
om5 Documentation

Version 1.2.0a4

om5

oct. 06, 2024

Table des matières

1	Installateur	3
1.1	Pré requis d'installation :	3
1.2	Installateur :	5
2	Développement Rapide	9
2.1	Saisir table et génération :	9
2.2	Saisir des champs dans la table :	13
2.3	Supprimer les contraintes de clé secondaire :	17
2.4	Saisir des actions :	18
2.5	Activer une procédure :	20
2.6	Saisir des triggers :	21
2.7	Saisir les procédures :	23
2.8	Saisir des vues :	27
2.9	Exemple : la gestion de ma bibliothèque :	32
2.10	Paramétrer les états :	42
2.11	Paramétrer les requêtes :	42
2.12	Paramétrer les sous états :	42
2.13	Paramétrer les widgets :	43
3	Technique	45
3.1	Le framework openMairie :	45
3.2	La surcharge om_gen_plus.class.php :	46
4	openstock	49
4.1	Elément d'uml :	49
4.2	les tables :	50
4.3	article	54
4.4	Client :	56
4.5	devis	57
4.6	Devis prestations :	58
4.7	Devis articles :	61
4.8	Entree des articles	65
4.9	Etat de la facture	68
4.10	facture	69
4.11	Famille d'article	72
4.12	Fournisseur	73
4.13	livraison	73

4.14	Mode de règlement du versement	75
4.15	Prestation	76
4.16	Sortie des articles	76
4.17	Sortie des prestations	80
4.18	Versement	83
5	Bibliographie	87
6	Contributeurs	89

Note : Cette création est mise à disposition selon le Contrat Paternité-Partage des Conditions Initiales à l'Identique 2.0 France disponible en ligne <http://creativecommons.org/licenses/by-sa/2.0/fr/> ou par courrier postal à Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.

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 composé 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 technique propose de décrire l'intégration du projet om5_rad dans le framework openMairie.

La partie openStock est une exemple d'application d'om5-no-code à une gestion de stock.

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/>

Ce chapitre 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 packages 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 :

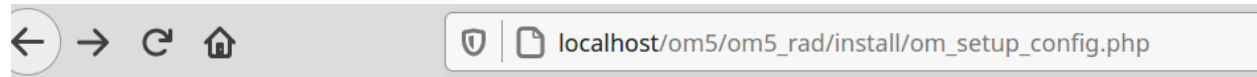
note :

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 ! »



Bienvenue sur l'installateur d'openFramework 4.10.0

Avant de nous lancer, nous avons besoin de certaines informations sur votre base de données. Il va vous falloir réunir les informations suivantes pour continuer.

- Nom de la base de données
- Nom du schéma de données
- Nom d'utilisateur Postgres
- Mot de passe de l'utilisateur
- Hôte de base de données
- Port de la base de données (par défaut 5432)

Nous allons utiliser ces informations pour créer le fichier dans dyn : database.inc.php.

Si pour une raison ou pour une autre la création automatique du fichier ne fonctionne pas, ne vous inquiétez pas. Sa seule action est d'ajouter les informations de la base de données dans un fichier de configuration. Vous pouvez aussi simplement ouvrir dyn/database.inc-sample.php dans un éditeur de texte, y remplir vos informations et l'enregistrer sous le nom de database.inc.php. Vous devriez normalement avoir reçu ces informations de la part de votre hébergeur. Si vous ne les avez pas, il vous faudra contacter votre hébergeur afin de continuer.

Si vous avez besoin d'aide, contactez le [forum](#)

vous avez des explications complémentaires au lien suivant [documentation du projet om5_rad](#)

[Cliquez sur ce lien pour configurer le fichier database.inc.php](#)

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

Vous devez saisir ci-dessous les détails de connexion à votre base de données. Si vous ne les connaissez pas, contactez votre hébergeur

Nom de la base de données	<input type="text" value="pm5_rad"/>	Le nom de la base de données avec laquelle vous souhaitez utiliser le framework
Nom du schéma de la base de données	<input type="text" value="om5"/>	Le nom du schéma avec lequel vous souhaitez utiliser le framework
Identifiant	<input type="text" value="postgres"/>	Nom d'utilisateur postgresql
Mot de passe	<input type="text" value="postgres"/>	Votre mot de passe de base de données
Adresse de la base de données	<input type="text" value="localhost"/>	Si localhost ne fonctionne pas, demandez cette information à l'hébergeur de votre site.
Port du serveur de base de données	<input type="text" value="5432"/>	Si 5432 ne fonctionne pas, demandez cette information à l'hébergeur de votre site.

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 :

la connexion est ok
 Vous avez accès en écriture et le fichier a été copié en repertoire dyn avec les données suivantes

```
<?php
$conn[1] = array(
    'om5_rad', // Titre
    'pgsql', // Type de base
    'pgsql', // Type de base
    'postgres',
    'postgres', // Mot de passe
    '', // Protocole de connexion
    'localhost', // Nom d hote
    '5432', // Port du serveur
    '', // Socket
    'om5_rad', // Nom de la base
    'AAAA-MM-JJ', // Format de la date
    'om5', // Nom du schéma
    '', // Préfixe
    null // Paramétrage pour l'annuaire LDAP
);
```

om5_rad	om5	postgres	postgres	localhost	5432
---------	-----	----------	----------	-----------	------

Créer le schéma des données pour le framework

vous pouvez aussi utiliser la procédure data/nosql/install.sql

1.2.2 Installation du schéma du framework :

Les scripts nécessaires au framawork sont lancés :

```
SET client_encoding = 'UTF8'; executé
SET search_path = om5, public, pg_catalog; executé
CREATE EXTENSION IF NOT EXISTS postgis; executé
DROP SCHEMA IF EXISTS om5 CASCADE; executé
CREATE SCHEMA om5 executé
var/www/html/om5/om5_rad/core/data/pgsql/init.sql
279 lignes traitées
var/www/html/om5/om5_rad/core/data/pgsql/init_permissions.sql
121 lignes traitées
var/www/html/om5/om5_rad/core/data/pgsql/init_parametrage.sql
3 lignes traitées
Ajout des vues et des droits du projet om5_rad
CREATE OR REPLACE VIEW om5.om_tables AS SELECT table_name, table_schema, table_type FROM information_schema.tables where table_type = 'BASE TABLE' and table_schema = 'om5' executé
CREATE OR REPLACE VIEW om5.om_champs AS SELECT concat(table_name, '.', column_name) as column_name, table_name, table_schema, data_type, is_nullable, character_maximum_length FROM
information_schema.columns where table_schema = 'om5' executé
CREATE OR REPLACE VIEW om_constraints AS SELECT tc.constraint_name, tc.table_name, kcu.column_name, ccu.table_name AS foreign_table_name, ccu.column_name AS foreign_column_name FROM
information_schema.table_constraints AS tc JOIN information_schema.key_column_usage AS kcu USING (constraint_schema, constraint_name) JOIN information_schema.constraint_column_usage AS ccu
USING (constraint_schema, constraint_name) WHERE constraint_type = 'FOREIGN KEY' AND tc.table_schema = 'om5' executé
INSERT INTO om5.om_droit (om_droit, libelle, om_profil) VALUES (nextval('om5.om_droit_seq'), 'om_tables', 1) executé
INSERT INTO om5.om_droit (om_droit, libelle, om_profil) VALUES (nextval('om5.om_droit_seq'), 'om_constraints', 1) executé
terminé
```

[Cliquez sur ce lien pour lancer le framework](#)

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

Ce chapitre 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.

Enfin, les modules de gestion d'édition (état, sous-état, requête) et le module de gestion des widgets du tableau de bord, ont été adaptés à la gestion des actions d'om5-no-code.

2.1 Saisir table et génération :

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets

Ce listing décrit les tables existantes pour votre application

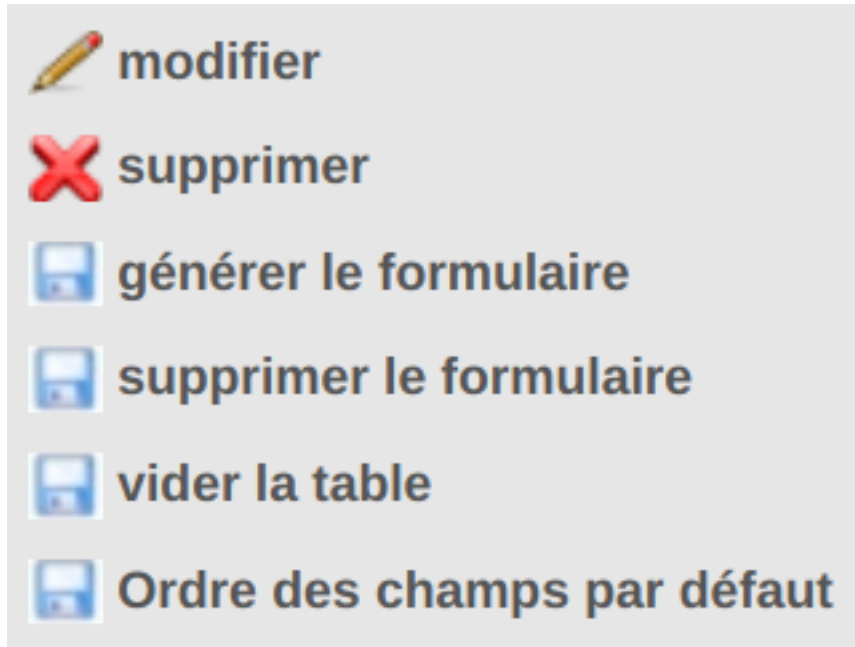
+	table	libelle	col1	col2	col3	menu	nb_col	recherche	order_by	asc_desc
	article	article	Article	Description	colonne 3	parametre	2	simple		
	client	client	Adresse	Cumul	colonne 3	application	2	avancée		
	devis	devis	colonne 1	colonne 2	colonne 3	application	1	avancée	devis.date_devis	desc
	devis_article	devis_article	article	colonne 2	colonne 3	sans	1	simple		
	devis_prestation	devis_prestation	prestation	colonne 2	colonne 3	sans	1	simple		
	entree_article	entree_article	Libellé	montant	colonne 3	sans	2	simple		
	etat	etat	Libellé	colonne 2	colonne 3	parametre	1	simple		
	facture	facture	données	cumuls	colonne 3	application	2	avancée	facture.date_facture	desc
	famille	famille	Libellé	colonne 2	colonne 3	parametre	1	simple		
	fournisseur	fournisseur	colonne 1	colonne 2	colonne 3	application	1	simple		

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table	Formulaire	Clé Secondaire	Action	Procédures	Triggers	Vues																		
<div style="display: flex; justify-content: space-between; align-items: center;"> Modifier Retour </div> <table border="1" style="width: 100%;"> <tr> <td>Nom de la table *</td> <td>facture</td> </tr> <tr> <td>libellé de la table</td> <td>facture</td> </tr> <tr> <td>affichage en menu</td> <td>afficher en menu application</td> </tr> <tr> <td>nombre de colonnes</td> <td>2 colonnes</td> </tr> <tr> <td>libellé colonne 1</td> <td>données</td> </tr> <tr> <td>libellé colonne 2</td> <td>cumuls</td> </tr> <tr> <td>moteur de recherche</td> <td>avancée</td> </tr> <tr> <td>ordre dans la liste</td> <td>facture.date_facture</td> </tr> <tr> <td>asc_desc</td> <td>descendant</td> </tr> </table> <div style="display: flex; justify-content: space-between; align-items: center; margin-top: 10px;"> Modifier Retour </div>							Nom de la table *	facture	libellé de la table	facture	affichage en menu	afficher en menu application	nombre de colonnes	2 colonnes	libellé colonne 1	données	libellé colonne 2	cumuls	moteur de recherche	avancée	ordre dans la liste	facture.date_facture	asc_desc	descendant
Nom de la table *	facture																							
libellé de la table	facture																							
affichage en menu	afficher en menu application																							
nombre de colonnes	2 colonnes																							
libellé colonne 1	données																							
libellé colonne 2	cumuls																							
moteur de recherche	avancée																							
ordre dans la liste	facture.date_facture																							
asc_desc	descendant																							

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
- Pour les listes, il est possible de proposer un moteur de recherche avancée (recherche simple par défaut)

Lorsque la table est créée :

- la clé primaire est créé automatiquement, elle est de type integer not null et elle porte le nom de la table
- un droit est créé sur la table dans om_droit avec le profil utilisateur
- la table est insérée dans le menu option « application » ou « paramètre métier » si l'affichage est demandé 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
- Pour les listes, il est possible de proposer moteur de recherche avancée, un champ de tri de la table et l'ordre de tri (ascendant ou descendant)

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 ainsi que les triggers et les procédures
- ensuite la séquence est détruite
- puis les paramètres d'om_tables_paramètres et om_forms sont détruits

— le ou les droits sur la table sont détruits

Note : Il n'est pas possible de supprimer une table existante dans une vue. Il faut détruire toutes les vues où la table est utilisée (et pas seulement celles existantes dans le sous formulaire). [contrôle postgresql]

2.1.2 Action générer le formulaire :

- la liste et le formulaire sont générées dans le répertoire gen ;
- la séquence est créée (nom de la table avec le suffixe « _seq ») ;
- les paramètres des champs (om_forms) et de la table (om_tables_parametre) sont pris en compte.
- il est pris en compte les actions du sous formulaire om_actions

Le message suivant est affiché :

The screenshot shows the Om5 No Code interface for the 'Emprunteur Om5' table. The breadcrumb path is 'Om5 No Code > Gestion De Tables Et Génération Des Objets > Emprunteur Om5'. The 'Table' tab is selected. A yellow message box displays the following information:

- Table : emprunteur
- Génération de ../gen/sql/pgsql/emprunteur.inc.php
- Génération de ../gen/obj/emprunteur.class.php

Below the message box is a 'Retour' button with a blue arrow icon. A table below shows the table details:

Nom de la table	emprunteur
Nom du schema	om5
Type de table	BASE TABLE
libellé de la table	Mes Amis

On the right side, there are three action buttons: 'modifier' (pencil icon), 'supprimer' (red X icon), and 'générer le formulaire' (blue document icon).

2.1.3 Action supprimer le formulaire :

la liste et le formulaire sont supprimés

Note : Il faut avant supprimer manuellement les actions. [contrôle de clé secondaire openMairie] Il est utile de supprimer les éditions générées (om_requete, om_etat, om_sousetat) et les widgets (om_widget et om_dashboard) dans le menu paramètre et administration d'openMairie. [non bloquant]

2.1.4 Action vider la table :

Cette action vide la table des enregistrements.

Note : Cette action est obligatoire avant de supprimer la table. [contrôle om5-no-code]

2.1.5 Action ordre des champs par défaut

Cette action met les champs dans l'ordre de saisie sur une colonne pour le formulaire et pour la liste.

