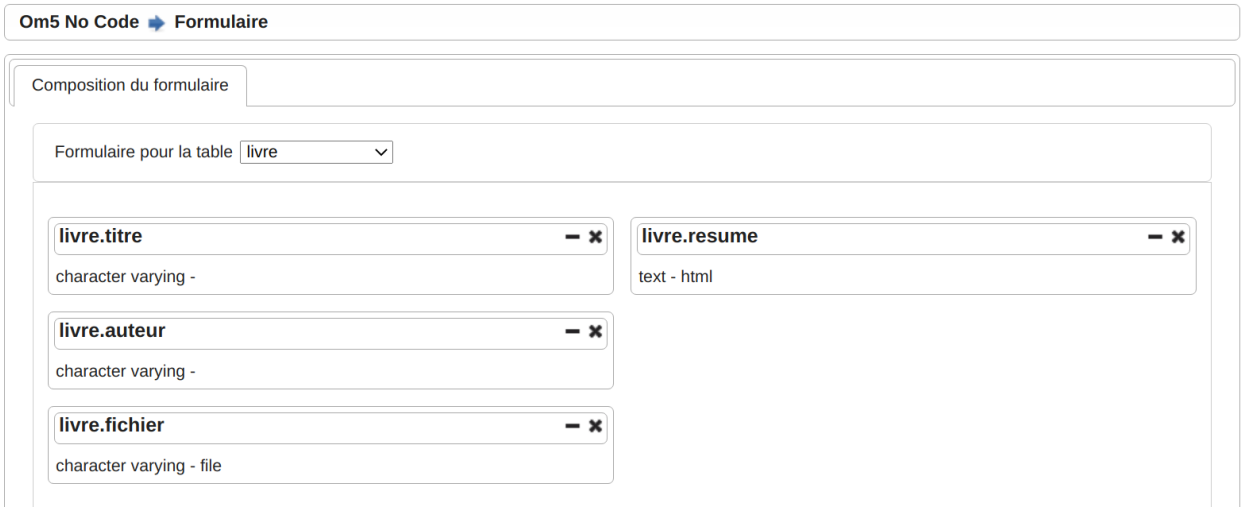
Il est possible de modifier les positions et colonnes dans les formulaires avec l'option du menu om5-no-code : composition formulaires

Il est possible de modifier les positions les listes avec l'option du menu om5-no-code : composition listes

2.2.5 Composition du formulaire :

L'option « composition formulaires » du menu om5-no-code permet de composer le formulaire en disposant les champs dans l'ordre et les colonnes.

La colonne (bloc) et la colonne position est modifiable par glisser/copier.



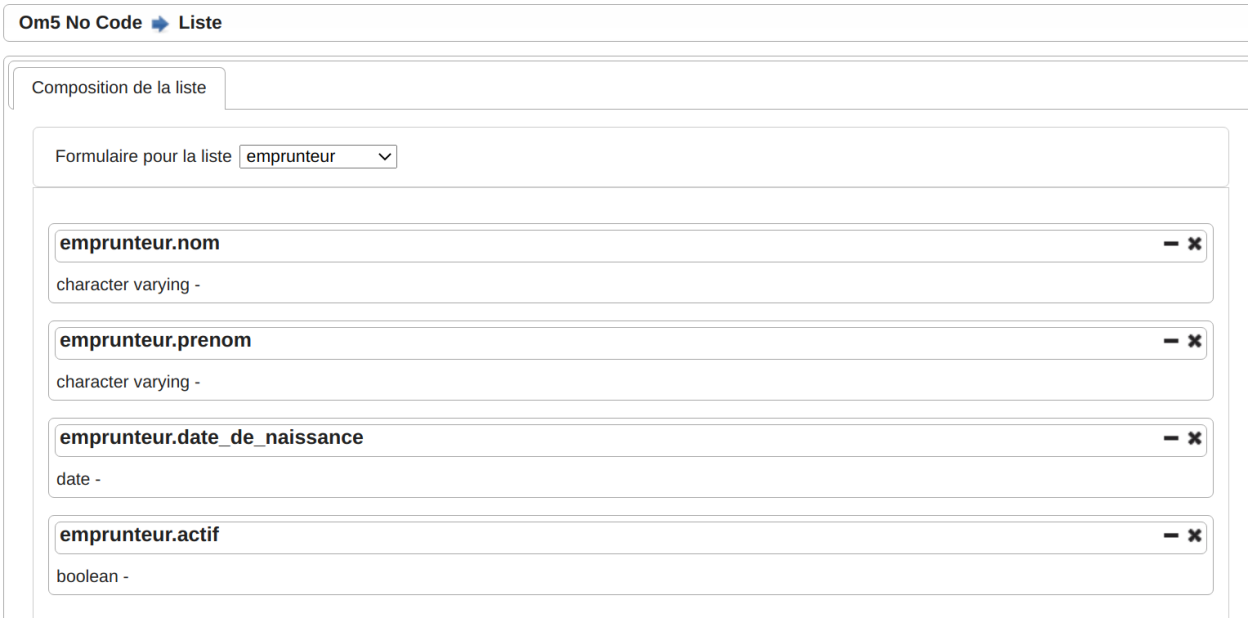
La suppression de champs se fait en cliquant sur la croix en haut à droite du champs.

Note : Dans la composition du fomulaire, la suppression d'un champs, rend le champs non visble avec un libellé vide.

Note : Il faut générer pour prendre en compte toute modification des champs

2.2.6 Composition de liste :

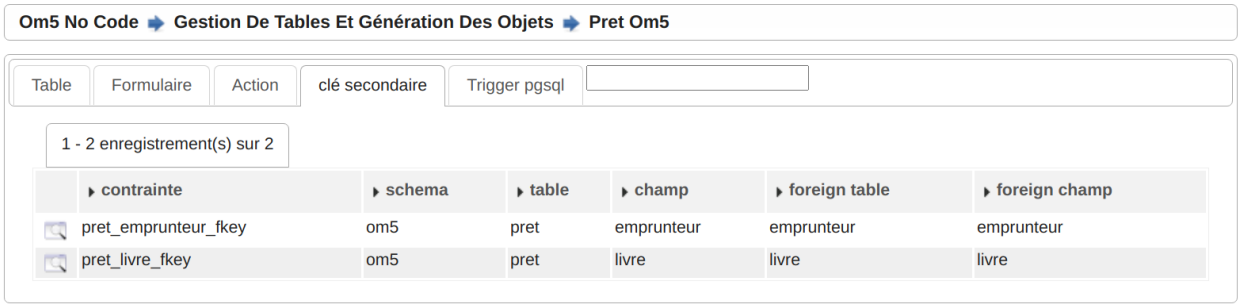
L'option « composition listes » du menu om5-no-code permet de composer les colonnes de la liste en disposant dans l'ordre les colonnes.



Note : Il faut générer pour prendre en compte toute modification des champs

2.3 Supprimer les contraintes de clé secondaire :

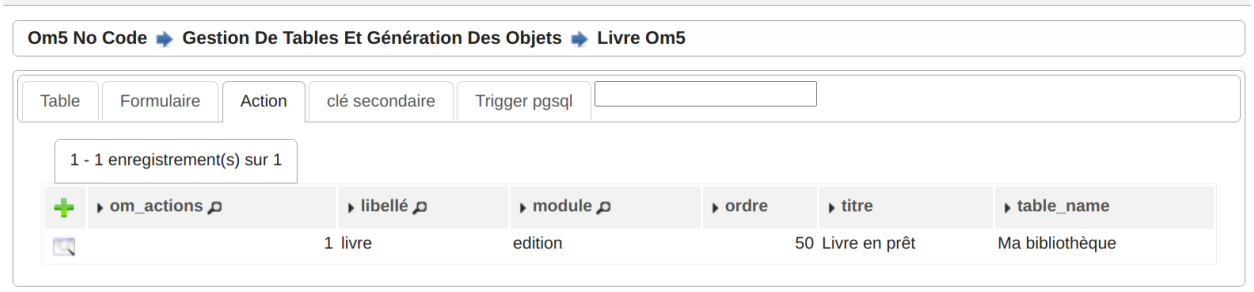
Il est possible de lister les contraintes de clé secondaire dans le menu om5-no-code -> option Gestion de Tables sous formulaire de om_table :



Il est possible de supprimer les contraintes de clé secondaires dans le formulaire des contraintes :

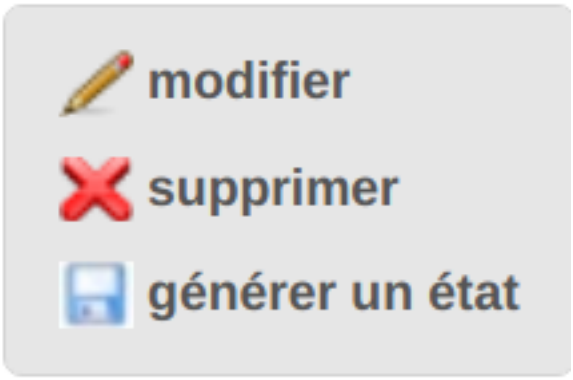
2.4 Saisir des actions :

Il est possible de lister les procédures dans le menu om-no-code -> option Gestion de Tables sous formulaire de om_actions :



Il est possible de modifier et supprimer les actions dans le sous formulaire actions :

Les actions dont les suivantes :

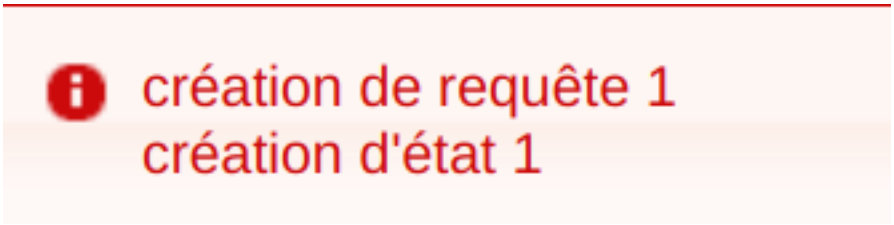


L'action « genérer un état » est accessible pour les actions du module « édition ».

