

Présentation du SI

Introduction

Aujourd'hui notre SI ne répond que aux demandes directes de gestion d'un magasin alimentaire et d'une partie de la gestion des membres. Le système que l'on utilise à « Les Grains de Sel » est basé sur le système utilisé par « La Louve » (coopérative alimentaire du 18ème) qui est construit avec la plateforme [Odoo](#) développé sur python. Il a été complété avec des applications tierces Django (python) qui ont été développées par des membres de « La Cagette » (coopérative alimentaire de Montpellier) fournis, chez nous, par notre fournisseur de service : [Coopératic](#).

Cette présentation est légèrement spécifique à la coopérative « Les Grains de Sel » mais s'applique en grande partie aussi aux autres coopératives qui utilisent les prestations de [Cooperatic](#)

Odoo

Selon [Wikipédia](#) :

Odoo, anciennement OpenERP2 et Tiny ERP, est initialement un progiciel open-source de gestion intégré comprenant de très nombreux modules permettant de répondre à de nombreux besoins de gestion des entreprises (ERP), ou de gestion de la relation client (CRM).

Comme vous pouvez le remarquer Odoo est donc un logiciel très générique qui peut être amélioré en lui ajoutant des « modules ».

En particulier, dans le monde des coopératives alimentaires il existe deux systèmes principaux, basés sur Odoo, utilisées en France :

- [AwesomeFoodCoops](#) (AFC) : dépôt développé depuis au moins 2016 par « [La Louve](#) » et au moins 8 autres coopératives françaises. Même s'il y a plusieurs coopératives impliquées, à cause de raisons historiques et financières, le logiciel répond surtout aux besoins de « La Louve ».

- [Obeesdoo](#) : dépôt développé depuis octobre 2015 par la coopérative bruxelloise « [Beescoop](#) » maintenu par [Coop IT Easy](#). Une société appelé « [Vracoop](#) » a également basé son logiciel de gestion sur cette version en demandant de l'aide à Coop IT Easy.

Notre Odoo

Le système actuel du supermarché utilise donc la version 9 de Odoo, avec les modules [AwesomeFoodCoops](#).

Actuellement, pour de motifs hérités de notre fournisseur, [Cooperatic](#), nous sommes en retard sur la version « officielle » AFC et utilisons le code dans sa version de 2018, un des projet prioritaire est d'ailleurs de s'aligner sur la version AFC officielle.

Le groupe informatique des Grains de sel essaye de maintenir sur un fork (dépôt de code chez nous) les différentes versions actuellement concernés :

- Une [branche](#) `9.0-lgds-cooperatic` qui contient le code officiel AFC de « la Louve », à jour depuis AFC officielle (**07-2021**) et avec les derniers changements Cooperatic, qui est **la version souhaitée** de notre système ;
- Une [branche](#) `9.0-cooperatic` qui contient le code utilisé par notre prestataire, Cooperatic, qui est la version actuelle de notre système.

L'intérêt de maintenir ces deux branches est de pouvoir construire (*automatiquement*) des images docker utilisables par les développeurs et accessibles via le *docker registry* fourni par Gitlab. Les images sont ensuite utilisables en choisissant un tag parmi [les images disponibles ici](#), par exemple, en admettant vouloir l'image avec le tag `9.0-cooperatic-2021-07-16` :

```
docker pull registry.gitlab.com/lgds/foodcoops:9.0-cooperatic-2021-07-16
```

Ou alors, en admettant vouloir l'image avec le tag `9.0-lgds-cooperatic-2021-07-16` :

```
docker pull registry.gitlab.com/lgds/foodcoops:9.0-lgds-cooperatic-2021-07-16
```

La dernière version de code utilisé par Cooperatic est celle disponible directement sur leur Gitlab ici : <https://gl.cooperatic.fr/cooperatic-foodcoops/odoo>

Applications tierces

Le système dispose de petites applications Django qui communiquent avec Odoo via [l'API \(XML-RPC\)](#) qui nous permettent de gérer des besoins très concrets et de simplifier les interfaces Odoo.

En facilitant vraiment des tâches spécifiques aux coopérateurs qui ne viennent qu'une fois tous les 28 jours. Toutes ces applications ont été codées originalement à Montpellier et sont maintenues par notre fournisseur, Cooperatic. En particulier, actuellement, on dispose de :

- **Borne d'accueil** : Pour vérifier les statuts de ceux qui viennent faire leurs courses, ajouter la photo à la fiche membre et enregistrer la présence dans les services. [Tutoriels d'utilisation](#)
- **Inscriptions** : permet de pré-inscrire et d'enregistrer les nouveaux coopérateurs en plusieurs phases (avec 2 phases de validations). [Tutoriels d'utilisation \(à compléter\)](#)
- **Réception** : gère les réceptions (vérification des quantités et prix), en partant des demandes de prix Odoo.
- **Espace membre** : pour accéder aux choix et échanges de services + confirmation inscription.
- **Mouvements de stocks** : saisie des pertes, auto-consommation et récap. par périodes.
- **Inventaire** : permet d'associer les produits à des emplacements physiques du magasin et à les inventorier. [Tutoriels d'utilisation](#)
- **Outils** : pour l'instant, pour détecter les anomalies dans les code-barres.
- **Brinks** : gestion des enveloppes d'espèces et de chèques des souscriptions pour remise en banque (**non utilisé actuellement**).
- **Boutique en ligne** : permettre les commandes pour les prendre en mode drive, développé pour des raisons du confinement (**non utilisé actuellement**).