Le message suivant s'affiche :

i remise en ordre des champs par défaut
formulaire

- prestation.libelle : position -> 1 | bloc -> C1
- prestation.prix : position -> 2 | bloc -> C1

liste

- prestation.libelle : position -> 1
- prestation.prix : position -> 2

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 avec le sous formulaire « formulaire » :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 4 enregistrement(s) sur 4

+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	pret.date_de_pret	pret	date		Oui	date de prêt		C1	3		1	3	date de prêt
	pret.emprunteur	pret	integer	pret_emprunteur_fkey	Oui	emprunteur		C1	1		1	1	emprunteur
	pret.livre	pret	integer	pret_livre_fkey	Oui	livre		C1	2		1	2	livre
	pret.retour	pret	boolean		Non	retour		C1	4		1	4	retour

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Article Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

om5 no code ➔ Formulaire

Ajouter Retour

Type de champs PGSQL

Nom du champs

Nom de la table

Obligatoire

Champs Form ajouter

Champs Form modifier

Champs Form consulter

Champs Form supprimer

Affichage en liste

Ajouter Retour

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

En choisissant le data-type "clé secondaire", 3 nouveaux champs apparaissent : table, vue select et vue select by id et le champs nom disparaît.

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.

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

Table Formulaire Action clé secondaire Procédures Triggers Vues

om5 no code > Formulaire

Ajouter Retour

Type de champs PGSQL clé secondaire

Nom du champs choisir une table

Nom de la table article

Obligatoire Non obligatoire

Champs Form ajouter visible

Champs Form modifier visible

Champs Form consulter visible

Champs Form supprimer visible

Vue du select choisir une vue

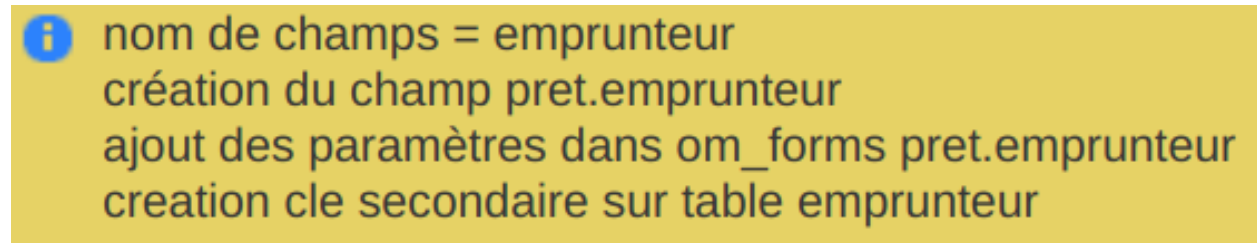
Vue du select by id choisir une vue

Affichage en liste Oui

Ajouter Retour

Il est possible de modifier le select d'affichage dans le formulaire en associant la clé secondaire avec une vue d'affichage en mise à jour (select) et en consultation (select by id). (voir om_vues) Attention, la vue ne doit pas avoir une condition restrictive excluant la relation (surtout en consultation).

le message suivant s'affiche en validation :



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.

Les règles de génération de la requête du sous formulaire (om_requête) sont les suivantes :

- le champ «numéric» est arrondi à deux chiffres après la virgule
- le champ « date » est au format français
- le champ « boolean » est « oui » si il est vrai, « non » sinon.
- le champ « clé secondaire » (integer + foreign key) est remplacé par le libellé si il existe ou par le 2ème champs ou en dernier lieu par la clé scondaire
- la clé primaire n'est pas affichée
- les champs doivent être affichés en liste et sont classés dans l'ordre de la liste (position)

Les règles de génération du sous état (om_sousetat) sont les suivantes :

- le nombre de colonne est le nombre de champs de la requête
- chaque champs occupe la même longueur de colonne.
- l'entête de colonne est celle du nom de champs

Note : Le sous-état généré est non actif. Il faut aller sur le menu « paramètre » option « sous état » et paramétrer le sous état en mode actif. Il faut insérer le sous état dans le corps de l'état correspondant. Il n'est pas besoin de générer la table dans om_tables

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.

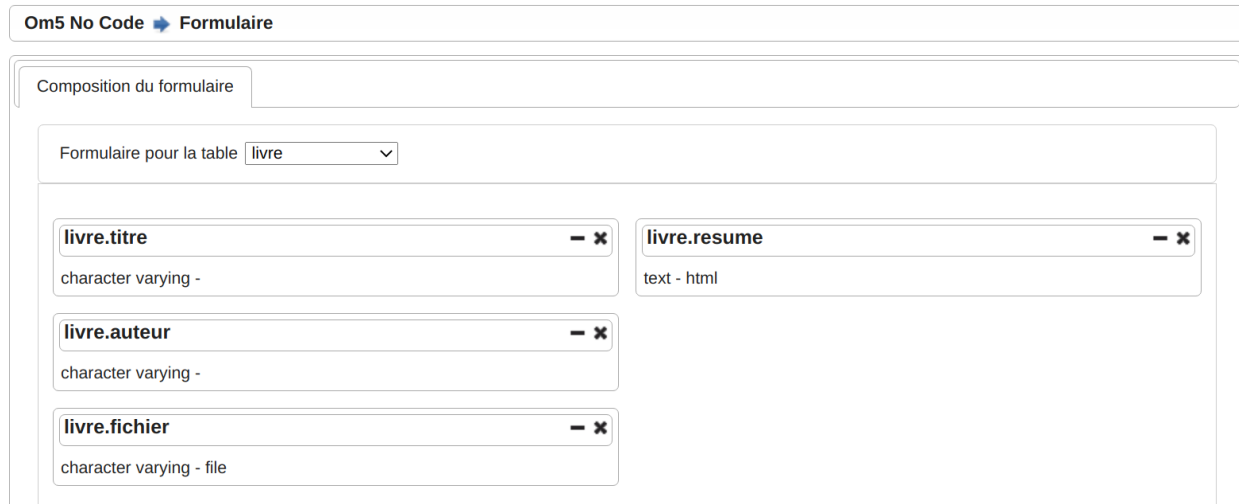
Il est possible de modifier les positions et colonnes dans les formulaires avec l'option du menu om5-no-code : composition formulaires

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 formulaire, la suppression d'un champs, rend le champs non visible 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.

Om5 No Code ➔ Liste

Composition de la liste

Formulaire pour la liste emprunteur ▼

emprunteur.nom	- x
character varying -	
emprunteur.prenom	- x
character varying -	
emprunteur.date_de_naissance	- x
date -	
emprunteur.actif	- x
boolean -	

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

Note : Pour intégrer les ajouts, suppression et modification de champs dans le formulaire, il faut re générer le formulaire.

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 :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 2 enregistrement(s) sur 2

	▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ
	pret_emprunteur_fkey	om5	pret	emprunteur	emprunteur	emprunteur
	pret_livre_fkey	om5	pret	livre	livre	livre

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger pgsq|

administration ➔ om_contraintes ➔ pret_emprunteur_fkey pret

[Retour](#)

Contrainte	pret_emprunteur_fkey	✖ supprimer
Table	pret	
Champs	emprunteur	
Foreign table	emprunteur	
Foreign champ	emprunteur	

[Retour](#)

2.4 Saisir des actions :

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table Formulaire Action clé secondaire Trigger pgsq|

1 - 1 enregistrement(s) sur 1

+	om_actions	libellé	module	ordre	titre	table_name
		1 livre	edition		50 Livre en prêt	Ma bibliothèque

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table Formulaire Action clé secondaire Trigger pgsq|

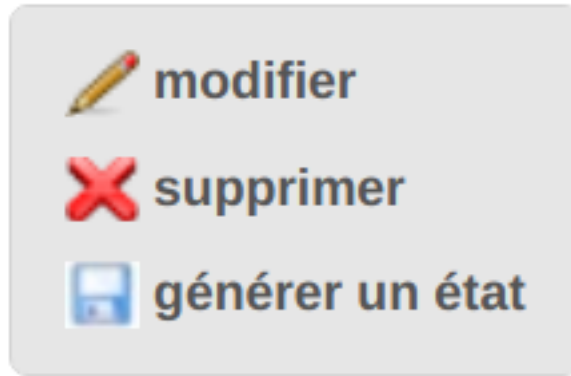
administration ➔ om_actions

Ajouter [Retour](#)

libellé *	<input type="text" value="livre"/>
module *	<input type="text" value="edition"/>
titre	<input type="text" value="Livre en prêt"/>
ordre	<input type="text" value="50"/>
table_name	Ma bibliothèque

Ajouter [Retour](#)

Les actions d'om_action sont les suivantes :



L'action « générer 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.

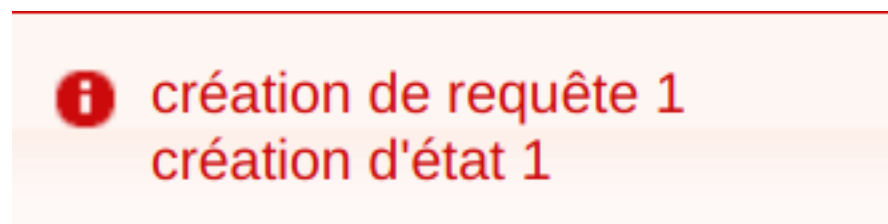
Les règles de génération de la requête (om_requête) sont les suivantes :

- le champ « numéric » est arrondi à deux chiffres après la virgule
- le champ « date » est au format français
- le champ « boolean » est « oui » si il est vrai, « non » sinon.
- le champ « clé secondaire » (integer + foreign key) est remplacé par le libellé si il existe ou par le 2ème champs ou en dernier lieu par la clé scondaire

Les règles de génération de l'état (om_etat) sont les suivantes :

- tous champs sont affichés dans le corps sauf la clé primaire sous forme de tableau avec en première colonne le nom du champ et dans une deuxième colonne, la valeur du champ.
- les champs sont classés dans l'ordre du formulaire (colonne - position)
- dans la zone titre, il est affiché la date du jour.

Suite à la génération, 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 oublier 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 Activer une procédure :

Une action peut activer une fonction existante comme ci dessous (facture\$3) :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

administration ➔ om_actions ➔ 3 changement_etat

Modifier Retour

om_actions * 3

libellé * changement_etat

module * facture\$3

titre Suivant

ordre 100

table_name facture

Modifier Retour

Note : Pour intégrer les actions dans le formulaire, il faut re générer le formulaire.

2.6 Saisir des triggers :

Un déclencheur dans PostgreSQL se compose de deux parties :

- une fonction de déclenchement (voir om_proc)
- le déclencheur réel, qui appelle la fonction de déclenchement

Cette architecture présente l'avantage que les fonctions de déclenchement peuvent être réutilisées : les déclencheurs sur différentes tables peuvent utiliser la même fonction de déclenchement.

Un trigger fait référence à une fonction qui est déclenchée automatiquement lorsqu'un événement de base de données se produit sur un objet de base de données

La procédure doit exister avant la création du trigger.

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Essai Om5

Table Formulaire Action clé secondaire Trigger pgsq

1 - 0 enregistrement(s) sur 0

+	nom	evenement	table	ordre	condition	état	orientation	timing
Aucun enregistrement.								

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)


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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5


Table Formulaire Action clé secondaire Procédures Triggers Vues

administration ➔ om_triggers

```
select proname FROM om5.om_proc WHERE SPLIT_PART(om_proc.proname, '_', 1)= 'pret'
```

Ajouter  Retour

nom declencheur *	pret_4
sur evenement *	<input type="text" value="choisir un evenement"/>
sur la table	pret
Si (condition)	
executer la procédure *	<input type="text" value="choisir une procedure"/>
sur	<input type="text" value="enregistrement courant"/>
quand ?	<input type="text" value="avant enregistrement"/>

Ajouter  Retour

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

Les options de l'assistant trigger sont les suivantes :

2.6.1 nom du trigger

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

2.6.2 Evénement :

Le trigger s'exécute lors de :

- la création d'enregistrement (insert)
- la modification d'un enregistrement (update)
- la destruction d'un enregistrement (delete)

2.6.3 table

Par défaut la table du formulaire

Non modifiable

2.6.4 ordre

ordre d'execution des trigers : non implémenté

2.6.5 condition

note :

La condition d'exécution du trigger n'est pas implémenté dans la version actuelle

La condition WHEN est testée pour voir si le déclencheur doit être déclenché. Dans les déclencheurs au niveau de la ligne, la condition WHEN peut examiner les anciennes et/ou les nouvelles valeurs des colonnes de la ligne.

2.6.6 Procédure

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

2.6.7 sur ?

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

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

2.6.8 quand ?

avant enregistrement : insert, update

après enregistrement : insert, update ou delete

L'option INSTEAD OF n'est pas implémentée.

il doit y avoir un return new ou old dans la procédure :

- NEW est NULL dans ON DELETE triggers
- OLD est NULL dans ON INSERT triggers

2.7 Saisir les procédures :

Note : Les procédures sont liés aux triggers postgresql ou aux action de la table.

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 3 enregistrement(s) sur 3

	nom	declare	begin
	pret_1	DECLARE nb_pret integer;	nb_pret:=(select count(*) from pret where emprunteur = NEW.emprunteur); update emprunteur set nombre_pret=nb_pret where emprunteur = NEW.emprunteur; RETURN NEW; END;
	pret_3	DECLARE nb TEXT;	update emprunteur set nombre_pret=coalesce(nombre_pret,0)-1 where emprunteur = OLD.emprunteur; RETURN OLD; END;
	pret_2	DECLARE nb TEXT;	update om5.emprunteur set nombre_pret=coalesce(nombre_pret,0)+1 where emprunteur = NEW.emprunteur; RETURN NEW; END;

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

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

Table Formulaire Action clé secondaire Procédures Triggers Vues

om5 no code ➔ gestion de procédure stockée postgresql ➔ pret_1

Modifier Retour

* pret_1

Déclaration de variable

Procédure

assistant

choisir un champ NEW ▼ ➔

choisir un champ OLD ▼ ➔

choisir un champ autre ▼ ➔

Procédure

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

Retour ▼

{search_path=om5, public, pg_catalog}

Modifier Retour

2.7.1 Retour

Le retour permet de préciser la catégorie de la procédure :

- procédure trigger : retour de l'enregistrement NEW ou OLD
- procédures action : retour du message d'exécution de l'action dans la procédure

2.7.2 nom du procedure

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

2.7.3 table

Par défaut la table du formulaire Non modifiable

2.7.4 Déclaration de variable

Champs optionnel utile si il est utilisé des variables internes :

L'assistant propose 10 variables v_1 à v_9 et la variable message (voir procédure action) et les types de variables

Déclaration de variable

assistant

variable ▾ Type ▾ ➔

Déclaration

```
v_1 INTEGER;
message TEXT;
v_2 INTEGER;
```

2.7.5 Procédure :

Pour construire la procédure (begin), l'assistant propose :

- l'association NEW et OLD au nom de champs de la table courante
- l'association condition ou agrégation + nom de champs des tables + opérateur de comparaison
- l'association requête (update, select ...), nom de champ, nom de table
- les expressions IF THEN, RAISE EXCEPTION ...

Procédure

assistant

NEW ▾	champs sortie_article ▾	comparaison ▾	➔
Condition/Agrégation ▾	table.champs ▾	comparaison ▾	➔
Requete ▾	Tables ▾	champs ▾	➔
Expression ▾	➔		

Procédure

```
v_1=(select COALESCE(sum(montant),0) from sortie_prestation where facture = OLD.facture);
v_2=(select COALESCE(sum(montant),0) from sortie_article where facture = OLD.facture);
v_3=v_1+v_2;
update facture set montant=v_3 where facture = OLD.facture;
```

Les erreurs exception sont remontées en erreur dans l'interface.

```
IF (NEW.quantite <= 0) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas sortir une quantité négative ou égale a_
↪0';
ELSE
...