2.4.1 Générer un état :

Cette action permet de générer un état dans la table om_état et une requête dans la table om_requête.

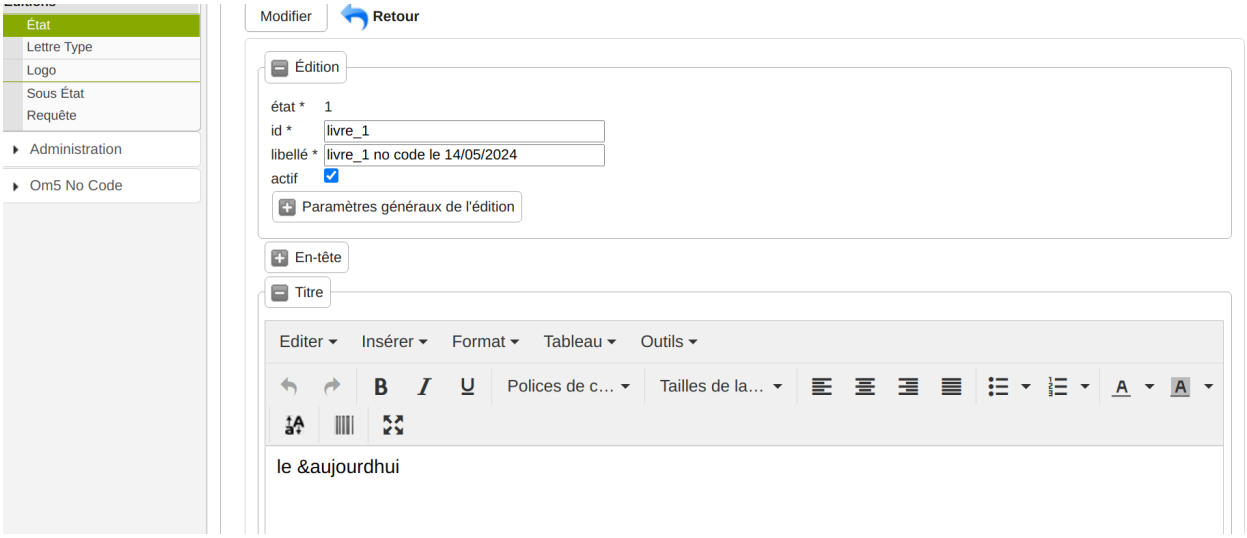
Le message suivant est affiché :



Note : L'état généré est non actif. Il faut aller sur le menu « paramètre » option état et paramétrer l'état en mode actif. La prise en compte de l'action se fait par la génération de la table (action générer d'om_tables)

Il faut donc aller dans le menu paramètres dans l option Etat :

et modifier l'état correspondant, il porte comme identifiant le nom de la table avec le numéro de l'action :



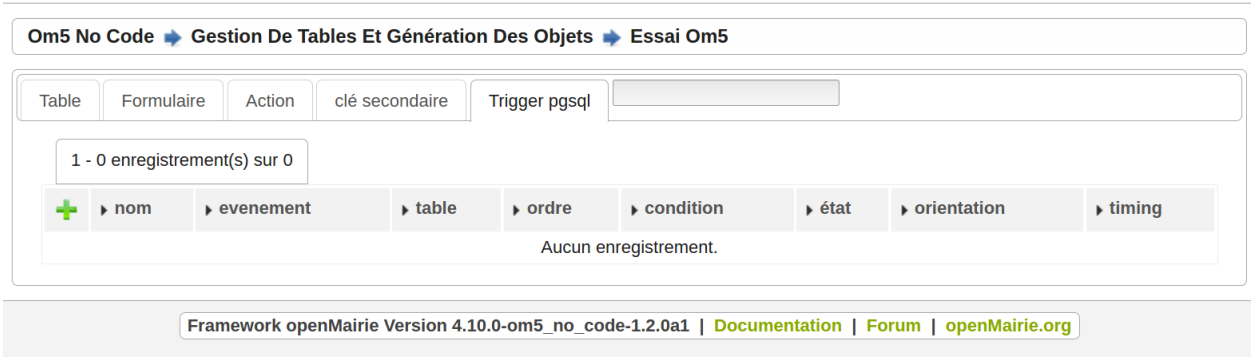
Ne pas oubler de cocher actif pour que l'état puisse être utilisé dans l'action d'édition.

En générant dans om_tables (action générer le formulaire), l'action est accessible dans le formulaire comme dans l'image suivante (action : livre en prêt :

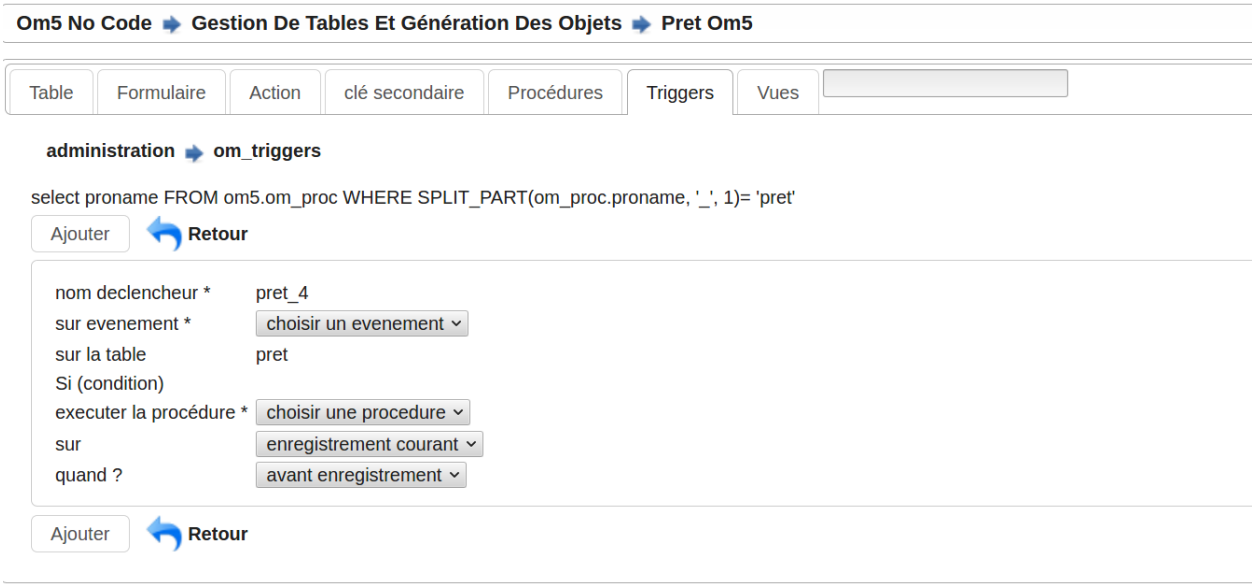
2.5 Saisir des triggers :

Note : Attention ce formulaire est en cours de développement.

Il est possible de lister les triggers dans le menu om-no-code -> option om_tables - onglet trigger



Il est possible de saisir et supprimer les triggers dans le formulaire des triggers :



Note : Il faut créer la procédure avant le trigger qui lance la procédure.

Les options de l'assistant trigger sont les suivantes :

2.5.1 nom du trigger

Le nom du trigger est automatique : nom de la table + numéro d'ordre.

2.5.2 Evénement :

Le trigger s'exécute lors de :
- la création d'enregistrement (insert)
- la modification d'un enregistrement (update)
- la destruction d'un enregistrement (delete)

2.5.3 table

Par défaut la table du formulaire

Non modifiable

2.5.4 ordre

ordre d'execution des trigers : non implémenté

2.5.5 condition

condition d'exécution du trigger : non implémenté

2.5.6 Procédure

Choix de la procédure de la table (voir chapitre procédure)

2.5.7 sur ?

Une seule option : enregistrement courant (for each row).

L'option traitement n'est pas implémenté (for each statement)

2.5.8 quand ?

avant enregistrement : insert, update

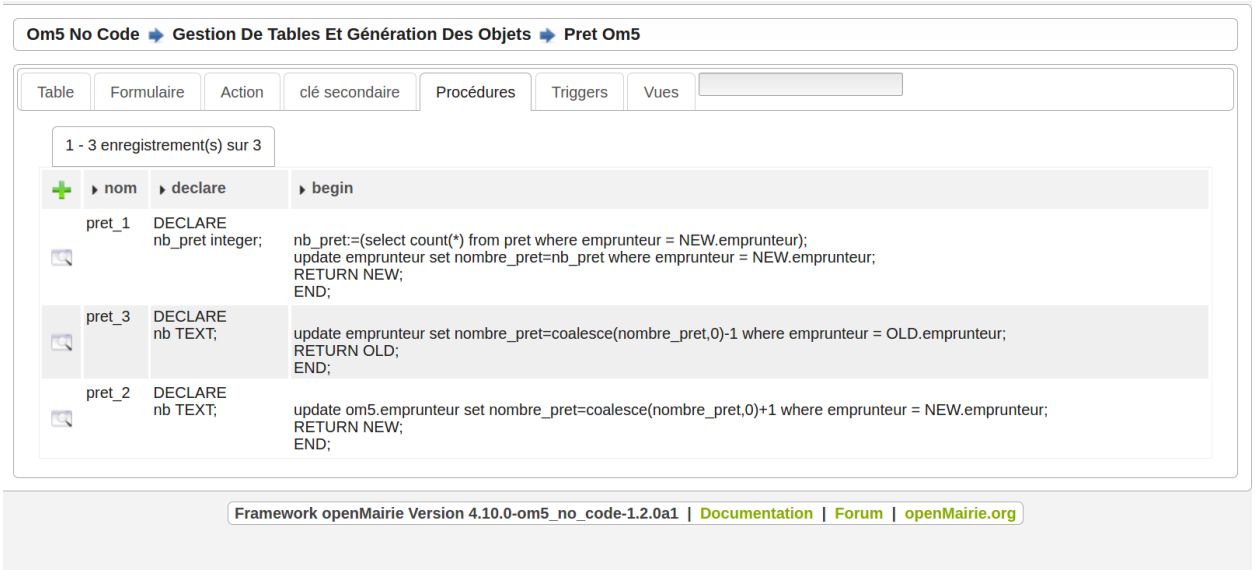
après enregistrement : insert, update ou delete

L'option INSTEAD OF n'est pas implémentée.