Le code de ces applications, dont la propriété et la maintenance, est organisé par [Cooperatic](#) est disponible [sur leur gitlab](#).

Le groupe informatique des Grains de sel **essaye de maintenir un fork** (*dépôt de code chez nous*) de ces applications afin de travailler en amont (*avant de pouvoir proposer des changements à Cooperatic*). Le dépôt de code des applications sur lequel nous travaillons est disponible ici :

[Applis-tierces \(si dev pour LGDS\) et Applis-tierces \(Cooperatic\)](#)

Nous essayons de centraliser les changements que l'on souhaite faire au sein des grains de sel sur une branche `dev_lgds` sur laquelle vous devrez proposer des « Merge Requests ». (Voir [le paragraphe dédié plus bas](#) si vous ne connaissez pas les merge requests)

Premier démarrage

Actuellement, nous utilisons deux images docker privées qui contiennent des données anonymisés de la base de donnée de production. Nous n'avons pas encore vérifié complètement l'anonymisation et ne préférons donc pas rendre publique ces images pour l'instant. Il vous faudra un accès au dépôt <https://gitlab.com/lgds/private-docker-images> pour

faire fonctionner les étapes ci-dessous. Vous pouvez nous contacter sur le salon public [#lgds-accueil](#) si vous avez des questions ou désirez un accès.

Pour votre premier démarrage de tout l'environnement (Odoo + Applis tierces), il vous faudra [git](#) et [docker](#) d'installé sur votre machine. Ensuite voici les étapes à réaliser :

- Vous authentifier auprès du registry docker Gitlab (*il nous sert pour stocker des images docker privées qui contiennent des données liés à LGDS que l'on ne souhaite pas partager en public*)

```
docker login registry.gitlab.com
```

- Cloner le projet [applis-tierces](#) sur votre ordinateur. (*Placez vous dans votre dossier de travail préféré avant de faire ça*)

```
git clone git@gitlab.com:lgds/applis-tierces.git
```

- Lire le fichier `README.md` du projet [applis-tierces](#) et copier les fichiers de configuration d'exemple pour votre installation (*Détails expliqués vers la fin du `README`*)
- Enfin, vous pouvez lancer tout l'environnement avec la commande suivante (*Le tout premier démarrage sera un peu long, environ 5 minutes, mais les prochains seront beaucoup plus rapide !*)

```
docker-compose up
```

- Visitez dans votre navigateur <http://127.0.0.1:8080> pour accéder aux applications tierces et <http://127.0.0.1:8069> pour accéder à Odoo (Login administrateur : `admin` / `dev`)
- Bon code ! Il y a une liste de tickets disponibles sur le board de l'application « [réception](#) » que vous pouvez regarder / vous assignez si vous vous sentez à l'aise ou juste améliorer le code en place !

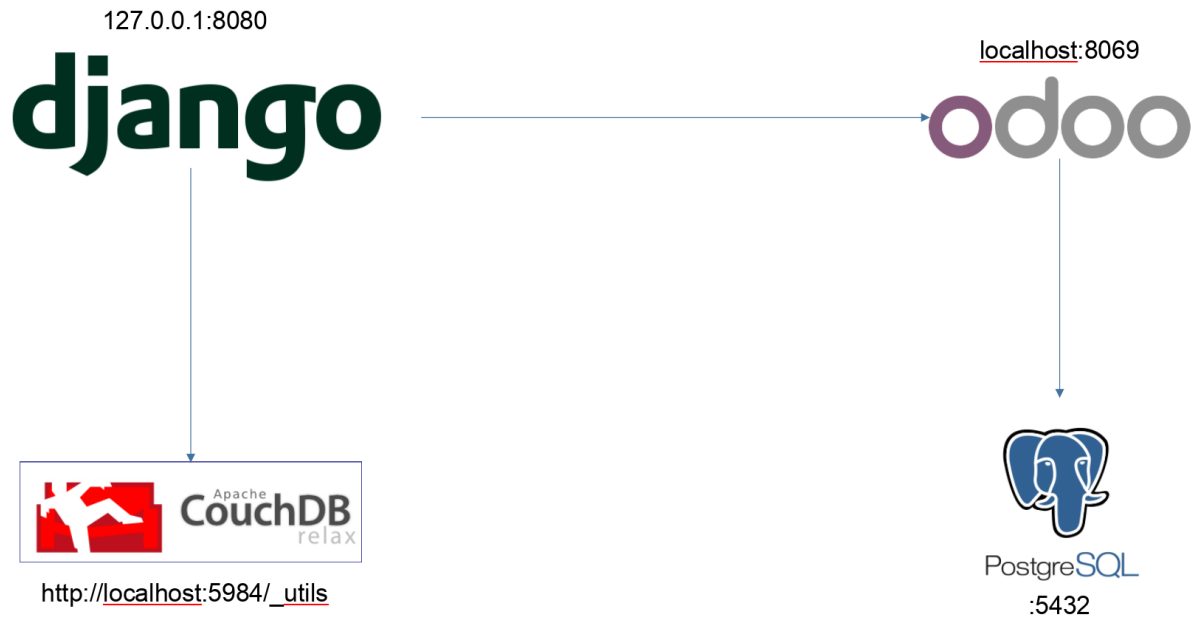
Détails sur l'architecture logicielle et les services utilisés