```

2.7.6 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;
```

2.7.7 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.

2.7.8 Fonction associée à une action :

La valeur de la clé primaire courante est envoyé par l'action dans la variable idx

Le retour du message s'affiche dans l'interface du formulaire. Il faut déclarer la variable message comme text

```
DECLARE
v_1 INTEGER;
message TEXT;
v_2 INTEGER;

BEGIN
-- etat courant
v_1=(select etat from facture where facture = idx);
-- au dela de etat=4 retour etat en cours
IF (v_1=4) THEN
```

(suite sur la page suivante)

(suite de la page précédente)

```

        v_2=1;
        message='Mise à jour état effectuée pour état origine';
ELSE
        v_2=v_1+1;
        message='Mise à jour état effectuée pour état suivant';
END IF;

-- requete
update facture set etat=v_2 where facture = idx;

RETURN message;
END;
```

2.8 Saisir des vues :

Les vues permettent :

- de créer des widgets dans le tableau de bord
- de créer un affichage particulier pour les clés secondaires dans le select d’affichage (le premier champs est la clé, le deuxième champ est l’affichage dans le select du formulaire). Voir om_forms/clé secondaire

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Emprunteur Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 1 enregistrement(s) sur 1

+	▶ table_name	▶ table_schema	▶ view_definition
	emprunteur_1	om5	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.prenom)::text = 'jean'::text) AND (emprunteur.age < 100)) ORDER BY emprunteur.nom;

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Emprunteur Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

om5 no code ➔ gestion des vues postgresql ➔ emprunteur_1 om5

Modifier Retour

```
table_name * emprunteur_1
table_schema om5
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.prenom = 'jean' AND emprunteur.age < 100
where
order_by    emprunteur.nom
```

Modifier Retour

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

2.8.1 nom de la vue

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

2.8.2 table

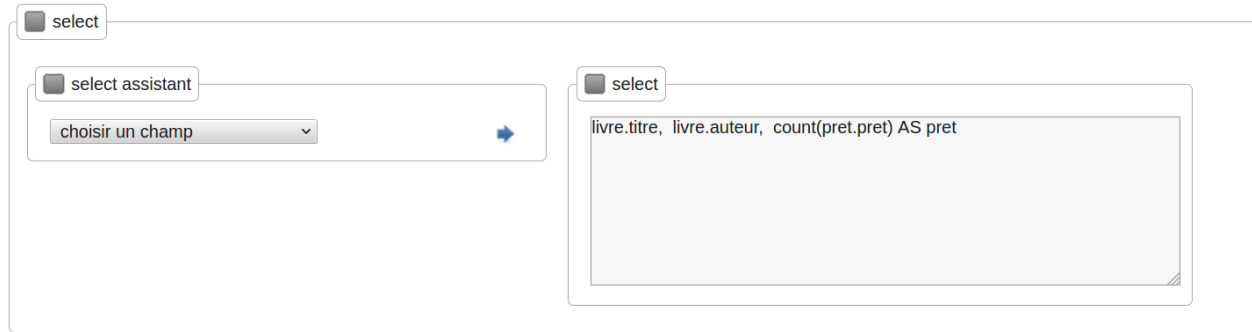
Par défaut la table du formulaire Non modifiable

2.8.3 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 :

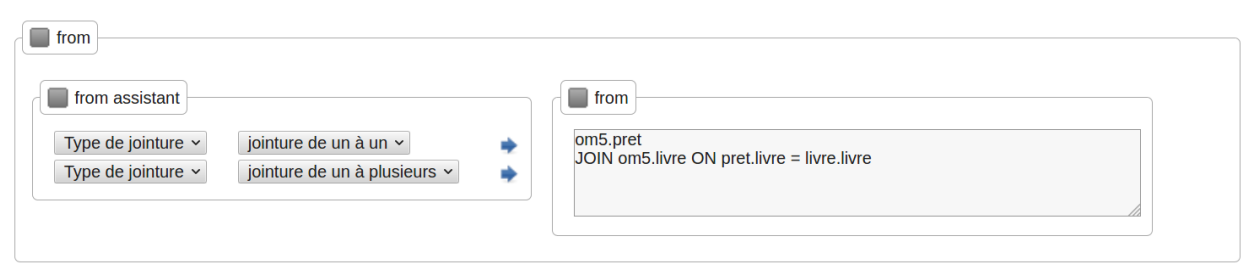


2.8.4 assistant from

Tables de la vue, par défaut la table courante.

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

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
- évaluation est une jointure de un à un sur livre

2.8.5 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 :

group_by

group_by assistant

choisir un champ

group_by

livre.livre, livre.titre, livre.auteur

2.8.6 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 :

having

having assistant

choisir un champ

having

emprunteur.nom='Dupont'

where

where assistant

choisir un champ

where

2.8.7 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 :

order_by

order_by assistant

choisir un champ

order_by

livre.auteur,

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

2.8.8 action « créer un widget » :

Il est possible de créer un widget dans le tableau de bord si celui-ci n'existe pas

Le widget est créé dans administration -> widget :

Administration ➔ Tableaux De Bord ➔ Widget

widget

1 - 2 enregistrement(s) sur 2

Tous Recherche

+	widget ↗	libellé ↗	type ↗
		9 vue emprunteur_1	file
		8 vue livre_1	file

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

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

Administration ➔ Tableaux De Bord ➔ Widget ➔ 9 Vue Emprunteur_1

widget | tableau de bord

Modifier  Retour

widget * 9

libellé * vue emprunteur_1

type * file - le contenu du widget provient d'un script sur le serveur

script * vue

arguments

emprunteur_1

Modifier  Retour

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

Tableau De Bord																				
<table border="1"> <thead> <tr> <th colspan="3">vue livre_1</th> </tr> </thead> <tbody> <tr> <td>titre</td> <td>auteur</td> <td>pret</td> </tr> <tr> <td>une vie</td> <td>Maupassant</td> <td>2</td> </tr> <tr> <td>Pensées</td> <td>Pascal</td> <td>7</td> </tr> <tr> <td colspan="3">2 enregistrement(s)</td> </tr> </tbody> </table>			vue livre_1			titre	auteur	pret	une vie	Maupassant	2	Pensées	Pascal	7	2 enregistrement(s)					
vue livre_1																				
titre	auteur	pret																		
une vie	Maupassant	2																		
Pensées	Pascal	7																		
2 enregistrement(s)																				
<table border="1"> <thead> <tr> <th colspan="3">vue emprunteur_1</th> </tr> </thead> <tbody> <tr> <td>nom</td> <td>prenom</td> <td>pret</td> </tr> <tr> <td>DUPONT</td> <td>Jean</td> <td>3</td> </tr> <tr> <td>DURANT</td> <td>Pierre</td> <td>5</td> </tr> <tr> <td>GILBERT</td> <td>Jean</td> <td>1</td> </tr> <tr> <td colspan="3">3 enregistrement(s)</td> </tr> </tbody> </table>			vue emprunteur_1			nom	prenom	pret	DUPONT	Jean	3	DURANT	Pierre	5	GILBERT	Jean	1	3 enregistrement(s)		
vue emprunteur_1																				
nom	prenom	pret																		
DUPONT	Jean	3																		
DURANT	Pierre	5																		
GILBERT	Jean	1																		
3 enregistrement(s)																				
<p>Framework openMairie Version 4.10.0-om5_no_code-1.2.0a1 Documentation Forum openMairie.org</p>																				

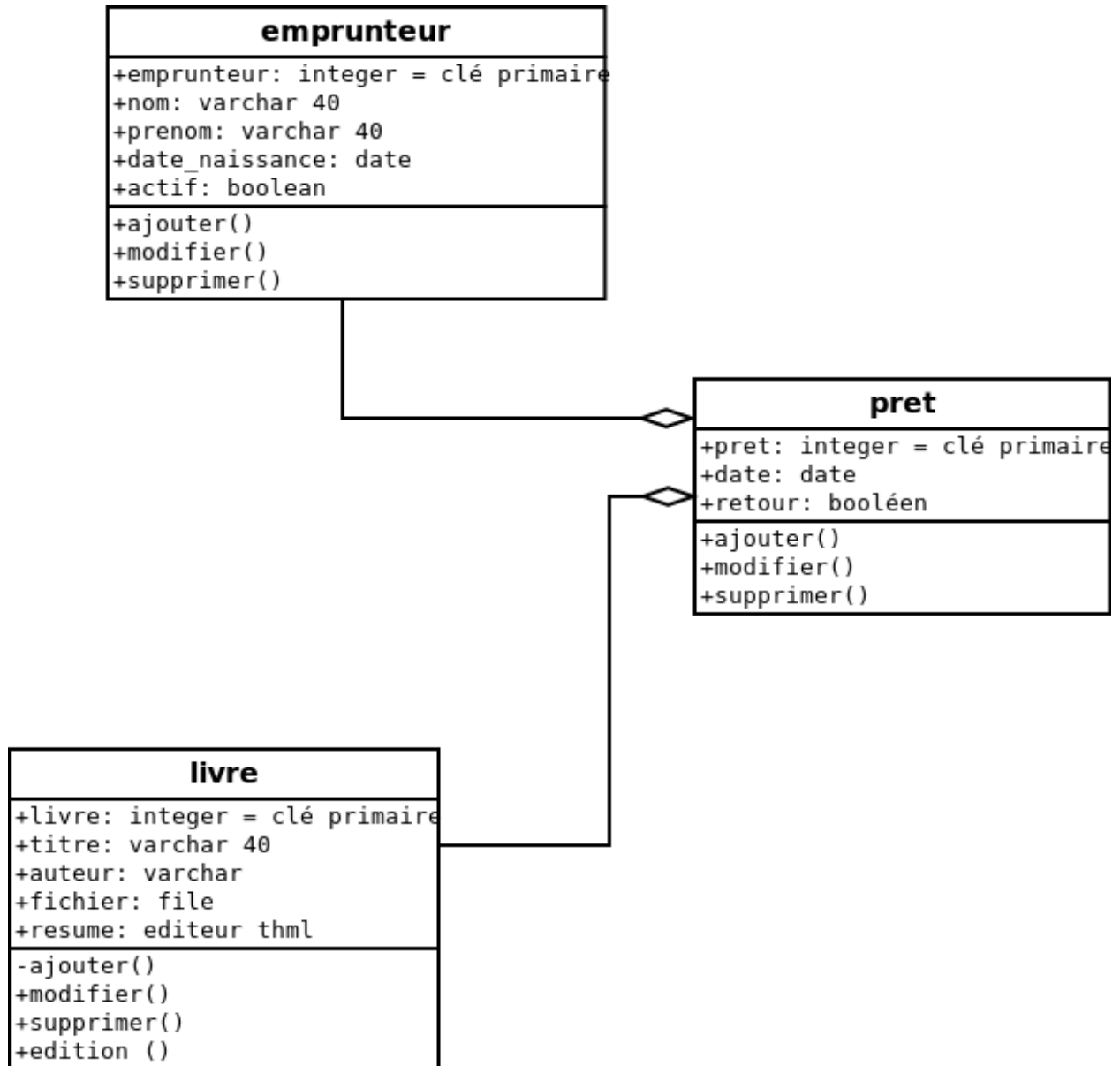
2.9 Exemple : la gestion de ma bibliothèque :

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 crée les tables avec om_tables en modifiant éventuellement les paramètres d'om_tables_parametre
- on crée les champs correspondants avec om_champs en modifiant éventuellement les paramètres d'om_forms
- on compose le formulaire avec l'option composeur du menu om5-no-code
- on génère les formulaires pour chaque table.
- on peut générer une action « générer un état » (sous formulaire 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

2.9.1 Diagramme de classe de la gestion de bibliothèque :



2.9.2 La création des tables :

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets

Ce listing décrit les tables existantes pour votre application

Table										
1 - 15 enregistrement(s) sur 16			Page 1 / 2		Tous		Recherche			
+	table	libelle	col1	col2	col3	menu	nb_col	recherche	order_by	asc_desc
	article	article	Article	Description	colonne 3	parametre	2	simple		
	client	client	Adresse	Cumul	colonne 3	application	2	avancée		
	devis	devis	colonne 1	colonne 2	colonne 3	application	1	avancée	devis.date_devis	desc
	devis_article	devis_article	article	colonne 2	colonne 3	sans	1	simple		
	devis_prestation	devis_prestation	prestation	colonne 2	colonne 3	sans	1	simple		
	entree_article	entree_article	Libellé	montant	colonne 3	sans	2	simple		
	etat	etat	Libellé	colonne 2	colonne 3	parametre	1	simple		
	facture	facture	données	cumuls	colonne 3	application	2	avancée	facture.date_facture	desc
	famille	famille	Libellé	colonne 2	colonne 3	parametre	1	simple		
	fournisseur	fournisseur	colonne 1	colonne 2	colonne 3	application	1	simple		

2.9.3 Les champs suivants ont été créés pour chaque table :

livre

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livre Om5

Table													
Formulaire		Action		clé secondaire		Trigger pgsq							
1 - 4 enregistrement(s) sur 4													
+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	livre.titre	livre	character varying		Oui	titre		C1	1		1	1	titre
	livre.auteur	livre	character varying		Oui	auteur		C1	2		1	2	auteur
	livre.fichier	livre	character varying		Non	fichier	file	C1	3		1	3	fichier
	livre.resume	livre	text		Non		html	C2	1		1	4	resumé

emprunteur

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Emprunteur Om5

Table													
Formulaire		Action		clé secondaire		Trigger pgsq							
1 - 4 enregistrement(s) sur 4													
+	champ	table	type	clé secondaire	obligatoire	libelle	type	bloc	no	calcul	liste	no_l	lib_l
	emprunteur.actif	emprunteur	boolean		Non	actif		C2	1		1	4	actif
	emprunteur.date_de_naissance	emprunteur	date		Non	date de naissance		C1	3		1	3	date de naissance
	emprunteur.nom	emprunteur	character varying		Oui	nom		C1	1		1	1	nom
	emprunteur.prenom	emprunteur	character varying		Oui	prénom		C1	2		1	2	prénom

pret

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger postgresql

1 - 4 enregistrement(s) sur 4

+	▶ champ	▶ table	▶ type	▶ clé secondaire	▶ obligatoire	▶ libelle	▶ type	▶ bloc	▶ no	▶ calcul	▶ liste	▶ no_l	▶ lib_l
	pret.emprunteur	pret	integer	pret_emprunteur_fkey	Oui	emprunteur		C1	1		1	1	emprunteur
	pret.livre	pret	integer	pret_livre_fkey	Oui	livre		C1	2		1	2	livre
	pret.date_de_pret	pret	date		Oui	date de prêt		C1	3		1	3	date de prêt
	pret.retour	pret	boolean		Non	retour		C1	4		1	4	retour

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

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Pret Om5

Table Formulaire Action clé secondaire Trigger postgresql

1 - 2 enregistrement(s) sur 2

▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ
pret_emprunteur_fkey	om5	pret	emprunteur	emprunteur	emprunteur
pret_livre_fkey	om5	pret	livre	livre	livre

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

2.9.4 Les formulaires ont été composés de la manière suivante :

Pour la table emprunteur :

Om5 No Code ➔ Formulaire

Composition du formulaire

Formulaire pour la table

emprunteur.nom — x	emprunteur.aktif — x
character varying -	boolean -
emprunteur.prenom — x	
character varying -	
emprunteur.date_de_naissance — x	
date -	

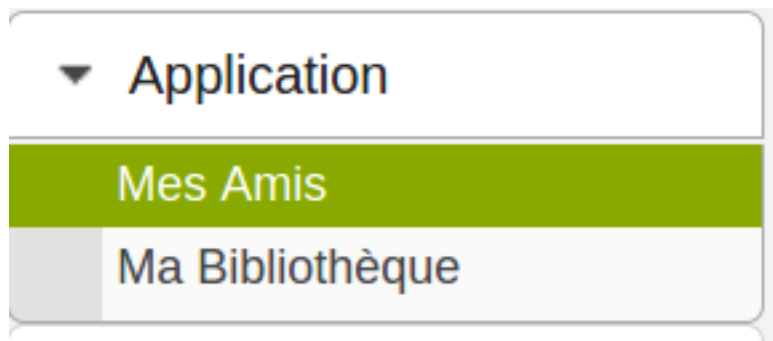
pour la table livre :

The screenshot shows the 'Formulaire' (Form) configuration window in Om5 No Code. At the top, it says 'Om5 No Code ➔ Formulaire'. Below that, a tab labeled 'Composition du formulaire' is active. A dropdown menu shows 'Formulaire pour la table' with 'livre' selected. The main area contains five form fields, each with a title, a description, and a delete icon (a minus sign and an 'x').

Field Name	Description
livre.titre	character varying -
livre.resume	text - html
livre.auteur	character varying -
livre.fichier	character varying - file

2.9.5 Le menu application :

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



2.9.6 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 »

Application ➔ Emprunteur


emprunteur



Afficher la recherche simple

nom prénom date de naissance de à actif

Utilisation de * pour zones de saisie: A*D peut correspondre à 'ABCD'. Par défaut * est ajouté au debut et à la fin des recherches.

Recherche


1 - 2 enregistrement(s) sur 2 

	▶ emprunteur	▶ nom	▶ prénom	▶ date de naissance	▶ actif
	2	DUPONT	Jean	02/05/2003	Oui
	1	DURANT	Pierre	01/05/2007	Oui

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

Application ➔ Emprunteur ➔ 2 DUPONT

emprunteur

 Retour

2

Etat Civil


nom DUPONT


prénom Jean


date de naissance 02/05/2003

Actif

actif Oui

 modifier

 supprimer



 Retour

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

Application ➔ Livre

livre

1 - 2 enregistrement(s) sur 2 Recherche

	▶ livre	▶ titre	▶ auteur	▶ fichier
	1	Les pensées	PASCAL	
	2	Une vie	Maupassant	

Framework openMairie Version 4.10.0-om5_RAD-1.1.0 | [Documentation d'om5_RAD](#) | [Forum](#) | [openMairie.org](#)

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

Application ➔ Livre ➔ 1 Pensées

livre pret

[Retour](#)

1

Titre
titre Pensées
auteur Pascal
fichier json_type_om.txt [Visualiser](#) [Télécharger](#)

[modifier](#)
[supprimer](#)
[Livres en prêt](#)

Résumé
Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

[Retour](#)

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

Application ➔ Livre ➔ 1 Pensées

livre pret

1 - 1 enregistrement(s) sur 1

	pret	emprunteur	livre	date de prêt	retour
	1	DUPONT	Pensées	09/05/2024	Non

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

Application ➔ Livre ➔ 1 Pensées

livre pret

application ➔ pret

[Ajouter](#) [Retour](#)

livres ou amis

emprunteur * DUPONT

livre * Pensées

date de prêt * 09/05/2024

Retour

[Ajouter](#) [Retour](#)

2.9.7 La génération des éditions :

Voir le chapitre précédent : « saisie des actions »

Application ➔ Livre ➔ 1 Pensées

livre pret

Retour

1

Titre
titre Pensées
auteur Pascal
fichier json_type_om.txt Visualiser ➔ Télécharger

modifier
supprimer
Livres en prêt

Résumé
Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

Retour

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



le 14/05/2024

Titre :

Pensées

Auteur :

Pascal

Résumé :

Les **Pensées de Pascal** sont un des rares textes du XVIIe siècle dont nous possédions les manuscrits originaux. Cela tient au fait que l'œuvre est demeurée inachevée en raison de la mort prématurée de son auteur. Dans les années qui ont suivi la polémique des *Provinciales* et le concours sur la roulette,

2.9.8 Les procédures

Exemple de procédures créées avec les formulaires om5

Calcul d'âge