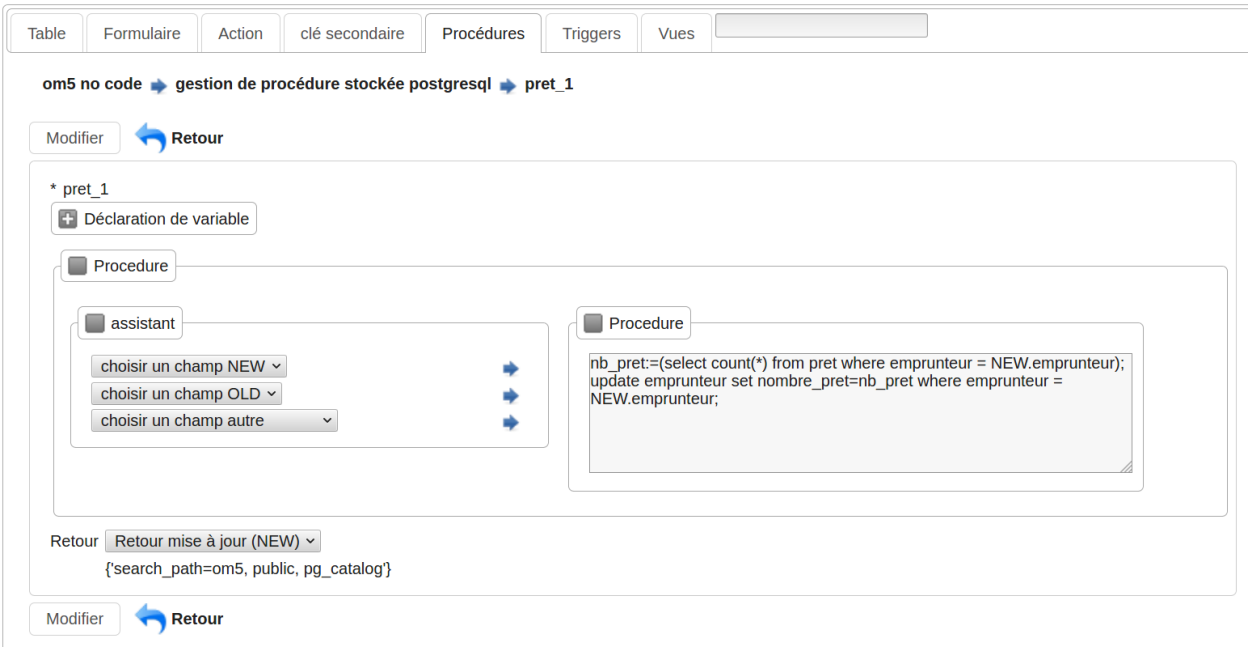
2.6 Saisir les procédures :

Note : Attention ce formulaire est en cours de développement. Les procédures seront liés aux triggers pgsql de la table.

Il est possible de lister les procédures dans le menu om-no-code -> option om_tables - onglet procédures



Il est possible de saisir, modifier et supprimer les procédures dans le formulaire des procédures :



2.6.1 nom du procedure

Le nom de la procédure est automatique : nom de la table + numéro d'ordre.

2.6.2 table

Par défaut la table du formulaire Non modifiable

2.6.3 Déclaration de variable

Champs optionnel utile si il est utilisé des variables internes :

2.6.4 Procédure :

Opération de la procédure

2.6.5 Retour

- retour mise à jour (NEW) pour INSERT et UPDATE. NEW is NULL dans un DELETE triggers
- retour delete (OLD) pour UPDATE ou DELETE. OLD is NULL dans un INSERT triggers

2.6.6 automaticité du code

Une partie du code est créé automatique et est complété par les champs : declare, procédure et retour.

exemple : calcul du nombre de prêt par emprunteur

```
-- automatique (nom de la procédure)
        CREATE OR REPLACE FUNCTION om5.pret_1()
        RETURNS trigger AS
        $BODY
        $DECLARE

-- champ déclare :
        nb_pret integer;

-- automatique
        BEGIN

-- champ procédure :
        nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur);
        update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur;

-- champ retour (select):
        RETURN NEW;

-- automatique : langage et schéma par défaut :
        END;
        $BODY$
          LANGUAGE plpgsql VOLATILE
          COST 100;
        ALTER FUNCTION om5.pret_1() SET search_path=om5, public, pg_catalog;
```

2.6.7 Assistant :

L'assistant permet de pouvoir récupérer les champs de la table courante :

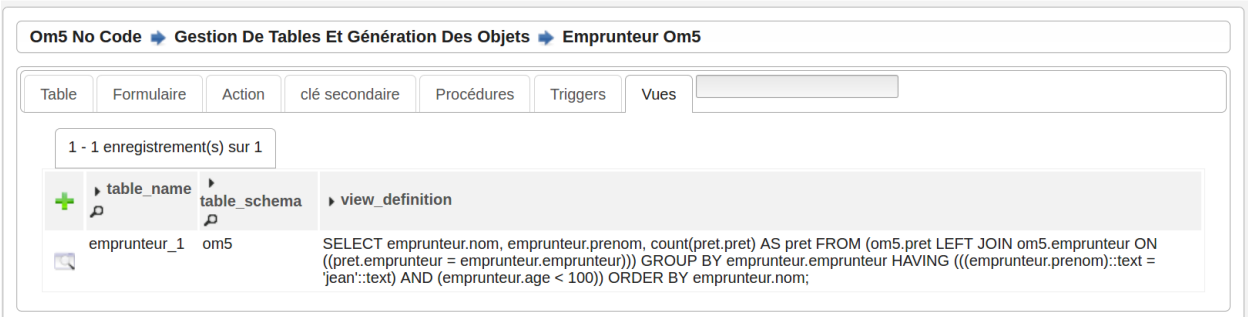
- NEW : valeur après la mise à jour
- OLD ; valeur avant la mise à jour

et les champs de la base.

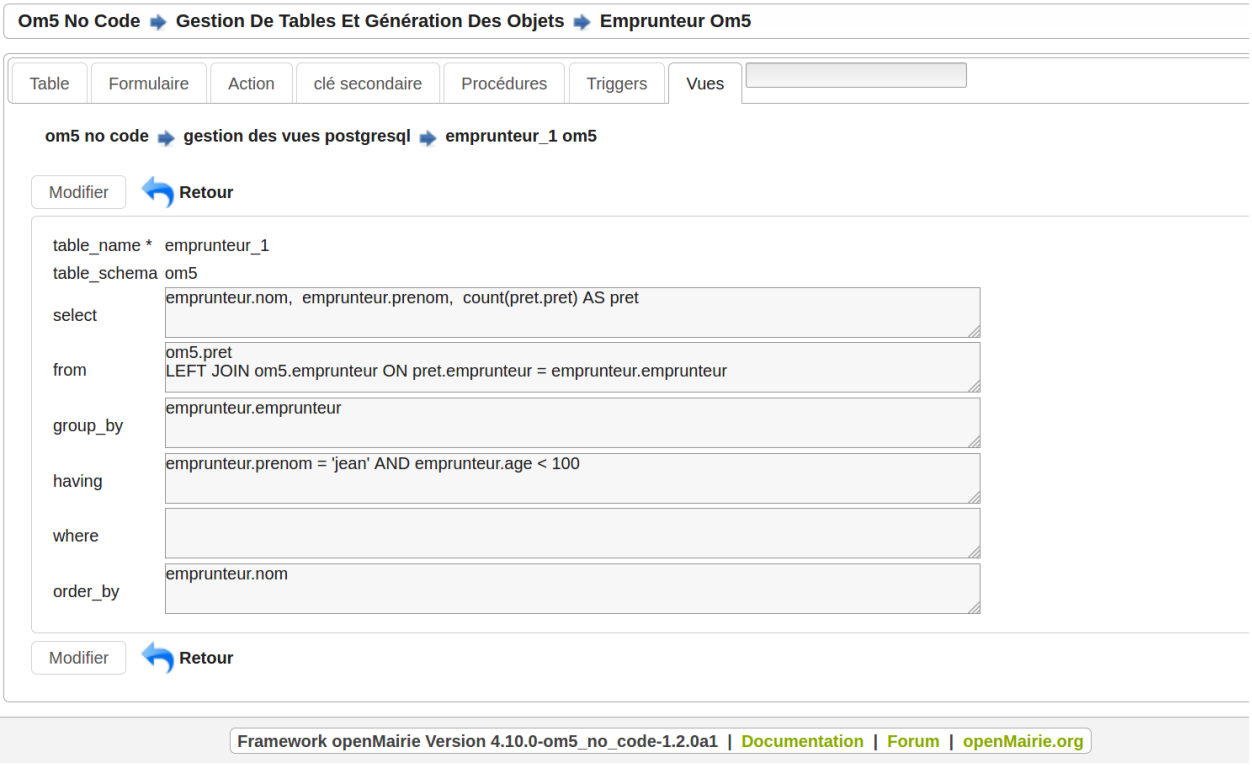
2.7 Saisir des vues :

Note : Attention ce formulaire est en cours de développement. Il est prévu une aide à la construction de vue notament d'agrégat pour les listes

Il est possible de lister les triggers dans le menu om-no-code -> option om_tables - onglet vues



Il est possible de saisir, modifier et supprimer les vues dans le formulaire des vues :



2.7.1 nom de la vue

Le nom de la vue est automatique : nom de la table + numéro d'ordre.