La commande `docker-compose up` créé et démarre les containers définis dans le fichier `docker-compose.yml` situé à la racine du projet [applis-tierces](#). Il contient les 4 services/[containers](#) suivant :

- `odoo` : le backend principal du SI ;
- `database` : Postgresql, la base de donnée d'`odoo` ;
- `app` : l'ensemble des applications tierces codés avec le framework django ;

- `couchdb` : la base de donnée de l'`app`.

On peut schématiser ces services et leurs dépendances comme suit:



N.B. Lors du 1er lancement, la `database` fera un `pg_restore` de la base de données d'Odoo (dump anonymisé créé par nos soins) ce qui aura pour conséquence de ne la rendre accessible qu'à la fin de ce processus (env 5 minutes)

Organisation du travail collaboratif

L'ensemble des tickets liés aux applications tierces sont centralisés dans le projet [gestion-informatique](#). Sentez vous libre de vous attribuer un ticket en vous mettant en tant qu'« assignee ». Sinon n'hésitez pas à demander au groupe informatique de vous en attribuer un.

De préférence, prévenez sur le salon Element [#lgds-informatique](#) que vous commencez à traiter un sujet. Chaque colonne du tableau représente un état d'avancement des différents dev que vous glisserez dans la colonne appropriée au fur et à mesure de son avancement.

Proposition de changement (Merge Request)

Pour travailler à plusieurs sur une base de code commune plusieurs choses sont à prendre en compte. Git est un élément essentiel pour favoriser cette collaboration. C'est pourquoi nous souhaitons utiliser le « pull request flow » qui est très commun dans les projets open-source.

Prenez 5-10 minutes pour lire cette article très bien écrit en français qui décrit ce flux de travail :

Guide sur le Pull (ou Merge) Request Workflow. Cela sera essentiel pour comprendre la méthode et pouvoir travailler ensemble. Dans notre cas, vous pouvez remplacer dans l'article :

- tous les mots `master` par `dev_lgds` - qui est notre branche de travail principale ;
- toutes les mentions à une « **Pull Request** » (abrégé en PR) par « **Merge Request** » (abrégé en MR). Github appelle cela une pull request, mais Gitlab - l'outil que nous utilisons - appelle cela une merge request.

Recommandations pour utiliser Git

Lors que vous faites des changements (`git commit`) essayez de prendre le temps d'écrire un message humain rattaché à vos changements de code. Un commit contient en effet, non seulement des changements de code, mais aussi **un message**. C'est un très bon réflexe à prendre que de bien rédiger ses messages de commit car cela aidera les personnes qui vont relire votre code, ou les personnes qui vont essayer de comprendre la raison de tel ou tel changement.

Exemple d'un commit qui donne beaucoup d'information : **44a623**. Si vous regardez, il contient **un titre** et **une description humaine** qui permet d'identifier plusieurs choses :

- les réflexions qui ont poussés à ce changement. Ici, c'est donner le `contexte des grains de sel`.
- un petit détail de la solution proposée. Ici, c'est `raccourcir les phrases` et `styler les boutons`.
- la partie du code qui a été touché. Ici, c'est le nom de l'appli `borne-accueil`.

Convention de nommage de la branche

Cette partie n'est qu'une suggestion

Pattern proposé `X-NomDeBranche` où :

- `X` = le numéro du ticket dans gitlab (Ex: [ce ticket](#) a le numéro 56)
- `NomDeBranche` doit être un nom court décrivant/identifiant très brièvement le contenu du dev.

Autres applications diverses

En plus des applis tierces, à Montpellier un coopérateur a développé des petites applications pour résoudre des problèmes qui impliquent des périphériques externes (imprimantes d'étiquettes et balances). Elles ont été développés sur VBA dans Microsoft Access. Toutes les applications ont un schéma similaire : transmission des données depuis un service Django sur un serveur WebDAV en format txt qui est lu et traité par l'application.

- **Impression des étiquettes rayon** : application de type *daemon* pour imprimer les étiquettes prix posés au rayon devant chaque produit. Impression automatique si le prix est changé avec l'application réception et bouton sur odoo pour lancer une étiquette particulier.
- **Impression des code-barres** : application de type *daemon* pour imprimer les autocollantes qui contient les code-barres pour ceux produits qui ne l'apportent pas par défaut. Impression automatique si les quantités sont saisis avec l'application réception.
- **Balance** : application qui contrôle les ordinateurs qui ont une balance connecté afin d'obtenir des code-barres pour les articles non conditionnés.

Schéma global

Voici un schéma global de notre SI actuel :