```
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;

```

2.9.9 Triggers pour calcul d'âge

Exemple de triggers créés 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();

```

2.9.10 Vues pour tableau de bord : nombre de prêt

Exemple de vues créés 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;

```

2.10 Paramétrer les états :

Voir la documentation complète sur le site de documentation openMairie :

<https://openmairie.readthedocs.io/projects/omframework/fr/4.9/reference/edition.html#parametrer-des-etats>

2.10.1 Modification du formulaire état :

les champs id et libellé ne sont pas accessibles pour laisser pérenne le lien action d'édition / génération d'état dans le module de gestion des actions d'om5-no-code (om_actions).

2.11 Paramétrer les requêtes :

Voir la documentation complète sur le site de documentation openMairie :

<https://openmairie.readthedocs.io/projects/omframework/fr/4.9/reference/edition.html#les-requetes>

2.11.1 Modification du formulaire requête :

les champs type, code, libellé et description ne sont pas accessibles en mise à jour pour laisser pérenne le lien état / requête dans le module de gestion des actions d'om5-no-code pour la génération d'état (om_action).

Il est mis en place un assistant à la création des requêtes :

The image shows a multi-step SQL query builder assistant interface. It consists of four main sections, each with an assistant dropdown and a corresponding text area for the SQL clause:

- select:** The assistant dropdown is labeled "select assistant" and contains "choisir un champ". The text area contains a complex SQL query: `article.libelle AS libelle, round(article.pmp,2) AS pmp, article.quantite AS quantite, round(article.prix,2) AS prix, article.minimum AS minimum, case article.archive when 'Y' then 'Oui' else 'Non' end AS archive`.
- from:** The assistant dropdown is labeled "from" and contains "Type" and "jointure table -> table étrangère". The text area contains `&DB_PREFIXEarticle`.
- where:** The assistant dropdown is labeled "where assistant" and contains "Condition", "choisir un champ", and "égal". The text area contains `article.famille='&idx'`.
- order_by:** The assistant dropdown is labeled "order_by assistant" and contains "choisir un champ". The text area is empty.

2.12 Paramétrer les sous états :

Voir la documentation complète sur le site de documentation openMairie :

<https://openmairie.readthedocs.io/projects/omframework/fr/4.9/reference/edition.html#les-sous-etats>

2.12.1 Modification du formulaire sous état :

les champs id et libellé ne sont pas accessibles en mise à jour pour laisser pérenne le lien état /sous état dans le module de gestion des actions d'om5-no-code pour les clés secondaires (om_forms).

Il est mis en place un assistant à la création des sous états :

The screenshot displays a SQL query builder interface with four sections, each featuring an assistant and a text input field:

- select:** The assistant shows a dropdown for "choisir un champ" and a plus sign. The text input contains:


```
article.libelle AS libelle,
round(article.pmp,2) AS pmp,
article.quantite AS quantite,
round(article.prix,2) AS prix,
article.minimum AS minimum,
case article.archive when '1' then 'Oui' else 'Non' end AS archive
```
- from:** The assistant shows a dropdown for "Type" and a dropdown for "jointure table -> table étrangère" with a plus sign. The text input contains:


```
&DB_PREFIXEarticle
```
- where:** The assistant shows a dropdown for "Condition", a dropdown for "choisir un champ", and a dropdown for "égal" with a plus sign. The text input contains:


```
article.famille='&idx'
```
- order_by:** The assistant shows a dropdown for "choisir un champ" and a plus sign. The text input is empty.

2.13 Paramétrer les widgets :

Voir la documentation complète sur le site de documentation openMairie :

https://openmairie.readthedocs.io/projects/omframework/fr/4.9/reference/dashboard.html?highlight=widget*

2.13.1 Modification du formulaire widget :

Seul le champ libellé du tableau de bord est accessible en mise à jour pour laisser pérenne le lien action / widget dans le module de gestion des actions d'om5-no-code pour la génération d'état (om_action).

Il est présenter dans ce chapitre la description technique 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 composé 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'intègrent pas les règles du modèle de CODD des bases de données (cle/valeur, redondances)
- aux systèmes d'information géographique qui n'intègrent 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 formulaires `om_champs` et `om_contrainte`

Les classes `om_état`, `om_sousetat`, `om_requête` et `om_widget` ont été surchargées :

- pour rendre pérenne les champs générés par les actions d'om5-no-code pour la génération d'état et de sous état : restriction de mise à jour des champs liant les actions aux états et sous états
- pour mettre en place des assistants à la mise en oeuvre de requêtes.

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/pgsql` : `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 paramétrage des libellés de champs

- def_obj_meth_settype_by_maj : pour le paramétrage 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.

Ce chapitre décrit la version om5-no-code d'openStock.

openStock, c'est

- 16 tables générées avec clé primaire, clés secondaires et séquences.
- 5 vues : 2 pour le tableau de bord et 3 vues pour l'affichage de clé secondaire
- 16 actions : 10 actions d'édition et 6 actions lançant des procédures
- 35 fonctions : 29 fonctions triggers lancées par 42 triggers et 6 procédures « action »
- 10 états générés et 10 requêtes générées avec 17 sous états générés associés,

Il est proposé à titre d'exemple, de décrire l'application de gestion de stock :

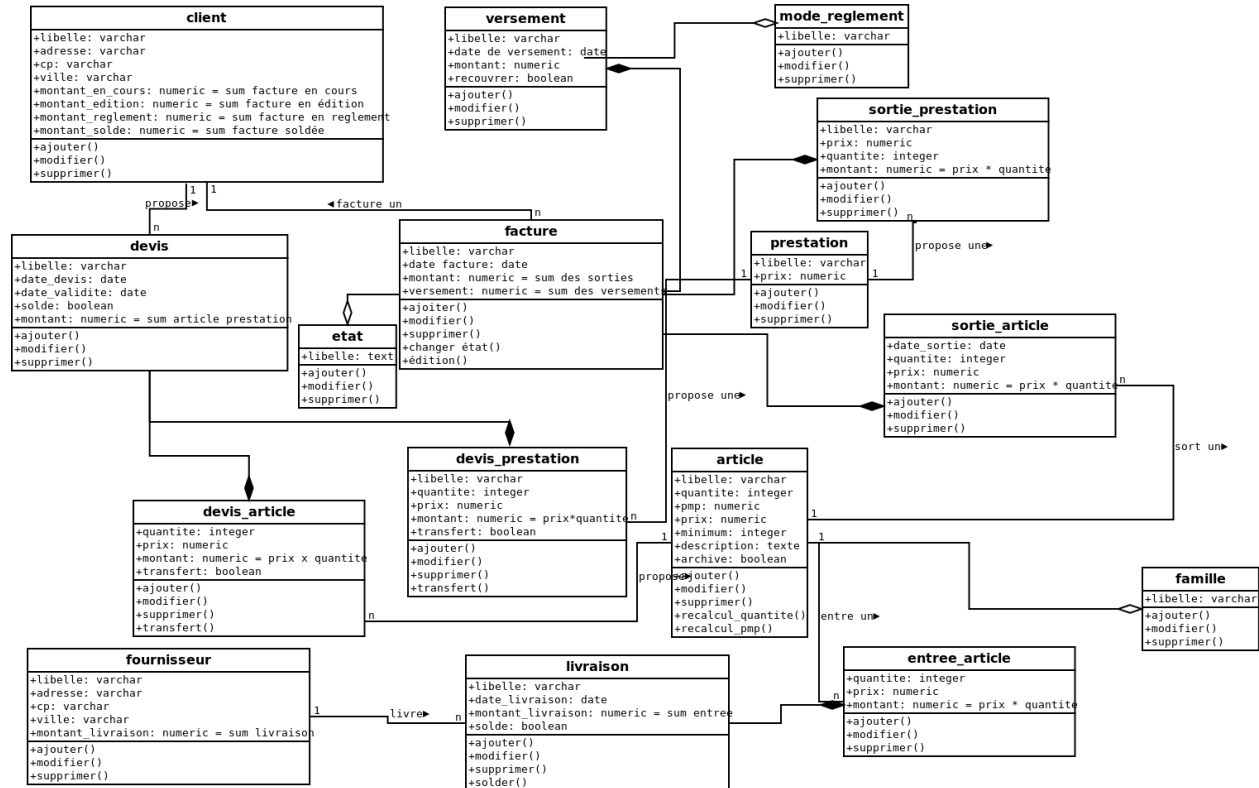
- les éléments d'uml
- les tables
- description de chaque table : champs, clés secondaires, actions, triggers, procédures et vues

4.1 Élément d'uml :

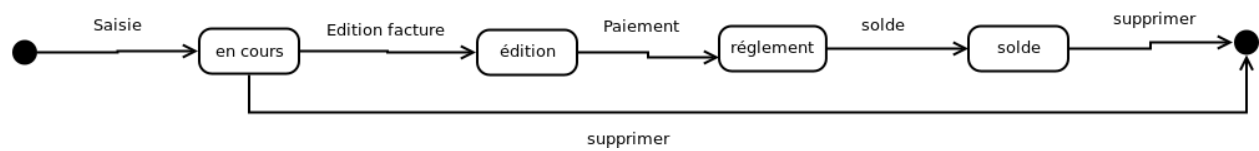
Il est proposé à titre d'exemple, de décrire l'application de gestion de stock :

- diagramme de classe
- diagramme état transition de la facture

4.1.1 Diagramme de classe :



4.1.2 Diagramme état/transition facture :



4.2 les tables :

openStock est composé de 16 tables

Table

1 - 15 enregistrement(s) sur 16 Page 1 / 2

Tous Recherche

+	table	libelle	col1	col2	col3	menu	nb_col	recherche
	article	article	Article	Description	colonne 3	parametre	2	ximple
	client	client	Adresse	Cumul	colonne 3	application	2	avancée
	devis	devis	colonne 1	colonne 2	colonne 3	application	1	avancée
	devis_article	devis_article	article	colonne 2	colonne 3	sans	1	ximple
	devis_prestation	devis_prestation	prestation	colonne 2	colonne 3	sans	1	ximple
	entree_article	entree_article	Libellé	montant	colonne 3	sans	2	ximple
	etat	etat	Libellé	colonne 2	colonne 3	parametre	1	ximple
	facture	facture	données	cumuls	colonne 3	application	2	avancée
	famille	famille	Libellé	colonne 2	colonne 3	parametre	1	ximple
	fournisseur	fournisseur	colonne 1	colonne 2	colonne 3	application	1	ximple
	livraison	livraison	colonne 1	colonne 2	colonne 3	application	1	ximple
	mode_reglement	mode de règlement	libelle	colonne 2	colonne 3	parametre	1	ximple
	prestation	prestation	colonne 1	colonne 2	colonne 3	parametre	1	ximple

table	menu	nb_col	recherche
article	2	2	0
client	1	2	1
devis	1	1	0
devis_prestation	0	1	0
devis_article	0	1	0
entree_article	0	2	0
etat	2	1	0
facture	1	2	1
famille	2	1	0
fournisseur	1	1	0
livraison	1	1	0
mode_reglement	2	1	0
prestation	2	1	0
sortie_article	0	1	0
sortie_prestation	0	1	0
versement	0	1	0

Les paramètres de la table sont stockées dans om_tables_parametre.

4.2.1 Menu application et menu paramètre :

▼ Application





sortie_article, sortie_prestation et versement sont accessibles dans les sous formulaires de facture.

entree_article est accessible dans un sous formulaire de livraison.

4.2.2 Le code sql créé par om_tables :

Les tables suivantes sont créées par om_tables :