2.7.2 table

Par défaut la table du form

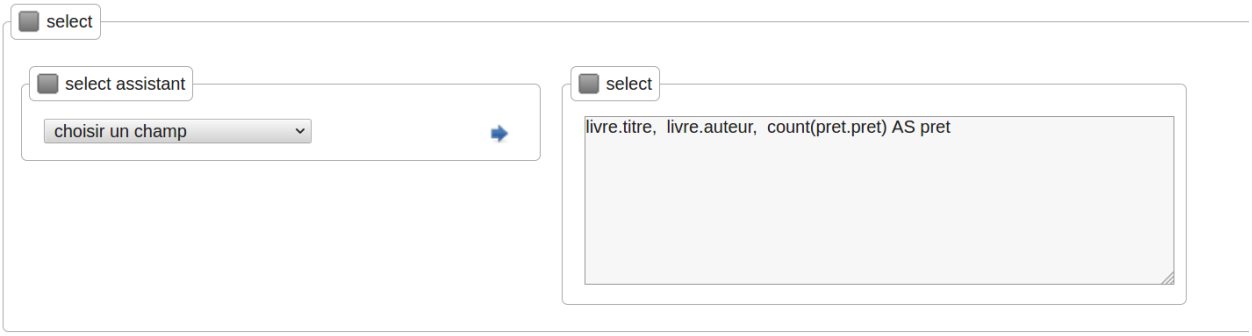
ulaire Non modifiable

2.7.3 assistant select

Liste des champs à sélectionner séparés par une virgule

```
livre.titre,  livre.auteur,  count(pret.pret) AS pret
```

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

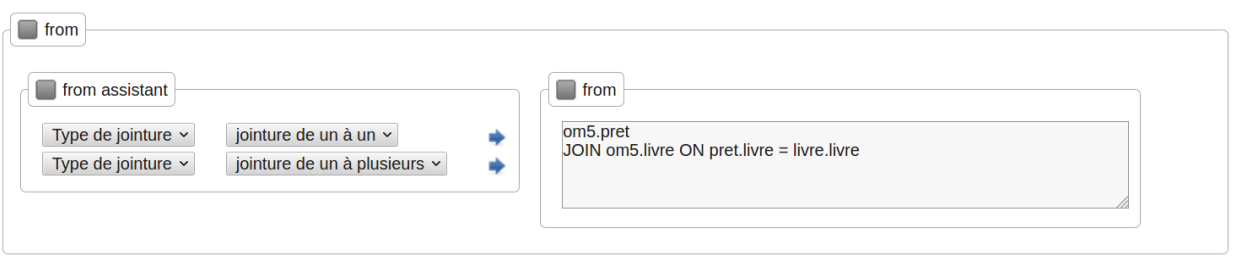


2.7.4 assistant from

Tables de la vue, par défaut la table courante.

```
om5.pret
JOIN om5.livre ON pret.livre = livre.livre
join om5.evaluation ON livre.evaluation=evaluation.evaluation
```

Il est possible d'utiliser l'assistant pour construire les jointures :



La jointure de un à un est une jointure de la table en cours sur une autre table.

La jointure de un à plusieurs est une jointure d'une autre table sur la table en cours.

exemple :

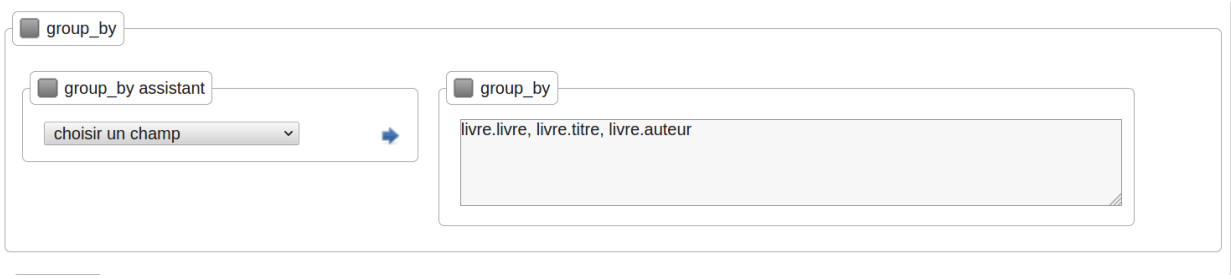
- prêt est une jointure de un à plusieurs sur livre
- evaluation est une jointure de un à un sur livre

2.7.5 assistant group_by

Dans le cas d'une requête d'agrégation

```
livre.livre, livre.titre, livre.auteur
```

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

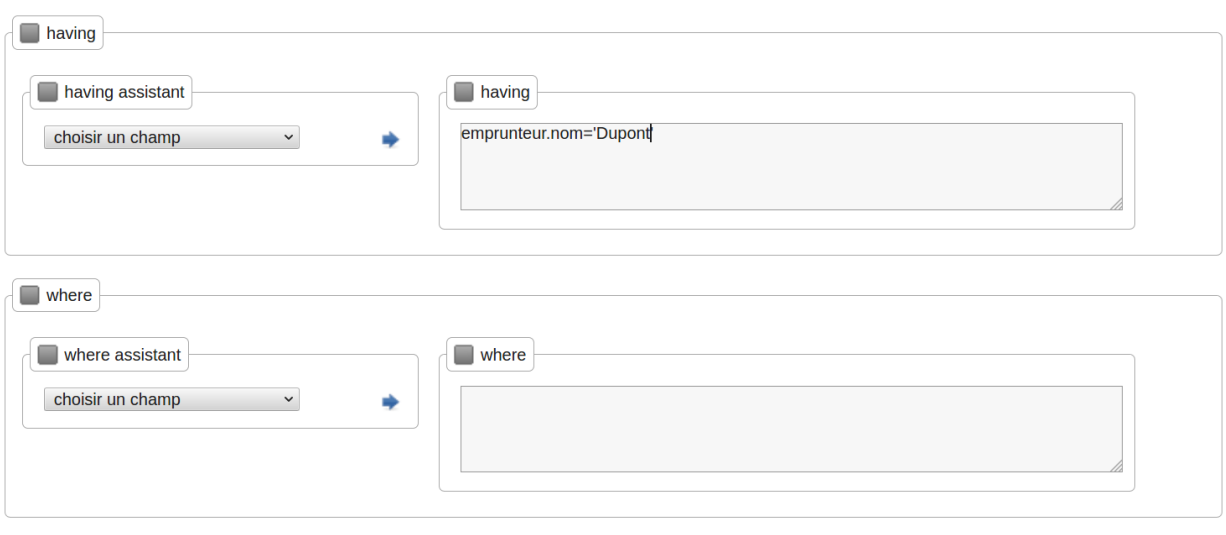


2.7.6 assistant having ou where

Having dans le cas d'une requête d'agrégation ou where dans les autres cas

les conditions sql sont séparées par les opérateurs AND, OR et/ou des parenthèses

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

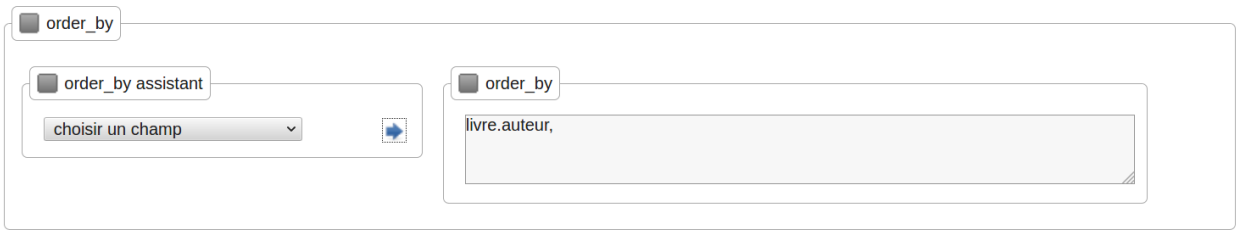


2.7.7 assistant order_by

Tri des enregistrements

Champs séparés par une virgule

Il est possible d'utiliser l'assistant pour rechercher les champs de la base :

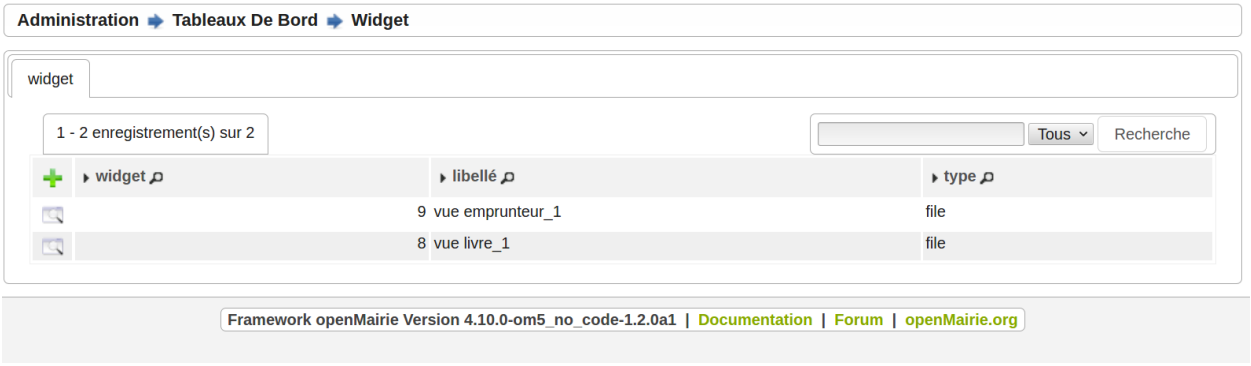


Note : Si vous utilisez l'assistant, la dernière virgule et le dernier retour charriot sont enlevés par traitement pour les champs select, group_by et order_by

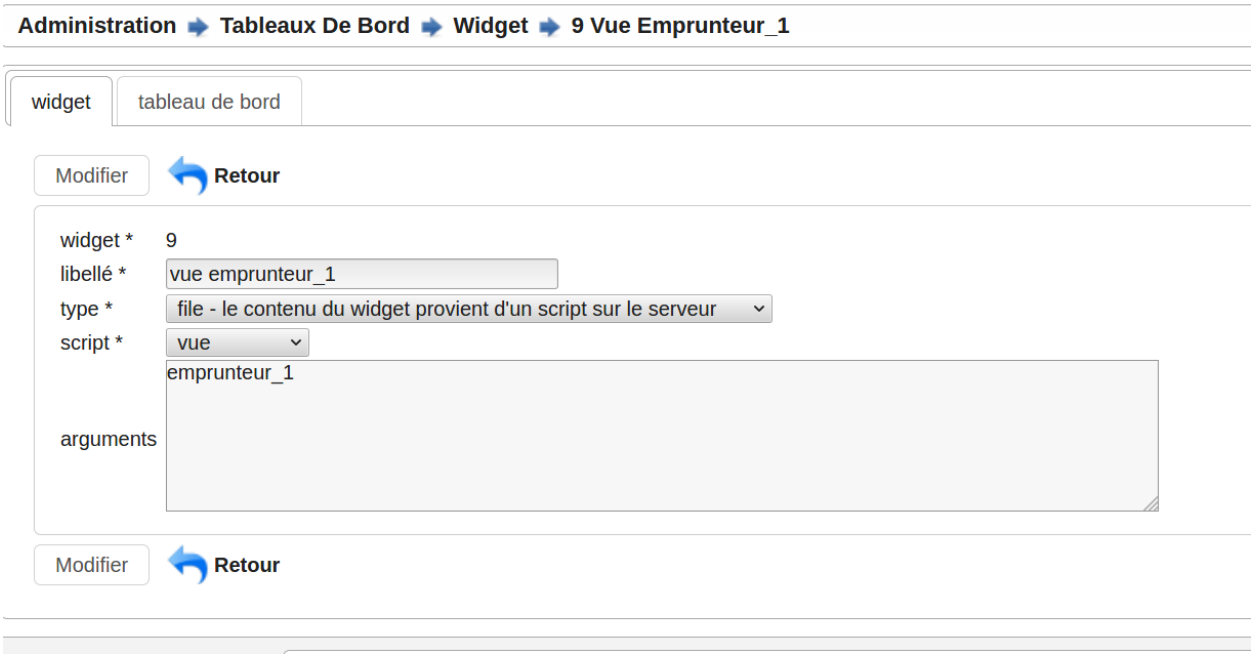
2.7.8 action « créer un widget » :

Il est possible de créer un widget dans le tableau de bord si celuici n est pas existant

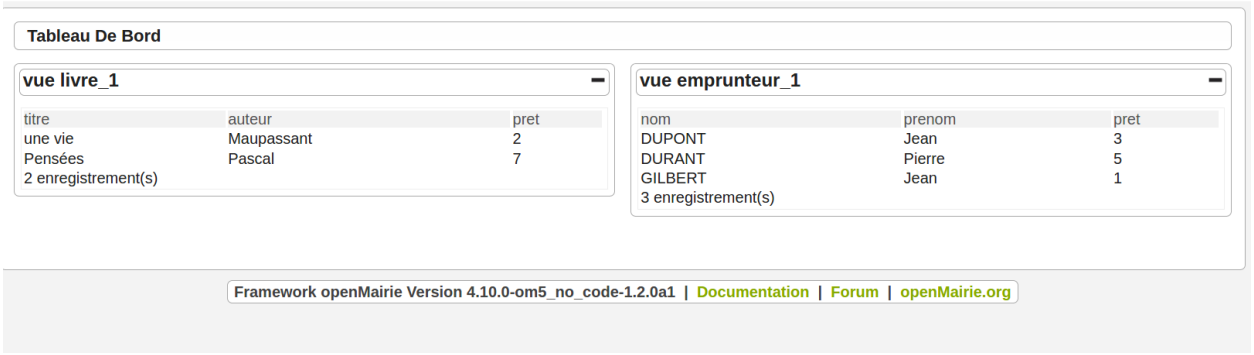
Le widget est créé dans administration -> widget :



Il est possible de modifier le titre. (l'argument indique la vue concernée à afficher)



Avec la composition du tableau de bord, le widget peut s'afficher dans le profil concerné :



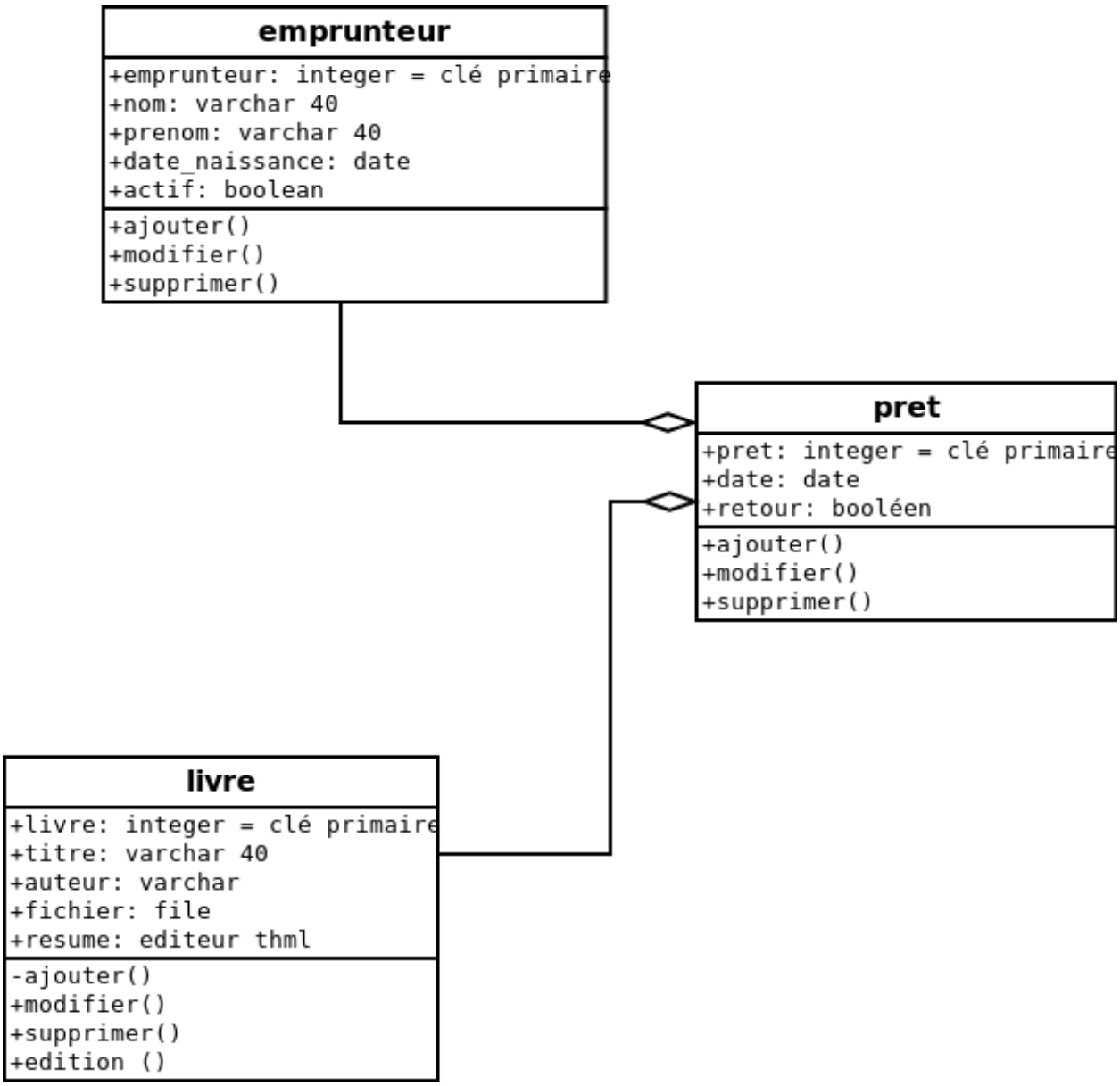
2.8 Exemple : la gestion de ma bibliothèque :

Note : Le sql de l'exemple est dans le fichier data/pgsql/om5_exemple.sql

Il est proposé à titre d'exemple, de créer l'application ma bibliothèque et le prêt à mes amis :

- dans un premier temps, on analyse ce que l'on veut faire et on peut produire un diagramme de classe
- on créé les tables avec om_tables en modifiant éventuellemnt les paramères d'om_tables_parametre
- on créé les champs correspondants avec om_champs en modifiant éventuellemnt les paramères d'om_forms
- on compose le formulaire avec l'option composeur du menu om5-no-code
- on génére les formulaires pour chaque table.
- on peut générer une action « générer un état » (sous formualire action)
- on peut générer un sous état (sous formulaire champs)
- il est possible de créer des triggers et des procédures
- il est possible de créer des vues et créer des widgets dans le tableau de bord