```
CREATE TABLE article ( article integer NOT NULL );
CREATE TABLE client ( client integer NOT NULL );
CREATE TABLE entree_article ( entree_article integer NOT NULL );
CREATE TABLE etat ( etat integer NOT NULL );
CREATE TABLE devis ( devis integer NOT NULL );
CREATE TABLE devis_prestation ( devis_prestation integer NOT NULL );
CREATE TABLE devis_article ( devis-article integer NOT NULL );
CREATE TABLE facture ( facture integer NOT NULL );
CREATE TABLE famille ( famille integer NOT NULL);
CREATE TABLE fournisseur (fournisseur integer NOT NULL);
CREATE TABLE livraison (livraison integer NOT NULL);
CREATE TABLE mode_reglement (mode_reglement integer NOT NULL);
CREATE TABLE prestation (prestation integer NOT NULL );
CREATE TABLE sortie_article (sortie_article integer NOT NULL);
CREATE TABLE sortie_prestation (sortie_prestation integer NOT NULL);
CREATE TABLE versement (versement integer NOT NULL);
```

Les séquences suivantes sont créés

```
CREATE SEQUENCE article_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE client_seq      START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE devis_seq      START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE devis_prestation_seq  START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE devis_article_seq    START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE etat_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
```

(suite sur la page suivante)

(suite de la page précédente)

```

CREATE SEQUENCE entree_article_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE facture_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE famille_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE fournisseur_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE livraison_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE mode_reglement_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE prestation_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE sortie_article_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE sortie_prestation_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;
CREATE SEQUENCE versement_seq START WITH 1 INCREMENT BY 1
    NO MINVALUE NO MAXVALUE CACHE 1;

```

Les clés primaires sont créées automatiquement dans om_tables, elles ont le même nom que la table et elles sont de type integer.

code

```

-- pk
ALTER TABLE ONLY article
    ADD CONSTRAINT article_pkey PRIMARY KEY (article);
ALTER TABLE ONLY client
    ADD CONSTRAINT client_pkey PRIMARY KEY (client);
ALTER TABLE ONLY entree_article
    ADD CONSTRAINT entree_article_pkey PRIMARY KEY (entree_article);
ALTER TABLE ONLY etat
    ADD CONSTRAINT etat_pkey PRIMARY KEY (etat);
ALTER TABLE ONLY devis
    ADD CONSTRAINT devis_pkey PRIMARY KEY (devis);
ALTER TABLE ONLY devis_prestation
    ADD CONSTRAINT devis_prestation_pkey PRIMARY KEY (devis_prestation);
ALTER TABLE ONLY devis_article
    ADD CONSTRAINT devis_article_pkey PRIMARY KEY (devis_article);
ALTER TABLE ONLY facture
    ADD CONSTRAINT facture_pkey PRIMARY KEY (facture);
ALTER TABLE ONLY famille
    ADD CONSTRAINT famille_pkey PRIMARY KEY (famille);
ALTER TABLE ONLY fournisseur
    ADD CONSTRAINT fournisseur_pkey PRIMARY KEY (fournisseur);
ALTER TABLE ONLY livraison
    ADD CONSTRAINT livraison_pkey PRIMARY KEY (livraison);
ALTER TABLE ONLY mode_reglement
    ADD CONSTRAINT mode_reglement_pkey PRIMARY KEY (mode_reglement);
ALTER TABLE ONLY om_actions
    ADD CONSTRAINT om_actions_pkey PRIMARY KEY (om_actions);
ALTER TABLE ONLY om_forms
    ADD CONSTRAINT om_forms_pkey PRIMARY KEY (column_name);
ALTER TABLE ONLY om_tables_parametre
    ADD CONSTRAINT om_tables_parametre_pkey PRIMARY KEY (table_name);

```

(suite sur la page suivante)

```
ALTER TABLE ONLY prestation
    ADD CONSTRAINT prestation_pkey PRIMARY KEY (prestation);
ALTER TABLE ONLY sortie_article
    ADD CONSTRAINT sortie_article_pkey PRIMARY KEY (sortie_article);
ALTER TABLE ONLY sortie_prestation
    ADD CONSTRAINT sortie_prestation_pkey PRIMARY KEY (sortie_prestation);
ALTER TABLE ONLY versement
    ADD CONSTRAINT versement_pkey PRIMARY KEY (versement);
```

Les droits d'accès sont créés dans om_droit pour chaque table.

4.3 article

4.3.1 formulaire

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 8 enregistrement(s) sur 8														
+	▶ champ	▶ pgsql	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	article.archive	boolean		Non	archive			Visible	Visible			C1-7	Oui	7-archive
	article.description	text		Non		html	html	Visible	Non visible			C2-1	Non	-description
	article.famille	integer	famille	Oui	famille			Visible	Visible			C1-2	Oui	2-famille
	article.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	article.minimum	integer		Non	minimum			Visible	Visible			C1-6	Oui	6-minimum
	article.pmp	numeric		Non	pmp	hiddenstatic	hiddenstatic	Visible	Non visible			C1-3	Oui	3-pmp
	article.prix	numeric		Oui	prix			Visible	Visible			C1-5	Oui	5-prix
	article.quantite	integer		Non	quantite	hiddenstatic	hiddenstatic	Non visible	Non visible			C1-4	Oui	4-quantite

La champs suivants sont créés

```
libelle character varying NOT NULL,
famille integer NOT NULL,
pmp numeric,
quantite integer,
prix numeric NOT NULL,
minimum integer
```

Les paramètres suivants ont été créés dans om_forms_parametre

pmp est le prix moyen pondéré.

4.3.2 clé secondaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Article Om5								
Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues		
1 - 1 enregistrement(s) sur 1								
▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ			
article_famille_fkey	om5	article	famille	famille	famille			

La clé secondaire suivante est créée par om_forms

```
ALTER TABLE ONLY article
ADD CONSTRAINT article_famille_fkey FOREIGN KEY (famille)
REFERENCES famille(famille);
```

4.3.3 Les Vues

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Article Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 3 enregistrement(s) sur 3

+	▶ table_name	▶ table_schema	▶ view_definition
🔍	article\$1	om5	SELECT article.libelle, article.pmp, article.prix, article.quantite, article.minimum FROM om5.article WHERE (article.minimum > article.quantite) ORDER BY article.libelle;
🔍	article\$2	om5	SELECT article.article, (concat(article.libelle, '[', famille.libelle, ']'))::character varying AS libelle FROM (om5.article LEFT JOIN om5.famille ON ((article.famille = famille.famille))) WHERE (article.archive IS NOT TRUE) ORDER BY article.libelle;
🔍	article\$3	om5	SELECT article.article, concat(article.libelle, '[', famille.libelle, ']') AS libelle FROM (om5.article JOIN om5.famille ON ((article.famille = famille.famille))) ORDER BY article.libelle;

```
-- article$1 : vue dans le widget des articles ou la quantité est inférieure au_
↳minimum

SELECT article.libelle, article.pmp,
article.prix, article.quantite,
article.minimum
FROM om5.article
WHERE (article.minimum > article.quantite)
ORDER BY article.libelle;

-- article$2 : vue utilisée en selection article : sortie_article select

SELECT article.article,
(concat(article.libelle, ' ', famille.libelle, ' '))::character varying AS_
↳libelle
FROM (om5.article
LEFT JOIN om5.famille ON ((article.famille = famille.famille)))
WHERE (article.archive IS NOT TRUE)
ORDER BY article.libelle;

-- article$2 : vue utilisée en selection article : sortie_article select by id

SELECT article.article,
concat(article.libelle, ' ', famille.libelle, ' ') AS libelle
FROM (om5.article
JOIN om5.famille ON ((article.famille = famille.famille)))
ORDER BY article.libelle;
```

4.3.4 Les actions :

- Recalcul de la quantité
- recalcul du pmp

```
-- mise a jour de la quantité article depuis les entrées et les sorties
v_1=(select coalesce(sum(quantite),0) from entree_article where article = idx);
v_2=(select coalesce(sum(quantite),0) from sortie_article where article = idx);
```

(suite sur la page suivante)

```

v_3=v_1-v_2;
IF (v_3<0 ) THEN
    v_3=0;
END IF;
update article set quantite=v_3 where article = idx;
message='quantité article mis à jour';

-- mise a jour du pmp article depuis les entrées et les sorties
v_1=(select coalesce(sum(montant),0) from entree_article where article = idx);
v_2=(select coalesce(sum(quantite),0) from sortie_article where article = idx);
IF (v_2=0 ) THEN
    v_3=0;
ELSE
    v_3=v_1/v_2;
END IF;
update article set prix=v_3 where article = idx;
message='pmp article mis à jour';

```

4.4 Client :

4.4.1 Formulaire

Om5 No Code → Gestion De Tables Et Génération Des Objets → Client Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 8 enregistrement(s) sur 8

+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	client.adresse	character varying		Non	adresse			Visible	Visible			C1-2	Oui	2-adresse
	client.cp	character varying		Non	cp			Visible	Visible			C1-3	Oui	3-cp
	client.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	client.montant_edition	numeric		Non	edition	hiddenstatic		Visible	Visible			C2-2	Oui	6-edition
	client.montant_en_cours	numeric		Non	en cours	hiddenstatic		Visible	Visible			C2-1	Oui	5-en cours
	client.montant_reglement	numeric		Non	reglement	hiddenstatic		Visible	Visible			C2-3	Oui	7-reglement
	client.montant_solde	numeric		Non	solde	hiddenstatic		Visible	Visible			C2-4	Oui	8-solde
	client.ville	character varying		Non	ville			Visible	Visible			C1-4	Oui	4-ville

La champs suivants sont créés

```

client integer NOT NULL,
libelle character varying NOT NULL,
adresse character varying,
cp character varying,
ville character varying,
montant_en_cours numeric,
montant_edition numeric,
montant_reglement numeric,
montant_solde numeric

```

montant_en_cours, montant_edition, montant_reglement, montant_solde sont des champs calculés (cumul de facture)

4.5 devis

4.5.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 7 enregistrement(s) sur 7														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	devis.client	integer	client	Oui	client			Visible	Visible			C1-4	Oui	4-client
	devis.date_devis	date		Oui	date_devis			Visible	Visible			C1-2	Oui	2-date_devis
	devis.date_validite	date		Non	date_validite	hiddenstatic		Visible	Visible			C1-6	Oui	6-date_validite
	devis.facture	integer	facture	Non	facture	hiddenstatic		Visible	Visible			C1-5	Oui	5-facture
	devis.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	devis.montant	numeric		Non	montant	hiddenstatic	hiddenstatic	Visible	Visible			C1-7	Oui	7-montant
	devis.solde	boolean		Non	solde			Visible	Visible			C1-3	Oui	3-solde

La champs suivants sont créés

```
libelle character varying NOT NULL,
date_devis date NOT NULL,
solde boolean,
client integer NOT NULL,
facture integer,
date_validite date,
montant numeric,
```

montant est un champs calculé.

Client et facture sont des clés secondaires.

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues	
1 - 2 enregistrement(s) sur 2							
	contrainte	schema	table	champ	foreign table	foreign champ	
	devis_client_fkey	om5	devis	client	client	client	
	devis_facture_fkey	om5	devis	facture	facture	facture	

Framework openMairie Version 4.10.0-om5_no_code-1.2.0a3 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

Les clés secondaires sont créées par om_forms

```
CONSTRAINT devis_client_fkey FOREIGN KEY (client)
REFERENCES om5.client (client) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT devis_facture_fkey FOREIGN KEY (facture)
REFERENCES om5.facture (facture) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
```

4.6 Devis prestations :

4.6.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 7 enregistrement(s) sur 7														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	devis_prestation.devis	integer	devis	Non	devis			Visible	Visible			C1-5	Oui	5-devis
	devis_prestation.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	devis_prestation.montant	numeric		Non	montant	hiddenstatic	hiddenstatic	Visible	Visible			C1-6	Oui	6-montant
	devis_prestation.prestation	integer	prestation	Oui	prestation		hiddenstatic	Visible	Visible			C1-2	Oui	2-prestation
	devis_prestation.prix	numeric		Non	prix	hiddenstatic	hiddenstatic	Visible	Visible			C1-3	Oui	3-prix
	devis_prestation.quantite	integer		Oui	quantite			Visible	Visible			C1-4	Oui	4-quantite
	devis_prestation.transfert	boolean		Non	transfert			Visible	Visible			C1-7	Oui	7-transfert

La champs suivants sont créés

```
libelle character varying NOT NULL,
prestation integer NOT NULL,
prix numeric,
quantite integer NOT NULL,
devis integer,
montant numeric,
transfert boolean,
```

Le champ montant est un champ calculé : prix x quantité.

devis et prestation sont des clés secondaires

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues	
1 - 2 enregistrement(s) sur 2							
	contrainte	schema	table	champ	foreign table	foreign champ	
	devis_prestation_devis_fkey	om5	devis_prestation	devis	devis	devis	
	devis_prestation_prestation_fkey	om5	devis_prestation	prestation	prestation	prestation	

Les clés secondaires suivantes sont créées par om_forms.