2.8.1 Diagramme de classe de la gestion de bibliothéque :

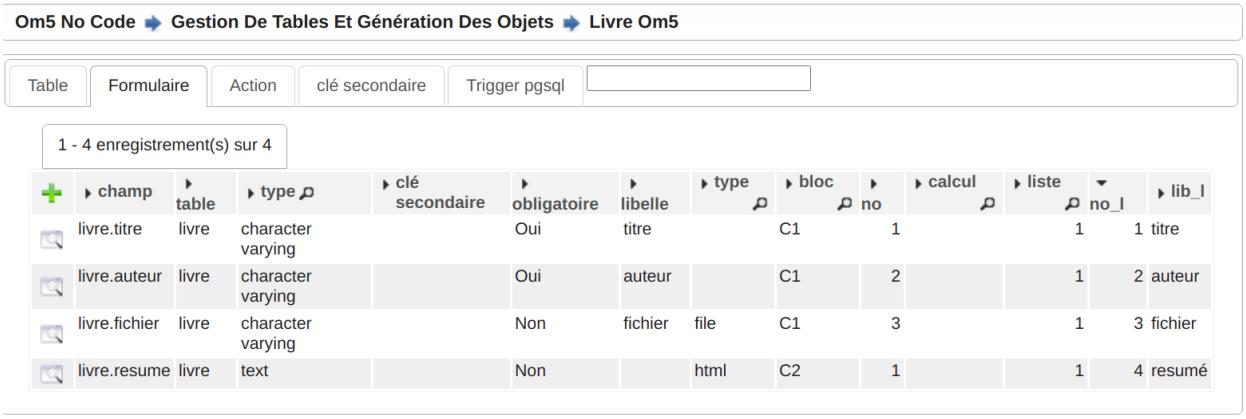


2.8.2 La création des tables :

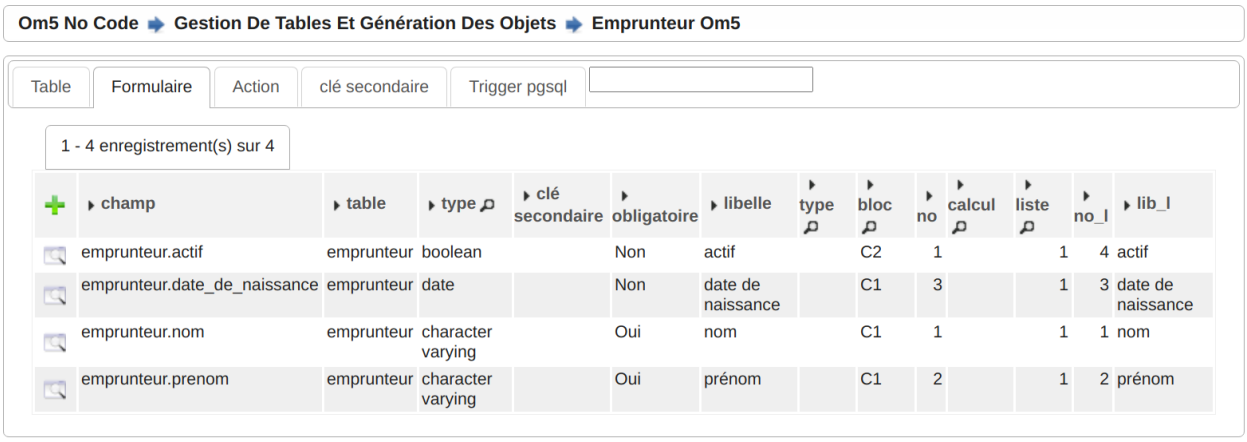
Les tables suivantes ont été crées avec om_tables : livre, emprunteur , pret

2.8.3 Les champs suivants ont été créés pour chaque table :

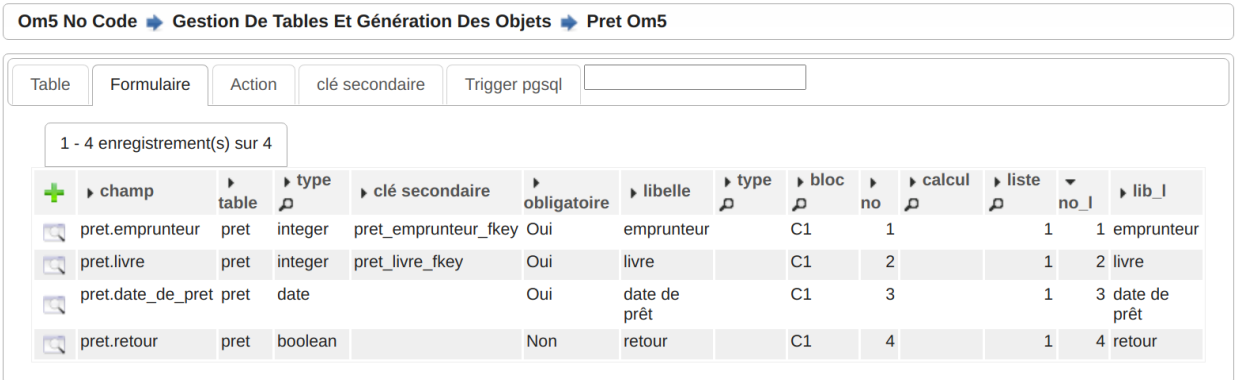
livre



emprunteur

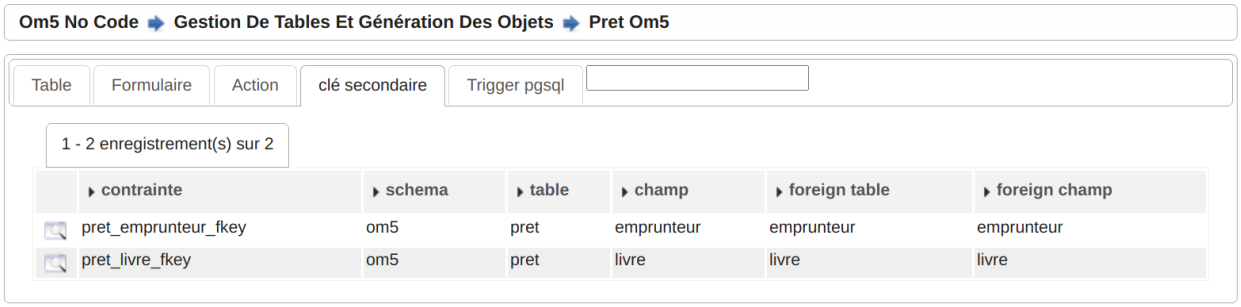


pret



Les contraintes de clé secondaires sont visibles dans le sous formulaire om_contrainte de om_tables (option du menu pg_admin) :

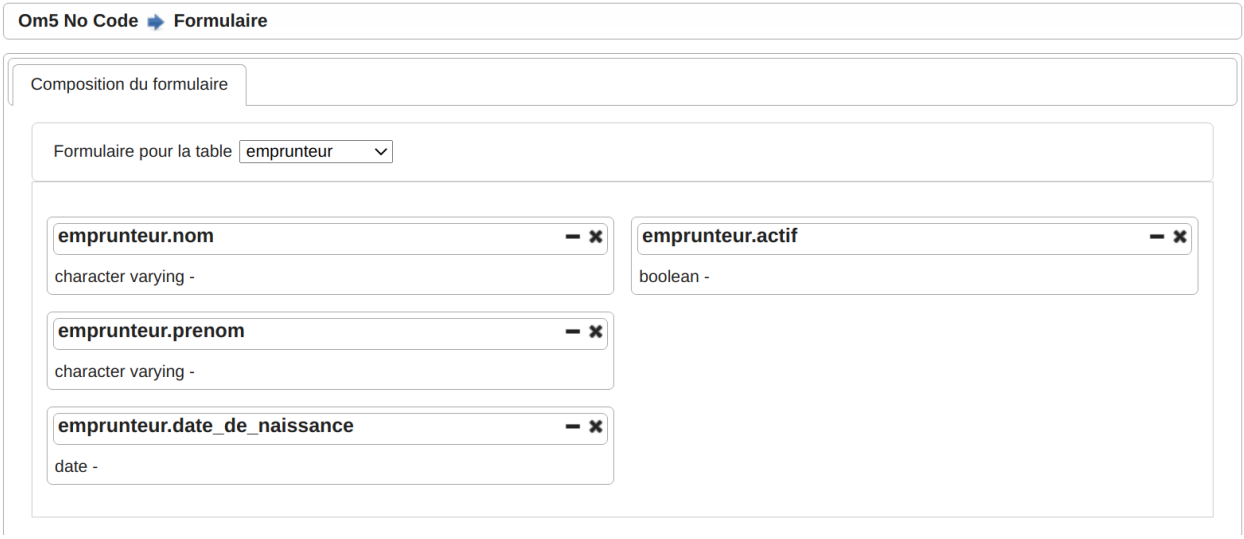
— pret_livre_fkey
— pret_emprunteur_fkey



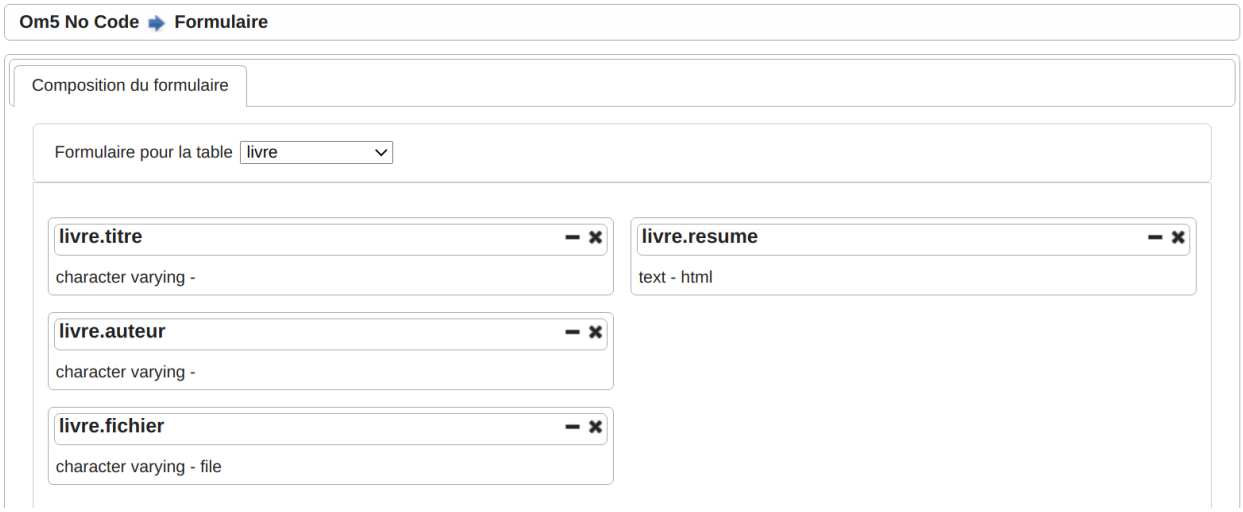
Note : Les sous formulaires sont créés par les clés secondaires.

2.8.4 Les formulaires ont été composés de la manière suivante :

Pour la table emprunteur :

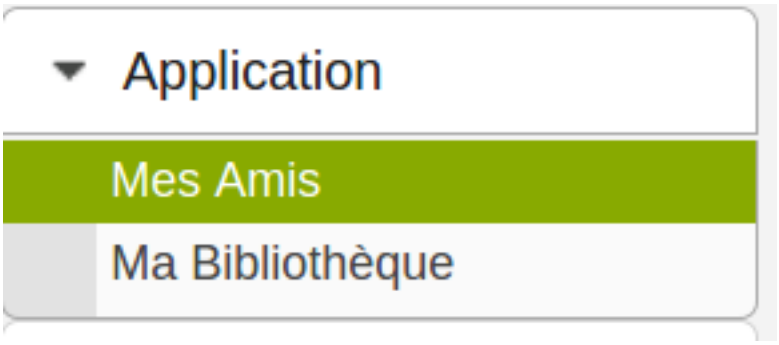


pour la table livre :



2.8.5 Le menu application :

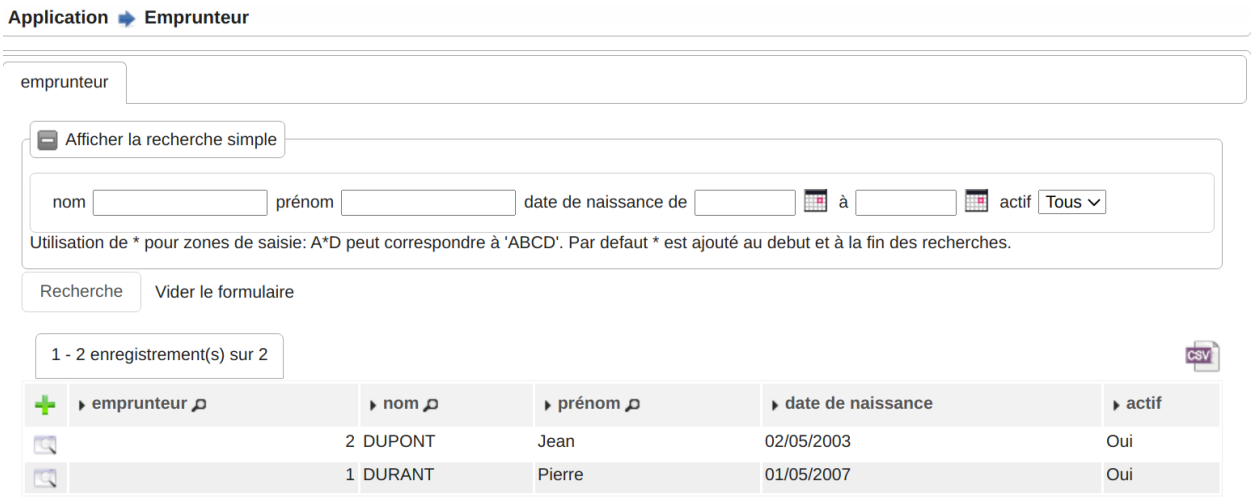
Le menu option application est créé au fur et à mesure de la création des tables :



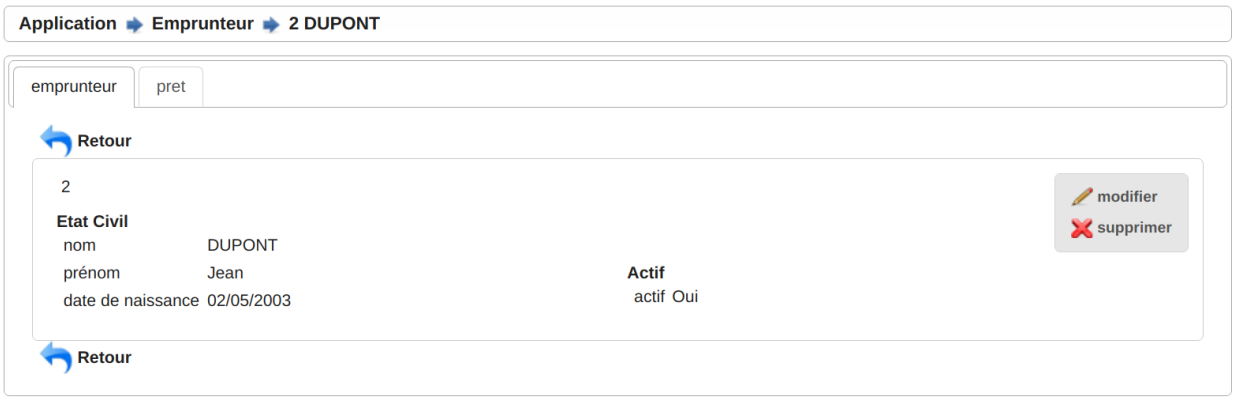
2.8.6 Les formulaires et le sous formulaire généré :

Il faut ensuite générer le formulaire avec chaque table (voir action générer d'om_table).

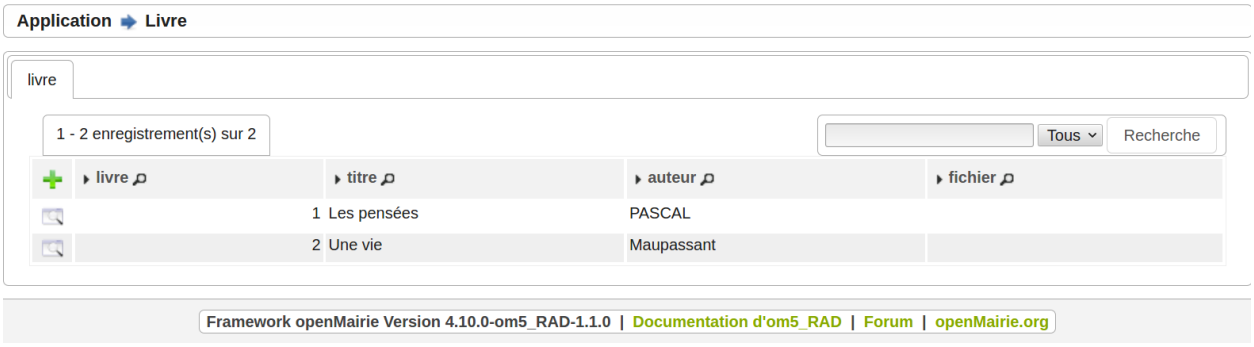
Il est possible de lister les emprunteurs dans le menu application -> option « mes amis »



Il est possible de saisir, modifier et supprimer les aemprunteurs dans le formulaire suivant :

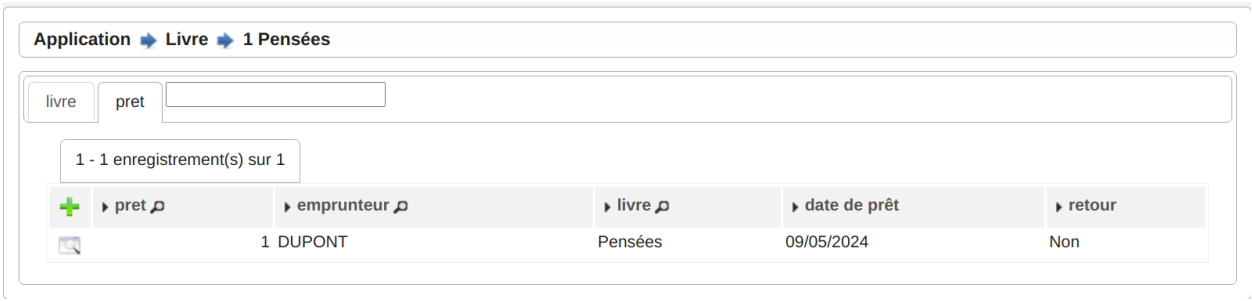


Il est possible de lister les livres dans le menu application -> option « ma bibliothèque »

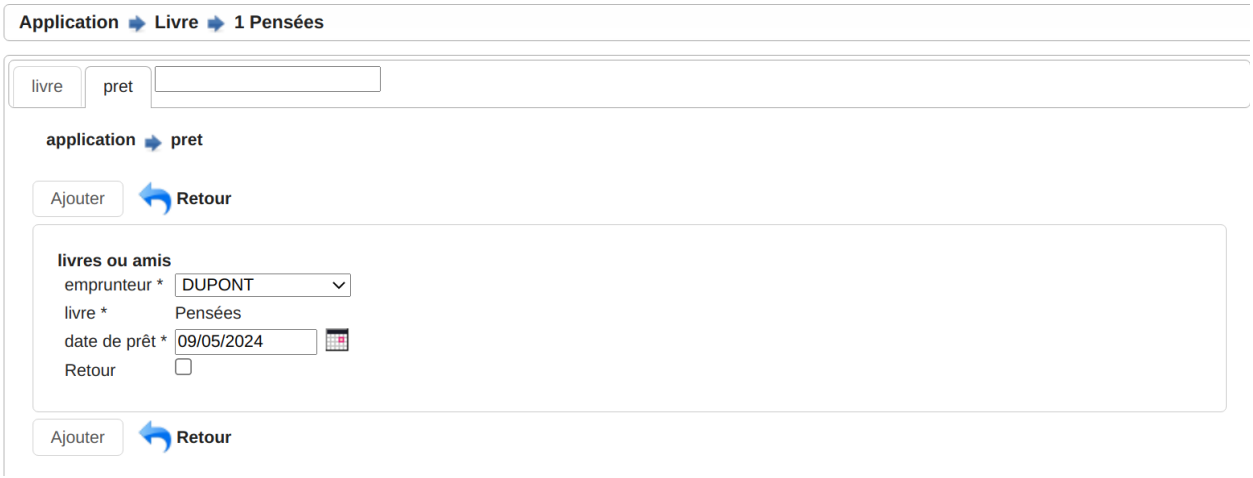


Il est possible de saisir, modifier et supprimer les livres dans le formulaire suivant :

Il est possible de lister les livres empruntés dans le menu application -> dans le sous formulaire de livre ou d'emprunteur :