```
CONSTRAINT devis_prestation_devis_fkey FOREIGN KEY (devis)
REFERENCES om5.devis (devis) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT devis_prestation_prestation_fkey FOREIGN KEY (prestation)
REFERENCES om5.prestation (prestation) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION
```

4.6.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_prestation Om5						
Table Formulaire Action clé secondaire Procédures Triggers Vues						
1 - 6 enregistrement(s) sur 6						
+	nom	evenement	condition	procedure	sur	quand
	devis_prestation\$1	INSERT		EXECUTE PROCEDURE om5."devis_prestation\$1"()	ROW	AFTER
	devis_prestation\$2	UPDATE		EXECUTE PROCEDURE om5."devis_prestation\$1"()	ROW	AFTER
	devis_prestation\$3	INSERT		EXECUTE PROCEDURE om5."devis_prestation\$2"()	ROW	BEFORE
	devis_prestation\$4	UPDATE		EXECUTE PROCEDURE om5."devis_prestation\$2"()	ROW	BEFORE
	devis_prestation\$5	DELETE		EXECUTE PROCEDURE om5."devis_prestation\$3"()	ROW	AFTER
	devis_prestation\$6	DELETE		EXECUTE PROCEDURE om5."devis_prestation\$4"()	ROW	BEFORE

Il y a 6 triggers devis_article before et after pour insert, update et delete.

4.6.3 Les procédures triggers

- règle : le devis ne doit pas être soldé
- règle : récupération du prix de vente en prestation
- calcul du montant du devis prestation
- calcul du montant du devis

```
-- devis_prestation$1 : calcul du montant du devis en insert et update
DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;
BEGIN
v_1=(select sum(montant) from devis_prestation where devis = NEW.devis);
v_2=(select sum(montant) from devis_article where devis = NEW.devis);
v_3=coalesce(v_1,0)+coalesce(v_2,0);
update devis set montant=v_3 where devis = NEW.devis;
RETURN NEW;
END;

-- devis_prestation$2 : vérification devis soldé en insert et update
DECLARE
v_1 BOOLEAN;
BEGIN
v_1=(select solde from devis where devis=NEW.devis);
IF (v_1 = true) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les sorties de
↳prestation car le devis est soldé';
ELSE
    NEW.prix= (select prix from prestation where prestation=NEW.prestation);
    NEW.montant =NEW.prix *NEW.quantite;
END IF;
RETURN NEW;
END;

-- devis_prestation$3 calcul du montant du devis en delete
```

(suite sur la page suivante)

```

DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;
BEGIN
v_1=(select sum(montant) from devis_prestation where devis = OLD.devis);
v_2=(select sum(montant) from devis_article where devis = OLD.devis);
v_3=coalesce(v_1,0)+coalesce(v_2,0);
update devis set montant=v_3 where devis = OLD.devis;
RETURN OLD;
END;

-- devis_prestation$4 vérification devis soldé en delete
DECLARE
v_1 BOOLEAN;
BEGIN
v_1=(select solde from devis where devis=OLD.devis);
IF (v_1 = true) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les sorties de_
↳prestation car le devis est soldé';
END IF;
RETURN OLD;
END;

```

4.6.4 Action :

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_prestation Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 1 enregistrement(s) sur 1

+	om_actions	libellé	module	ordre	titre	table_name
	6	transfert_facture	devis_prestation\$5	100	Transfert	devis_prestation

Framework openMairie Version 4.10.0-om5 no code-1.2.0a3 | [Documentation](#) | [Forum](#) | [openMairie.org](#)

Cette action permet le transfert de la prestation en facture

Règles :

- le devis ne doit pas être soldé
- le devis doit être associé à une facture
- le transfert ne doit pas avoir été déjà effectué

```

-- devis_prestation$5 : transfert en facture
DECLARE
v_3 BOOLEAN;
message TEXT;
v_1 BOOLEAN;
v_2 INTEGER;
BEGIN
v_1=(select solde from devis inner join devis_prestation on devis.devis=devis_
↳prestation.devis where devis_prestation.devis_prestation=idx);
IF (v_1 = true) THEN
    message = 'Vous ne pouvez pas transférer une sortie de prestation car le_
↳devis est soldé';

```

(suite sur la page suivante)

(suite de la page précédente)

```

ELSE
    v_2=(select facture from devis inner join devis_prestation on devis.
↳devis=devis_prestation.devis where devis_prestation.devis_prestation=idx);
    v_2=coalesce(v_2,0);
    IF ( v_2=0) THEN
        message= 'Vous ne pouvez pas transférer une sortie de
↳prestation en facture car le devis n''est pas associé à une facture';
    ELSE
        v_3=(select transfert from devis_prestation where devis_
↳prestation.devis_prestation=idx);
        IF ( v_3= true ) THEN
            message= 'Le transfert de cette prestation en
↳facture a déjà été effectué';
        ELSE
            insert into sortie_prestation (sortie_prestation,
↳libelle, prestation, facture, quantite) select nextval('sortie_prestation_seq'),
↳devis_prestation.libelle, prestation, devis.facture, quantite from devis inner join
↳devis_prestation on devis.devis=devis_prestation.devis where devis_prestation.devis_
↳prestation=idx;
            update devis_prestation set transfert = true where
↳devis_prestation = idx;
            message= 'Transfert de la prestation effectué en
↳facture et mise à jour transfert';
        END IF;
    END IF;
END IF;
RETURN message;
END;

```

4.7 Devis articles :

4.7.1 formulaire

Om5 No Code ➤ Gestion De Tables Et Génération Des Objets ➤ Devis_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 6 enregistrement(s) sur 6														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	devis_article.article	integer	article	Oui	article			Visible	Visible	article\$2		C1-1	Oui	1-article
	devis_article.devis	integer	devis	Oui	devis			Visible	Visible			C1-2	Oui	2-devis
	devis_article.montant	numeric		Non	montant	hiddenstatic	hiddenstatic	Visible	Visible			C1-5	Oui	5-montant
	devis_article.prix	numeric		Non	prix	hiddenstatic	hiddenstatic	Visible	Visible			C1-4	Oui	4-prix
	devis_article.quantite	integer		Oui	quantite			Visible	Visible			C1-3	Oui	3-quantite
	devis_article.transfert	boolean		Non	transfert			Visible	Visible			C1-6	Oui	6-transfert

La champs suivants sont créés

```

article integer NOT NULL,
devis integer NOT NULL,
quantite integer NOT NULL,
prix numeric,
montant numeric,
transfert boolean,

```

Le champ montant est un champ calculé : prix x quantité.

devis et article sont des clés secondaires

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 2 enregistrement(s) sur 2						
contrainte	schema	table	champ	foreign table	foreign champ	
devis_article_article_fkey	om5	devis_article	article	article	article	
devis_article_devis_fkey	om5	devis_article	devis	devis	devis	

Les clés secondaires suivantes sont créées par om_forms.

```

CONSTRAINT devis_article_article_fkey FOREIGN KEY (article)
REFERENCES om5.article (article) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT devis_article_devis_fkey FOREIGN KEY (devis)
REFERENCES om5.devis (devis) MATCH SIMPLE
ON UPDATE NO ACTION ON DELETE NO ACTION

```

4.7.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Devis_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 6 enregistrement(s) sur 6						
nom	evenement	condition	procedure	sur	quand	
devis_article\$1	INSERT		EXECUTE PROCEDURE om5."devis_article\$1"()	ROW	AFTER	
devis_article\$2	UPDATE		EXECUTE PROCEDURE om5."devis_article\$1"()	ROW	AFTER	
devis_article\$3	INSERT		EXECUTE PROCEDURE om5."devis_article\$2"()	ROW	BEFORE	
devis_article\$4	UPDATE		EXECUTE PROCEDURE om5."devis_article\$2"()	ROW	BEFORE	
devis_article\$5	DELETE		EXECUTE PROCEDURE om5."devis_article\$3"()	ROW	AFTER	
devis_article\$6	DELETE		EXECUTE PROCEDURE om5."devis_article\$4"()	ROW	BEFORE	

Il y a 6 triggers devis_article before et after pour insert, update et delete.

4.7.3 Les procédures

- règle : le devis ne doit pas être soldé
- règle : récupération du prix de vente en prestation
- calcul du montant du devis prestation
- calcul du montant du devis

```

-- devis_article$1 : calcul montant devis pour insert et update
DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;
BEGIN
v_1=(select sum(montant) from devis_prestation where devis = NEW.devis);

```

(suite sur la page suivante)

(suite de la page précédente)

```

v_2=(select sum(montant) from devis_article where devis = NEW.devis);
v_3=coalesce(v_1,0)+coalesce(v_2,0);
update devis set montant=v_3 where devis = NEW.devis;
RETURN NEW;
END;

-- devis_article$2 : vérification du solde devis et calcul montant pour insert et
↳delete
DECLARE
v_1 BOOLEAN;
BEGIN
v_1=(select solde from devis where devis=NEW.devis);
IF (v_1 = true) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les sorties de
↳article
    car le devis est solde';
ELSE
    NEW.prix= (select prix from article where article=NEW.article);
    NEW.montant =NEW.prix *NEW.quantite;
END IF;
RETURN NEW;
END;

--devis_article$3 : calcul montant du devis pour delete
DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;
BEGIN
v_1=(select sum(montant) from devis_prestation where devis = OLD.devis);
v_2=(select sum(montant) from devis_article where devis = OLD.devis);
v_3=coalesce(v_1,0)+coalesce(v_2,0);
update devis set montant=v_3 where devis = OLD.devis;
RETURN OLD;
END;

-- devis_article$4 : vérification devis soldé pour delete
DECLARE
v_1 BOOLEAN;
BEGIN
v_1=(select solde from devis where devis=OLD.devis);
IF (v_1 = true) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les sorties
article car le devis est soldé';
END IF;
RETURN OLD;
END;

```

4.7.4 Action :

Om5 No Code ➤ Gestion De Tables Et Génération Des Objets ➤ Devis_article Om5

Table Formulaire **Action** clé secondaire Procédures Triggers Vues

1 - 1 enregistrement(s) sur 1

+	om_actions	libellé	module	ordre	titre	table_name
		7 transfert_facture	devis_article\$5		100 Transfert	devis_article

Cette action permet le transfert de l'article en facture

Règles :

- le devis ne doit pas être soldé
- le devis doit être associé à une facture
- le transfert ne doit pas avoir été déjà effectué
- la quantité sortie ne peut pas être égale ou inférieure à 0 (règle de sortie_article)
- la quantité en article doit être inférieure ou égale à la quantité sortie (règle de sortie_article)

```
-- devis_article$5
DECLARE
v_1 BOOLEAN;
v_2 INTEGER;
v_3 BOOLEAN;
v_4 INTEGER;
v_5 INTEGER;
message TEXT;
BEGIN
v_1=(select solde from devis inner join devis_article on devis.devis=devis_article.
↳devis where devis_article.devis_article=idx);
IF (v_1 = true) THEN
    message = 'Vous ne pouvez pas transférer cet article car le devis est soldé';
ELSE
    v_2=(select facture from devis inner join devis_article on devis.
↳devis=devis_article.devis where devis_article.devis_article=idx);
    v_2=coalesce(v_2,0);
    IF ( v_2=0) THEN
        message= 'Vous ne pouvez pas transférer cet article en
↳facture car le devis n''est pas associé à une facture';
    ELSE
        v_3=(select transfert from devis_article where devis_article.
↳devis_article=idx);
        IF ( v_3= true ) THEN
            message= 'Le transfert de cette article en facture
↳a déjà été effectué';
        ELSE
            -- quantité négative
            v_4=(select quantite from devis_article where devis_
↳article.devis_article=idx);
            IF (v_4 <= 0) THEN
                message = 'Vous ne pouvez pas sortir une
↳quantité négative ou égale a 0';
            ELSE
                -- quantite insuffisante en article
                v_5=(select article.quantite from article
↳inner join devis_article on devis_article.article=article.article where devis_
↳article.article=idx);
```

(suite sur la page suivante)

(suite de la page précédente)

```

                                IF ( v_4>v_5) THEN
                                    message = 'Vous n''avez pas les_
↳quantités en stock de cet article';
                                ELSE
                                    insert into sortie_article (sortie_
↳article, date_sortie, article, facture, quantite) select nextval('sortie_article_seq
↳'), CURRENT_DATE, article, devis.facture, quantite from devis inner join devis_
↳article on devis.devis=devis_article.devis where devis_article.devis_article=idx;
                                    update devis_article set transfert_
↳= true where devis_article = idx;
                                    message= 'Transfert de l''article_
↳effectué en facture et mise à jour transfert';
                                END IF;
                                END IF;
                                END IF;
END IF;
RETURN message;
END;
```

4.8 Entree des articles

4.8.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Entree_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 5 enregistrement(s) sur 5														
+	▶ champ	▶ pgsq	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	entree_article.article	integer	article	Oui	article		hiddenstatic	Visible	Visible			C1-1	Oui	1-article
	entree_article.livraison	integer	article	Oui	livraison			Visible	Visible			C1-4	Oui	4-livraison
	entree_article.montant	numeric		Non	montant	hiddenstatic	hiddenstatic	Visible	Visible			C1-5	Oui	5-montant
	entree_article.prix	numeric		Oui	prix			Visible	Visible			C1-3	Oui	3-prix
	entree_article.quantite	integer		Oui	quantite			Visible	Visible			C1-2	Oui	2-quantite

La champs suivants sont créés

```

quantite integer NOT NULL,
prix numeric NOT NULL,
article integer NOT NULL,
livraison integer NOT NULL,
montant numeric
```

Le champ montant est un champ calculé : prix x quantité.

Livraison et article sont des clés secondaires

4.8.2 clé secondaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Entree_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 2 enregistrement(s) sur 2						
contrainte	schema	table	champ	foreign table	foreign champ	
entree_article_article_fkey	om5	entree_article	article	article	article	
entree_article_livraison_fkey	om5	entree_article	livraison	livraison	livraison	

Les clés secondaires suivantes sont créées par om_forms.

```
-- article
ALTER TABLE ONLY entree_article
    ADD CONSTRAINT entree_article_article_fkey
        FOREIGN KEY (article) REFERENCES article(article);
-- livraison
ALTER TABLE ONLY entree_article
    ADD CONSTRAINT entree_article_livraison_fkey
        FOREIGN KEY (livraison) REFERENCES livraison(livraison);
```

4.8.3 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Entree_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 3 enregistrement(s) sur 3						
+	nom	evenement	condition	procedure	sur	quand
	entree_article\$1	INSERT		EXECUTE PROCEDURE om5."entree_article\$1"()	ROW	BEFORE
	entree_article\$2	UPDATE		EXECUTE PROCEDURE om5."entree_article\$1"()	ROW	BEFORE
	entree_article\$3	DELETE		EXECUTE PROCEDURE om5."entree_article\$2"()	ROW	AFTER

Il y a 3 triggers entree_article\$1 (insert), entree_article\$2 (update) et entree_article\$3 (delete)

4.8.4 Les procédures

- règle : la livraison ne doit pas être soldée
- règle : en delete, la quantité restante d'article ne peut pas être négative
- règle : en mise à jour, on ne peut pas changer d'article
- calcul : quantité et prix moyen pondéré article (pmp)
- calcul : montant de la livraison

```
-- entree_article$1 (insert et update)

DECLARE
v_1 numeric;
v_2 integer;
v_3 numeric;
v_4 INTEGER;
v_5 BOOLEAN;
```

(suite sur la page suivante)

(suite de la page précédente)

```

BEGIN
-- livraison soldée
v_5=(select solde_livraison from livraison where livraison = NEW.livraison);
IF (v_5='t' ) THEN
    RAISE EXCEPTION 'vous ne pouvez pas modifier une livraison soldée';
END IF;