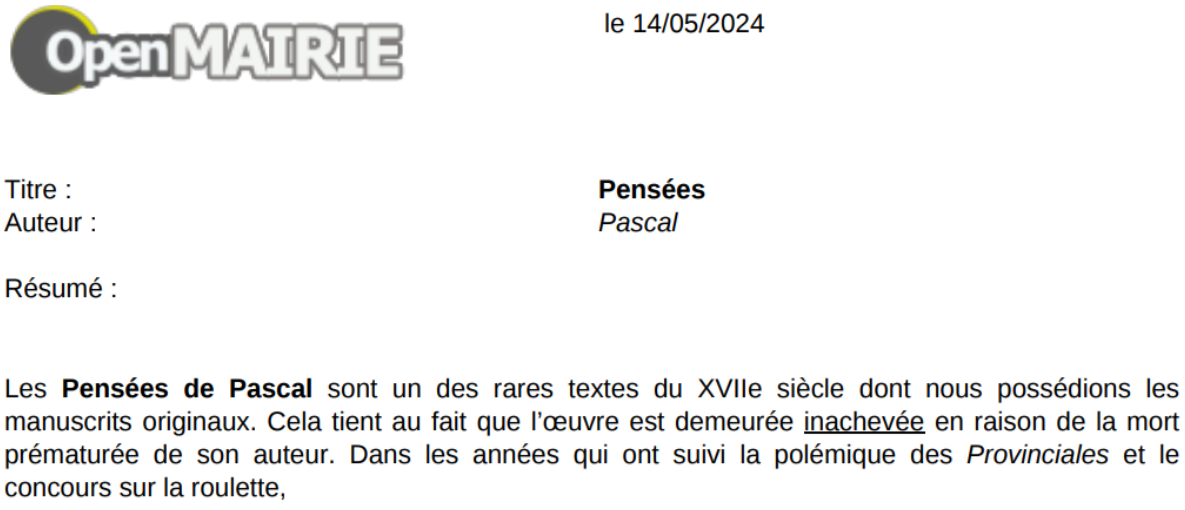
Il est possible de saisir, modifier et supprimer les emprunts de livre dans le formulaire généré :



2.8.7 La génération des éditions :

Voir le chapître précédent : « saisie des actions »

Il est possible en cliquant sur l'action « livre en prêt » d'accéder à l'état suivant :



2.8.8 Les procédures

Exemple de procèdures crées avec les formulaires om5

Calcul d'age

```
CREATE OR REPLACE FUNCTION om5.emprunteur_1()
  RETURNS trigger AS
$BODY
$DECLARE
BEGIN
NEW.age =  date_part('year', age(NEW.date_de_naissance));
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.emprunteur_1() SET search_path=om5, public, pg_catalog;
```

Nombre de prêt par emprunteur

```
CREATE OR REPLACE FUNCTION om5.pret_1()
  RETURNS trigger AS
$BODY
$DECLARE
nb_pret integer;
BEGIN
nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur);
update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur;
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.pret_1() SET search_path=om5, public, pg_catalog;

*** OU :

CREATE OR REPLACE FUNCTION om5.pret_2()
  RETURNS trigger AS
$BODY
$DECLARE
nb TEXT;
BEGIN
update om5.emprunteur set nombre_pret=coalesce(nombre_pret,0)+1 where emprunteur =␣
↪NEW.emprunteur;
RETURN NEW;
END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.pret_2() SET search_path=om5, public, pg_catalog;

*** ET (en delete)

CREATE OR REPLACE FUNCTION om5.pret_3()
  RETURNS trigger AS
$BODY$DECLARE
nb TEXT;
BEGIN
update emprunteur set nombre_pret=coalesce(nombre_pret,0)-1 where emprunteur = OLD.
↪emprunteur;
RETURN OLD;
```

(suite sur la page suivante)

(suite de la page précédente)

```
END; $BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION om5.pret_3() SET search_path=om5, public, pg_catalog;
```

2.8.9 Triggers pour calcul d'age

Exemple de triggers crées avec les formulaires om5

```
*** en insert

CREATE TRIGGER emprunteur_1
  BEFORE INSERT
  ON om5.emprunteur
  FOR EACH ROW
  EXECUTE PROCEDURE om5.emprunteur_1();

*** en update

CREATE TRIGGER emprunteur_2
  BEFORE UPDATE
  ON om5.emprunteur
  FOR EACH ROW
  EXECUTE PROCEDURE om5.emprunteur_1();
```

2.8.10 Vues pour tableau de bord : nombre de prêt

Exemple de vues crées avec les formulaires om5

```
*** par emprunteur

CREATE OR REPLACE VIEW om5.emprunteur_1 AS
 SELECT emprunteur.nom,
        emprunteur.prenom,
        count(pret.pret) AS pret
   FROM om5.pret
         LEFT JOIN om5.emprunteur ON pret.emprunteur = emprunteur.emprunteur
  GROUP BY emprunteur.emprunteur
 HAVING emprunteur.age < 25
  ORDER BY emprunteur.nom;

*** par livre

CREATE OR REPLACE VIEW om5.livre_1 AS
 SELECT livre.titre,
        livre.auteur,
        count(pret.pret) AS pret
   FROM om5.pret
         JOIN om5.livre ON pret.livre = livre.livre
  GROUP BY livre.livre, livre.titre, livre.auteur;
```

CHAPITRE 3

Technique

Il est présenter dans ce chapitre la description téchnique du projet dans le cadre du framework openMairie :

3.1 Le framework openMairie :

Le projet om5-no-code est un projet de développement rapide qui permet une prise en main rapide du framework.

Il est composer de 2 modules :
- un installateur du framework,
- les formulaires de l'option om5-no-code qui permet de créer les tables et de générer des formulaires et des éditions.

Le but est de créer rapidement une mini application ou une maquette sans écrire une seule ligne de code.

Le public visé est les développeurs peu expérimentés ou débutant, en proposant par la suite une montée en puissance pour maîtriser l'ensemble du framework (formation, assistance technique, forum ...). Il s'agit de mettre en place un développement « no code » accessible avec une connaissance de modèle de données mais aussi un modèle de développement pour les utilisateurs avancés car om5-no-code veut être aussi une alternative aux :
- applications sur tableur qui deviennent souvent complexes avec beaucoup de colonnes et de nombreuses redondances
- aux gestionnaires de contenu qui n'intégrent pas les règles du modèle de CODD des bases de données (cle/valeur, redondances)
- aux systèmes d'information géographique qui n'intégrent pas la dimension relationnelle

Ce projet qui prolonge le module du générateur, veut s'intégrer dans les développements futurs d'openMairie :
- poc_de_query : pour le passage aux versions php supérieures à 8.0 et faciliter l'installation des composants
- poc_view_js : pour l'utilisation de librairie d'affichage java script plus moderne.

3.1.1 Le projet ne modifie pas le core :

Le projet utilise le framework version 4.10.0 et ne modifie aucune classe du core.

Le projet modifie seulement 4 fichiers :

- le fichier index.php à la racine car il appelle l'installateur dans le répertoire app (om_setup_config.php), si le fichier dyn/database.inc.php n'existe pas,
- le fichier data/pgsql/install.sql avec l'ajout de la création des vues : init_gen.sql,
- le fichier framework_openmairie.class.php par surcharge de la méthode set_config__menu() en ajoutant 2 options :
  - l'option « application » qui présente la liste des tables générées par l'utilisateur,
  - l'option « om5-no-code » qui appelle les formulaires d'administration de la base de données : om_tables, avec les formulaire om_champs et om_contrainte

Les autres ajouts sont uniquement des objets générés et leurs surcharges.

3.1.2 Le projet se base sur des vues sur information_schema :

Il est créé dans le répertoire data/pgsql 3 vues sur information_schema :
- om_tables : cette vue contient les tables du schéma sauf celles préfixées par om
- om_champs : cette vue contient les champs des tables ci dessus
- om_contraintes : cette vue contient les clés secondaires des tables ci dessus
- om_vues : cette vue contient les vues sur les tables ci dessus
- om_triggers : cette vue contient les triggers des tables ci dessus
- om_proc : cette vue contient les procédures utilisées par les tables ci dessus

Les répertoires de gen sont rajoutés avec les classes relatives aux vues qui ont été créés par le générateur :
- dans gen/obj : om_tables.class.php, om_champs.class.php, om_contraintes.class.php, om_triggers.class.php, om_proc.class.php, om_vues.class.php
- dans gen/sql/psql : om_tables.inc.php, om_champs.inc.php, om_contraintes.inc.php. om_triggers.inc.php, om_proc.inc.php, om_vues.inc.php

Enfin ces classes sont surchargées dans répertoires obj et sql/pgsql.

Le principe est simple : les classes sont liées aux vues d'information_shéma dans l'affichage.

La mise à jour d'information_schema se fait avec :
- les commandes de création et suppression de table pour om_table,
- les commandes de création, modification et suppression de champs pour om_forms
- la commande de suppression de clé secondaire pour om_contraintes.
- les commandes de création, modification et suppression de vues pour om_vues
- les commandes de création, modification et suppression de trigger pour om_triggers
- les commandes de création, modification et suppression de procédures pour om_proc

---

Note : Les paramètres particuliers des tables pour gen_plus.class.php sont stockés dans la table om_tables_parametres. Les paramètres particuliers des champs pour gen_plus.class.php sont stockés dans la table om_forms Les paramètres particuliers des actions pour gen_plus.class.php sont stockés dans la table om_actions

---

3.2 La surcharge om_gen_plus.class.php :

Il a été surchargé om_gen.class.php par om_gen_plus.class.php dans le repertoire app.

om_gen_plus prend en compte le paramétrage de om_tables_parametre et de om_forms.

Les méthodes suivantes ont été surchargées :
- def_obj_meth_setlib : pour le parémétrage des libellés de champs
- def_obj_meth_settype_by_maj : pour le paramètrage des types de champs
- def_obj_meth_get_var_sql_forminc pour le paramétrage de l'ordre des champs
- def_php_script_header : pour la mise a jour du header avec genplus
- stream_slightly_equals_file : la génération se fait à chaque fois et non quand la base est changée

— table_obj_class_gen : pour ajouter de la méthode setLayout

Il a été ajouter les méthodes suivantes :

— def_obj_meth_setlayout : définition de la méthode setlayout pour regrouper les champs
— def_moteur_recherche_inc : définition du moteur de recherche
— def_obj_meth_init_class_actions() : définition des actions
— edition_action() : méthode d'édition.

CHAPITRE 4

Bibliographie

— http://www.openmairie.org/

CHAPITRE 5

Contributeurs

(par ordre alphabétique)

— François Raynaud