-- quantite négative ou égal a 0 et prix negatif
IF ( NEW.quantite <= 0) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas entrer une quantite negative ou egal a 0
↳';
END IF;
IF ( NEW.prix < 0) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas entrer un prix negatif';
END IF;
-- montant
NEW.montant =NEW.prix *NEW.quantite;
-- montant livraison
v_3=(SELECT coalesce(sum(entree_article.montant),0) FROM entree_article WHERE entree_
↳article.livraison =NEW.livraison) ;
UPDATE livraison SET montant_livraison=v_3 WHERE livraison.livraison = NEW.livraison_
↳;
-- calcul de quantite et de prix moyen pondere (pmp) pour article
v_1=(SELECT coalesce(pmp,0) FROM article WHERE article.article =NEW.article) ;
v_2=(SELECT coalesce(quantite,0) FROM article WHERE article.article =NEW.article) ;
IF ( TG_OP='INSERT' ) THEN
    v_4=v_2+NEW.quantite;
    IF (v_4=0 ) THEN -- éviter la division par 0
        UPDATE article SET quantite =0, pmp=0 where article.article =_
↳NEW.article ;
    ELSE
        v_3=round((v_1+v_2+NEW.montant)/(v_2+NEW.quantite),2);
        UPDATE article SET quantite = v_2+NEW.quantite, pmp=v_3 where_
↳article.article = NEW.article ;
    END IF;
ELSE -- UPDATE
    IF ( NEW.article != OLD.article ) THEN -- changement article interdit
        RAISE EXCEPTION 'Vous ne pouvez pas changer d'article en_
↳modification';
    ELSE
        v_4= v_2+NEW.quantite-OLD.quantite;
        IF (v_4=0 ) THEN -- eviter division par 0 -> quantite et pmp = 0
            UPDATE article SET quantite =0, pmp=0 where article.
↳article = NEW.article ;
        ELSE
            v_3=round((v_1*v_2+(NEW.montant-OLD.montant))/(v_2+NEW.
↳quantite-OLD.quantite),2);
            UPDATE article SET quantite = v_2+NEW.quantite-OLD.
↳quantite, pmp=v_3 where article.article = NEW.article ;
        END IF;
    END IF;
END IF;

RETURN NEW;

END;

```

(suite sur la page suivante)

```

-- entree_article$2 : delete

DECLARE
v_1 NUMERIC;
v_2 INTEGER;
v_3 NUMERIC;
v_4 BOOLEAN;

BEGIN
-- livraison soldée
v_4=(select solde_livraison from livraison where livraison = NEW.livraison);
IF (v_4='t' ) THEN
    RAISE EXCEPTION 'vous ne pouvez pas modifier une livraison soldée';
END IF;
-- montant livraison
v_3=(SELECT sum(entree_article.montant) FROM entree_article WHERE entree_article.
↳ livraison =OLD.livraison) ;
UPDATE livraison SET montant_livraison=v_3 WHERE livraison.livraison = OLD.livraison_
↳ ;
-- calcul de quantite et de prix moyen pondere (pmp) pour article
v_1=(SELECT coalesce(pmp,0) FROM article WHERE article.article =OLD.article) ;
v_2=(SELECT coalesce(quantite,0) FROM article WHERE article.article =OLD.article) ;
-- cas division par 0 ou quantite restante negative
IF (v_2<=OLD.quantite ) THEN
    UPDATE article SET quantite =0, pmp=0 where article.article = OLD.article_
↳ ;
ELSE
    v_3=round((v_1*v_2-OLD.montant)/(v_2-OLD.quantite),2);
    UPDATE article SET quantite = v_2-OLD.quantite, pmp=v_3 where article.
↳ article = OLD.article ;
END IF;

RETURN OLD;

```

4.9 Etat de la facture

4.9.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Etat Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues									
1 - 1 enregistrement(s) sur 1															
	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos	
	etat.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle	

Le champs suivant est créé

```
libelle character varying NOT NULL
```


4.9.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Etat Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 1 enregistrement(s) sur 1						
+	nom	evenement	condition	procedure	sur	quand
	etat\$1	DELETE		EXECUTE PROCEDURE om5."etat\$1"()	ROW	AFTER

Il y a 1 trigger etat\$1 delete before

4.9.3 Les procédures

— règle : on ne peut pas supprimer les états 1 à 4 qui sont dans le workflow

```
-- etat$1 (delete)

DECLARE
BEGIN
IF (OLD.etat <= 4) THEN
RAISE EXCEPTION 'Vous ne pouvez pas supprimer un état d'origine pour l''état %',
↳OLD.etat
↳OLD.etat
USING ERRCODE = 'unique_violation';
END IF;

RETURN OLD;
END;
```

4.10 facture

4.10.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 6 enregistrement(s) sur 6														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	facture.client	integer	client	Non	client			Visible	Visible			C1-2	Oui	2-client
	facture.date_facture	date		Oui	date_facture			Visible	Visible			C1-3	Oui	3-date_facture
	facture.etat	integer	etat	Oui	etat			Visible	Visible			C1-4	Oui	4-etat
	facture.libelle	character varying		Oui	Libelle			Visible	Visible			C1-1	Oui	1-Libelle
	facture.montant	numeric		Non	montant	hidden		Visible	Visible			C1-5	Oui	5-montant
	facture.versement	numeric		Non	versement	hiddenstatic		Visible	Visible			C1-6	Oui	6-versement

La champs suivants sont créés

```
facture integer NOT NULL,
libelle character varying NOT NULL,
client integer,
date_facture date NOT NULL,
```

(suite sur la page suivante)

```
etat integer NOT NULL,
montant numeric,
versement numeric
```

Versement et montant sont des champs calculés.

Client et état sont des clés secondaires.

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 2 enregistrement(s) sur 2						
contrainte	schema	table	champ	foreign table	foreign champ	
 facture_client_fkey	om5	facture	client	client	client	
 facture_etat_fkey	om5	facture	etat	etat	etat	




Les clés secondaires sont créées par om_forms

```
ALTER TABLE ONLY facture
ADD CONSTRAINT facture_client_fkey
FOREIGN KEY (client) REFERENCES client(client);

ALTER TABLE ONLY facture
ADD CONSTRAINT facture_etat_fkey
FOREIGN KEY (etat) REFERENCES etat(etat);
```

4.10.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 3 enregistrement(s) sur 3						
+	nom	evenement	condition	procedure	sur	quand
	facture\$1	UPDATE		EXECUTE PROCEDURE om5."facture\$1"()	ROW	AFTER
	facture\$2	INSERT		EXECUTE PROCEDURE om5."facture\$1"()	ROW	AFTER
	facture\$3	DELETE		EXECUTE PROCEDURE om5."facture\$2"()	ROW	AFTER

4.10.3 Les actions

- action de changement d'état
- action d'édition de facture

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Facture Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 2 enregistrement(s) sur 2						
+	om_actions	libellé	module	ordre	titre	table_name
		3 changement_etat	facture\$3		100 Suivant	facture
		5 facture	edition		50 Facture	facture

4.10.4 Les procédures

— calcul des montants de client pour chaque état de facture

```
-- facture$1 : calcul montant pour insert et update

DECLARE
v_1 numeric;
v_2 numeric;
v_3 numeric;
v_4 numeric;

BEGIN
v_1=(select sum(montant) from facture where client = NEW.client and NEW.etat = 1);
update client set montant_en_cours=v_1 where client = NEW.client;
v_2=(select sum(montant) from facture where client = NEW.client and NEW.etat = 2);
update client set montant_edition=v_2 where client = NEW.client;
v_3=(select sum(montant) from facture where client = NEW.client and NEW.etat = 3);
update client set montant_reglement=v_3 where client = NEW.client;
v_4=(select sum(montant) from facture where client = NEW.client and NEW.etat = 4);
update client set montant_solde=v_4 where client = NEW.client;

RETURN NEW;
END;

-- facture$2 : calcul des montant client delete

DECLARE
v_1 numeric;
v_2 numeric;
v_3 numeric;
v_4 numeric;

BEGIN
v_1=(select sum(montant) from facture where client = OLD.client and OLD.etat = 1);
update client set montant_en_cours=v_1 where client = OLD.client;
v_2=(select sum(montant) from facture where client = OLD.client and OLD.etat = 2);
update client set montant_edition=v_2 where client = OLD.client;
v_3=(select sum(montant) from facture where client = OLD.client and OLD.etat = 3);
update client set montant_reglement=v_3 where client = OLD.client;
v_4=(select sum(montant) from facture where client = OLD.client and OLD.etat = 4);
update client set montant_solde=v_4 where client = OLD.client;

RETURN OLD;
END;
```

(suite sur la page suivante)

```

-- facture$3 : action de changement d'état

DECLARE
v_1 INTEGER;
message TEXT;
v_2 INTEGER;

BEGIN
-- etat courant
v_1=(select etat from facture where facture = idx);
-- au dela de etat=4 retour etat en cours
IF (v_1=4) THEN
    v_2=1;
    message='Mise à jour état effectuée pour état origine';
ELSE
    v_2=v_1+1;
    message='Mise à jour état effectuée pour état suivant';
END IF;
-- requete
update facture set etat=v_2 where facture = idx;

RETURN message;
END;

```

4.11 Famille d'article

4.11.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Famille Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 1 enregistrement(s) sur 1														
	▶ champ	▶ pgsql	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	famille.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle

La champs suivants sont créés

```
libelle character varying NOT NULL
```

4.12 Fournisseur

4.12.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Fournisseur Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 5 enregistrement(s) sur 5														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	fournisseur.adresse	character varying		Non	adresse			Visible	Visible			C1-2	Oui	2-adresse
	fournisseur.cp	character varying		Non	cp			Visible	Visible			C1-3	Oui	3-cp
	fournisseur.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	fournisseur.montant_livraison	numeric		Non	livraison			Visible	Visible			C1-6	Oui	6-livraison
	fournisseur.ville	character varying		Non	ville			Visible	Visible			C1-4	Oui	4-ville

La champs suivants sont créés

```
libelle character varying NOT NULL,
adresse character varying,
cp character varying,
ville character varying,
montant_livraison character varying
```

montant_livraison est un champ calculé lors de livraisons

4.13 livraison

4.13.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livraison Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 5 enregistrement(s) sur 5														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	livraison.date_livraison	date		Oui	date_livraison		hiddenstatic	Visible	Visible			C1-2	Oui	2-date_livraison
	livraison.fournisseur	integer	fournisseur	Non	fournisseur			Visible	Visible			C1-6	Oui	6-fournisseur
	livraison.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	livraison.montant_livraison	numeric		Non	montant_livraison	hiddenstatic	hiddenstatic	Visible	Visible			C1-4	Oui	5-montant livraison
	livraison.solde_livraison	boolean		Non	solde_livraison	hiddenstatic	hiddenstatic	Visible	Non visible			C1-5	Oui	4-solde livraison

La champs suivants sont créés

```
libelle character varying NOT NULL,
date_livraison date NOT NULL,
solde_livraison boolean,
montant_livraison numeric,
fournisseur integer
```

- montant_livraison est un montant calculé des entrées d'article
- fournisseur est une clé secondaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livraison Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 1 enregistrement(s) sur 1						
contrainte	schema	table	champ	foreign table	foreign champ	
livraison_fournisseur_fkey	om5	livraison	fournisseur	fournisseur	fournisseur	

La clé secondaire suivante est créée par om_forms

```
ALTER TABLE ONLY livraison
ADD CONSTRAINT livraison_fournisseur_fkey
FOREIGN KEY (fournisseur) REFERENCES fournisseur(fournisseur);
```

4.13.2 Les actions

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livraison Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 1 enregistrement(s) sur 1						
om_actions	libellé	module	ordre	titre	table_name	
	4 solde_livraison	livraison\$3		100 Solde	livraison	

Cette action permet de changer l'état de la livraison : soldée ou pas

4.13.3 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Livraison Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 3 enregistrement(s) sur 3						
nom	evenement	condition	procedure	sur	quand	
livraison\$1	INSERT		EXECUTE PROCEDURE om5."livraison\$1"()	ROW	AFTER	
livraison\$2	UPDATE		EXECUTE PROCEDURE om5."livraison\$1"()	ROW	AFTER	
livraison\$3	DELETE		EXECUTE PROCEDURE om5."livraison\$2"()	ROW	AFTER	

Ces triggers insert, update et delete permettent de calculer le montant fournisseur

4.13.4 Les procédures

```
-- livraison$1 : calcul montant fournisseur insert et update
DECLARE
v_1 NUMERIC;

BEGIN
v_1=(SELECT COALESCE(SUM(montant_livraison),0) FROM livraison where fournisseur =NEW.
↪fournisseur) ;
```

(suite sur la page suivante)

(suite de la page précédente)

```

UPDATE fournisseur SET montant_livraison = v_1 WHERE fournisseur.fournisseur =NEW.
↳fournisseur ;

RETURN NEW;
END;

-- livraison$2 : calcul montant fournisseur delete

DECLARE
v_1 NUMERIC;

BEGIN
v_1=(SELECT COALESCE(SUM(montant_livraison),0) FROM livraison where fournisseur =OLD.
↳fournisseur) ;
UPDATE fournisseur SET montant_livraison = v_1 WHERE fournisseur.fournisseur =OLD.
↳fournisseur ;

RETURN OLD;
END;

-- livraison$3 : action de changement d'état du solde livraison

DECLARE
message TEXT;
v_1 BOOLEAN;

BEGIN
v_1=(select solde_livraison from livraison where livraison=idx);
IF ( v_1 = 't' ) THEN
    UPDATE livraison SET solde_livraison = FALSE where livraison=idx;
    message='la livraison n''est pas soldée';
ELSE
    UPDATE livraison SET solde_livraison = TRUE where livraison=idx;
    message='la livraison est soldée';
END IF;

RETURN message;
END;

```

4.14 Mode de règlement du versement

4.14.1 formulaire

Om5 No Code ➤ Gestion De Tables Et Génération Des Objets ➤ Mode_reglement Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 1 enregistrement(s) sur 1														
+	▶ champ	▶ pgsq1	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	mode_reglement.libelle	character varying		Oui	libellé			Visible	Visible			C1-1	Oui	1-libellé

La champs suivants sont créés

```
libelle character varying NOT NULL
```

4.15 Prestation

4.15.1 formulaire

Om5 No Code → Gestion De Tables Et Génération Des Objets → Prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 2 enregistrement(s) sur 2														
+	▶ champ	▶ pgsq	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	prestation.libelle	character varying		Oui	libelle			Visible	Visible			C1-1	Oui	1-libelle
	prestation.prix	numeric		Oui	prix			Visible	Visible			C1-2	Oui	2-prix

La champs suivants sont créés

```
libelle character varying NOT NULL
prix numeric NOT NULL
```

4.16 Sortie des articles

4.16.1 formulaire

Om5 No Code → Gestion De Tables Et Génération Des Objets → Sortie_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 6 enregistrement(s) sur 6														
+	▶ champ	▶ pgsq	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	sortie_article.article	integer	article	Oui	article		hiddenstatic	Visible	Visible	article\$2	article\$3	C1-1	Oui	1-article
	sortie_article.date_sortie	date		Oui	date de sortie			Visible	Visible			C1-2	Oui	2-sortie le
	sortie_article.facture	integer	article	Oui	facture			Visible	Visible			C1-6	Oui	6-facture
	sortie_article.montant	numeric		Non	montant	hiddenstatic	hiddenstatic	Visible	Visible			C1-5	Oui	5-montant
	sortie_article.prix	numeric		Non	prix	hiddenstatic	hiddenstatic	Visible	Visible			C1-4	Oui	4-prix
	sortie_article.quantite	integer		Oui	quantite			Visible	Visible			C1-3	Oui	3-quantite

La champs suivants sont créés

```
article integer NOT NULL,
date_sortie date NOT NULL,
quantite integer NOT NULL,
prix numeric,
montant numeric,
facture integer NOT NULL
```

Le champ montant est un champ calculé : prix x quantité.

facture et article sont des clés secondaires

Om5 No Code ➤ Gestion De Tables Et Génération Des Objets ➤ Sortie_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 2 enregistrement(s) sur 2						
contrainte	schema	table	champ	foreign table	foreign champ	
sortie_article_article_fkey	om5	sortie_article	article	article	article	
sortie_article_facture_fkey	om5	sortie_article	facture	facture	facture	

Les clés secondaires suivantes sont créées par om_forms.

```
ALTER TABLE ONLY sortie_article
ADD CONSTRAINT sortie_article_article_fkey
FOREIGN KEY (article) REFERENCES article(article);

ALTER TABLE ONLY sortie_article
ADD CONSTRAINT sortie_article_facture_fkey
FOREIGN KEY (facture) REFERENCES facture(facture);
```

4.16.2 Les triggers

Om5 No Code ➤ Gestion De Tables Et Génération Des Objets ➤ Sortie_article Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 6 enregistrement(s) sur 6						
+	nom	evenement	condition	procedure	sur	quand
	sortie_article\$1	INSERT		EXECUTE PROCEDURE om5."sortie_article\$1"()	ROW	AFTER
	sortie_article\$2	UPDATE		EXECUTE PROCEDURE om5."sortie_article\$1"()	ROW	AFTER
	sortie_article\$3	DELETE		EXECUTE PROCEDURE om5."sortie_article\$2"()	ROW	AFTER
	sortie_article\$4	INSERT		EXECUTE PROCEDURE om5."sortie_article\$3"()	ROW	BEFORE
	sortie_article\$5	UPDATE		EXECUTE PROCEDURE om5."sortie_article\$3"()	ROW	BEFORE
	sortie_article\$6	DELETE		EXECUTE PROCEDURE om5."sortie_article\$4"()	ROW	BEFORE

Il y a 5 triggers sortie_article insert before et after, update before et after et delete after

4.16.3 Les procédures

- règle : la facture doit être en cours
- règle : on ne peut pas changer d'article en mise à jour
- règle : la quantité sortie doit exister en stock d'article
- règle : récupération du prix de vente en article
- calcul du montant de la sortie d'article
- calcul du montant de la facture

```
-- sortie_article$1 : calcul du montant de la facture update et insert (after)

DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;

BEGIN
```

(suite sur la page suivante)

```

v_1=(select COALESCE(sum(montant),0) from sortie_prestation
      where facture = NEW.facture);
v_2=(select COALESCE(sum(montant),0) from sortie_article
      where facture = NEW.facture);
v_3=v_1+v_2;
update facture set montant=v_3 where facture = NEW.facture;

RETURN NEW;
END;

-- sortie_article$2 sortie_article$1 calcul du montant de la facture en delete after

DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;

BEGIN
v_1=(select COALESCE(sum(montant),0) from sortie_prestation
      where facture = OLD.facture);
v_2=(select COALESCE(sum(montant),0) from sortie_article
      where facture = OLD.facture);
v_3=v_1+v_2;

update facture set montant=v_3 where facture = OLD.facture;
RETURN OLD;
END;

-- sortie_article$3 insert et update before

DECLARE
v_1 NUMERIC;
v_2 NUMERIC;
v_3 INTEGER;
v_4 INTEGER;
v_5 INTEGER;

BEGIN
v_5=(select etat from facture where facture=NEW.facture);
IF (v_5 != 1) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les sorties
d''article car l''état de la facture n est pas ''en cours''';
END IF;
-- changement article non permis
IF (TG_OP='UPDATE' AND NEW.article !=OLD.article ) THEN
    RAISE EXCEPTION 'vous ne pouvez pas changer d''article';
END IF;
IF (NEW.quantite <= 0) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas sortir
une quantité négative ou égale a 0';
ELSE
    -- recherche du prix de vente et de la quantite en stock
    v_1=(SELECT COALESCE(prix,0) FROM article
        WHERE article.article = NEW.article );
    v_2=(SELECT COALESCE(quantite,0) FROM article

```

(suite de la page précédente)

```

        WHERE article.article = NEW.article );
NEW.prix = v_1;
NEW.montant =NEW.prix *NEW.quantite;
-- sortie de stock
IF (TG_OP='INSERT' ) THEN -- trigger insert
    IF ( NEW.quantite > v_2 ) THEN
        RAISE EXCEPTION 'Vous n''avez
            pas assez de stock pour cet article';
    ELSE
        v_3=v_2-NEW.quantite;
        UPDATE article SET quantite = v_3
            WHERE article.article =NEW.article ;
    END IF;
ELSE -- trigger update
    v_4=NEW.quantite-OLD.quantite ;
    IF ( v_4 > v_2 ) THEN
        RAISE EXCEPTION 'Vous n''avez
            pas assez de stock pour cet article';
    ELSE
        v_3=v_2-NEW.quantite+OLD.quantite ;
        UPDATE article SET quantite = v_3
            WHERE article.article =NEW.article ;
    END IF;
END IF;
END IF;

RETURN NEW;
END;

-- sortie_article$4 vérification état facture avant delete

DECLARE
v_1 INTEGER;

BEGIN
v_1=(select etat from facture where facture=OLD.facture);
IF (v_1 != 1) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou
        modifier les sorties de prestation car l''état de la facture n est pas '
    ↪ 'en cours'';
END IF;

RETURN OLD;
END;
```

4.17 Sortie des prestations

4.17.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Sortie_prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues								
1 - 6 enregistrement(s) sur 6														
+	champ	pgsql	lien	obl	libelle	ajout	modif	cons	sup	vue	vue_id	pos	lst	pos
	sortie_prestation.facture	integer	prestation	Oui	facture			Visible	Visible			C1-5	Oui	5-facture
	sortie_prestation.libelle	character varying		Non	libelle			Visible	Visible			C1-1	Oui	1-libelle
	sortie_prestation.montant	numeric		Non	montant	hidden		Visible	Visible			C1-7	Oui	7-montant
	sortie_prestation.prestation	integer	prestation	Oui	prestation			Visible	Visible			C1-6	Oui	6-prestation
	sortie_prestation.prix	numeric		Non	prix	hidden		Visible	Visible			C1-2	Oui	2-prix
	sortie_prestation.quantite	integer		Oui	quantite			Visible	Visible			C1-3	Oui	3-quantite

La champs suivants sont créés

```
libelle character varying,
prix numeric,
quantite integer NOT NULL,
facture integer NOT NULL,
prestation integer NOT NULL,
montant numeric
```

Le champ montant est un champ calculé : prix x quantité.

facture et prestation sont des clés secondaires

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Sortie_prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues				
1 - 2 enregistrement(s) sur 2										
	contrainte	schema	table	champ	foreign table	foreign champ				
	sortie_prestation_facture_fkey	om5	sortie_prestation	facture	facture	facture				
	sortie_prestation_prestation_fkey	om5	sortie_prestation	prestation	prestation	prestation				

Les clés secondaires suivantes sont créées par om_forms.

```
ALTER TABLE ONLY sortie_prestation
  ADD CONSTRAINT sortie_prestation_facture_fkey
  FOREIGN KEY (facture) REFERENCES facture(facture);
ALTER TABLE ONLY sortie_prestation
  ADD CONSTRAINT sortie_prestation_prestation_fkey
  FOREIGN KEY (prestation) REFERENCES prestation(prestation);
```

4.17.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Sortie_prestation Om5

Table	Formulaire	Action	clé secondaire	Procédures	Triggers	Vues
1 - 6 enregistrement(s) sur 6						
+	nom	evenement	condition	procedure	sur	quand
	sortie_prestation\$2	UPDATE		EXECUTE PROCEDURE om5."sortie_prestation\$2"()	ROW	BEFORE
	sortie_prestation\$4	UPDATE		EXECUTE PROCEDURE om5."sortie_prestation\$1"()	ROW	AFTER
	sortie_prestation\$1	INSERT		EXECUTE PROCEDURE om5."sortie_prestation\$2"()	ROW	BEFORE
	sortie_prestation\$3	INSERT		EXECUTE PROCEDURE om5."sortie_prestation\$1"()	ROW	AFTER
	sortie_prestation\$5	DELETE		EXECUTE PROCEDURE om5."sortie_prestation\$3"()	ROW	AFTER
	sortie_prestation\$6	DELETE		EXECUTE PROCEDURE om5."sortie_prestation\$4"()	ROW	BEFORE

Il y a 5 triggers sortie_article insert before et after, update before et after et delete after

4.17.3 Les procédures

- règle : la facture doit être en cours
- règle : récupération du prix de vente en prestation
- calcul du montant de la sortie prestation
- calcul du montant de la facture

```

sortie_prestation$1 : mise a jour montant facture : insert et update after

DECLARE

v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;

BEGIN

v_1=(select sum(montant) from sortie_prestation where facture = NEW.facture);
v_2=(select sum(montant) from sortie_article where facture = NEW.facture);
v_3=v_1+v_2;
update facture set montant=v_3 where facture = NEW.facture;

RETURN NEW;
END;

-- sortie_prestation$2 : calcul montant et verification état facture ; insert et
↪update before

DECLARE
v_1 numeric;

BEGIN
v_1=(select etat from facture where facture=NEW.facture);
IF (v_1 != 1) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier
    les sorties de prestation car l'état de la facture n est pas 'en cours'';
ELSE
    NEW.prix= (select prix from prestation where prestation=NEW.prestation);

```

(suite sur la page suivante)

```

        NEW.montant =NEW.prix *NEW.quantite;
END IF;

RETURN NEW;
END;

-- sortie_prestation$3 calcul montant de la facture : delete after

DECLARE

v_1 NUMERIC;
v_2 NUMERIC;
v_3 NUMERIC;

BEGIN

v_1=(select sum(montant) from sortie_prestation where facture = OLD.facture);
v_2=(select sum(montant) from sortie_article where facture = OLD.facture);
v_3=v_1+v_2;
update facture set montant=v_3 where facture = OLD.facture;

RETURN OLD;
END;

-- sortie_prestation$4 : verification état facture en delete before

DECLARE
v_1 INTEGER;

BEGIN
v_1=(select etat from facture where facture=OLD.facture);
IF (v_1 != 1) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier
    les sorties de prestation car l'état de la facture n est pas 'en cours'
    ↪'';
END IF;

RETURN OLD;
END;

```

4.17.4 Vues

```

-- sortie_prestation$1 : utilisée comme widget dans le tableau de bord

SELECT count(prestation.prestation) AS count,
       prestation.libelle
FROM (om5.sortie_prestation
      JOIN om5.prestation ON ((sortie_prestation.prestation = prestation.
↪prestation)))
GROUP BY prestation.prestation, prestation.libelle
ORDER BY prestation.libelle;

```

4.18 Versement

4.18.1 formulaire

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Versement Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 6 enregistrement(s) sur 6

+	▶ champ	▶ pgsq	▶ lien	▶ obl	▶ libelle	▶ ajout	▶ modif	▶ cons	▶ sup	▶ vue	▶ vue_id	▶ pos	▶ lst	▶ pos
	versement.date_versement	date		Oui	date_versement			Visible	Visible			C1-2	Oui	2-date_versement
	versement.facture	integer	facture	Oui	facture			Visible	Visible			C1-5	Oui	5-facture
	versement.libelle	character varying		Oui	libellé			Visible	Visible			C1-1	Oui	1-libellé
	versement.mode_reglement	integer	mode	Non	mode_reglement			Visible	Visible			C1-4	Oui	4-mode_reglement
	versement.montant	numeric		Oui	montant			Visible	Visible			C1-3	Oui	3-montant
	versement.recouvrer	boolean		Non	recouvrer			Visible	Visible			C1-6	Oui	6-recouvrer

La champs suivants sont créés

```
libelle character varying NOT NULL,
date_versement date NOT NULL,
montant numeric NOT NULL,
mode_reglement integer,
facture integer NOT NULL,
recouvrer boolean
```

facture et mode_reglement sont des clés secondaires

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Versement Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 2 enregistrement(s) sur 2

	▶ contrainte	▶ schema	▶ table	▶ champ	▶ foreign table	▶ foreign champ
	versement_facture_fkey	om5	versement	facture	facture	facture
	versement_mode_reglement_fkey	om5	versement	mode_reglement	mode_reglement	mode_reglement

Les clés secondaires suivantes sont créées par om_forms.

```
ALTER TABLE ONLY versement
  ADD CONSTRAINT versement_facture_fkey
  FOREIGN KEY (facture) REFERENCES facture(facture);
ALTER TABLE ONLY versement
  ADD CONSTRAINT versement_mode_reglement_fkey
  FOREIGN KEY (mode_reglement) REFERENCES mode_reglement(mode_reglement);
```

4.18.2 Les triggers

Om5 No Code ➔ Gestion De Tables Et Génération Des Objets ➔ Versement Om5

Table Formulaire Action clé secondaire Procédures Triggers Vues

1 - 6 enregistrement(s) sur 6

+	nom	evenement	condition	procedure	sur	quand
	versement\$1	INSERT		EXECUTE PROCEDURE om5."versement\$1"()	ROW	BEFORE
	versement\$2	UPDATE		EXECUTE PROCEDURE om5."versement\$1"()	ROW	BEFORE
	versement\$3	INSERT		EXECUTE PROCEDURE om5."versement\$2"()	ROW	AFTER
	versement\$4	UPDATE		EXECUTE PROCEDURE om5."versement\$2"()	ROW	AFTER
	versement\$5	DELETE		EXECUTE PROCEDURE om5."versement\$3"()	ROW	AFTER
	versement\$6	DELETE		EXECUTE PROCEDURE om5."versement\$4"()	ROW	BEFORE

Il y a 6 triggers insert before et after, update before et after et delete after et before

4.18.3 Les procédures

- règle : la facture doit être en reglement
- calcul du montant versé de la facture

```
-- versement$1 : vérification facture en cours , insert et update
DECLARE
v_1 integer;

BEGIN
v_1=(select etat from facture where facture=NEW.facture);
IF (v_1 != 3) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter
    ou modifier les versements car l''état de la facture n''est pas '
    ↳'reglement'';
END IF;

RETURN NEW;
END;

-- versement$2 mise à jour du montant des versements dans facture : insert et update,
↳after

DECLARE
v_1 numeric;

BEGIN
v_1=(select sum(montant) from versement where facture = NEW.facture);
update facture set versement=v_1 where facture = NEW.facture;

RETURN NEW;
END;

versement$3 mise à jour du montant des versements dans facture : delete after
```

(suite sur la page suivante)

(suite de la page précédente)

```
DECLARE
v_1 numeric;

BEGIN
v_1=(select sum(montant) from versement where facture = OLD.facture);
update facture set versement=v_1 where facture = OLD.facture;

RETURN OLD;
END;

-- versement$4 la facture doit être en règlement , before delete

DECLARE
v_1 numeric;

BEGIN
v_1=(select etat from facture where facture=OLD.facture);
IF (v_1 != 3) THEN
    RAISE EXCEPTION 'Vous ne pouvez pas ajouter ou modifier les versements car l'
↪'état de la facture n'est pas ''reglement''';
END IF;

RETURN OLD;
END;
```


CHAPITRE 5

Bibliographie

— <http://www.openmairie.org/>

CHAPITRE 6

Contributeurs

(par ordre alphabétique)
— François Raynaud