POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Automate codeur d'amélioration continue : application à l'implantation d'un réseau d'entraide avec un ERP libre

MATHIEU BENOIT
Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie industriel

mars 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Automate codeur d'amélioration continue : application à l'implantation d'un réseau d'entraide avec un ERP libre

présenté par **Mathieu BENOIT**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

PTODO NTODO, présidente
Samuel BASSETTO, membre et directeur de recherche
Martin TRÉPANIER, membre
PTODO NTODO, membre externe

DÉDICACE

REMERCIEMENTS

Samuel Bassetto

Directeur de recherche en génie industriel, aide à l'amélioration continue en contexte industriel, aide dans la création de lien avec le projet d'étude de l'Accorderie et aux projets similaires.

Alexandre Benoit

Relecture

Célia Lignon

Pour la maquette du projet Espace Membre Accorderie fait en collaboration avec DOMUS de l'université de Sherbrooke.

Centre d'excellence sur le partenariat avec les patients et le public

Projet d'étude 2

Fondation Trottier

Financement

Hassan Kassi

Relecture

Marie-Michèle Poulin

Relecture

Nadia Mohamed-Azizi

Directrice du Réseau Accorderie.

Réseau Accorderie

Projet d'étude 1

Simon Montigny

Relecture

TechnoLibre

Prêt d'équipement et d'investissement en salaire pour avancer le projet ORE pour le projet d'étude.

RÉSUMÉ

ABSTRACT

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AGPL	GNU Affero General Public License
AGPLv3	GNU Affero General Public License version 3
AMD	Advanced Micro Devices
AST	Abstract Syntax Tree
CC0	No copyright reserved
CEPPP	Centre d'excellence sur le partenariat avec les patients et le public
CPU	Central processing unit
CSV	Comma-separated values
DB	Database
ERP	Progiciel de gestion intégré
i18n	Internationalisation et la localisation
JSON	JavaScript Object Notation
LCNC	Low-Code-No-Code
LGPL	GNU Lesser General Public License
MVC	Modèle-Vue-Contrôleur
NLP	Natural language processing
OCA	Odoo Community Association
ORM	Object-Relational Mapping
OSRM	Open Source Routing Machine
PEP8	Python Enhancement Proposal 8
PHP	PHP : Hypertext Preprocessor
PME	Petite et moyenne entreprise
RAM	Random-access memory
SCSS	Sass Cascading Style Sheet
SRS	Spécification des exigences logicielles
SSH	Secure Shell
SQL	Structured Query Language
USD	United States Dollar
XML	Extensible Markup Language

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Contexte et problématique

La valeur du marché des solutions ERP s'établissait autour de 40 milliards USD mondialement en 2020 [1, 2]. Le coût moyen par utilisateur, sur 5 ans, s'élevait à 9 000$, pour une PME, en 2022 [3].

Le développement de système ERP est complexe, nécessite une maintenance exigeante et le risque d'introduire des erreurs est important. Comment accélérer le développement de fonctionnalités de la plateforme ERP Odoo [1] 12 communautaire ?

La plateforme ERPLibre [4] a été créée dans l'objectif d'accélérer le développement de la plateforme Odoo 12 communautaire. Ce mémoire va mettre l'accent sur la génération de code par des techniques de rétro-ingénierie et la gestion d'une communauté dans un contexte d'un projet logiciel libre.

1.1.1 Choix de la plateforme ERP

Choisir une plateforme ERP libre peut offrir des avantages significatifs en termes de coût, de flexibilité, de sécurité, de communauté et d'indépendance. Odoo a été choisi puisqu'il répondait à ces critères, cependant quelle est la version qui offre le plus de fonctionnalité ?

En janvier 2023, les versions 9.0 à 12.0 ne sont plus supportées officiellement par la compagnie Odoo, voir tableau 1.1, mais elles le sont encore par OCA. La version 16.0 est la version stable actuelle. La démonstration commence à partir de 9.0, là où débute la divergence entre une version communautaire et entreprise.

Au printemps 2020, Odoo version 12.0 a été choisi par ERPLibre [2]. Une recherche de module par version d'Odoo a été effectué sur 11 Go de code et de données sur le projet ERPLibre version 1.5.0, voir le tableau 1.2. Ainsi, en date du 1 janvier 2023, la version 12.0 est encore le bon choix avec 2 977 modules, puisqu'il est celui qu'il a le plus de module sur les 133 répertoires gérés par ERPLibre. Cette tendance pourrait changer en 2024 selon l'évolution.

Pour obtenir les résultats du tableau 1.2, un script a été développé pour chercher la quantité de modules en cherchant dans les 133 répertoires Git [3], puis pour toutes les versions d'Odoo, pour tous les modules qui contiennent le fichier «manifest» et que ceux-ci inclus le paramètre

1. Anciennement OpenERP, ERP libre web, lien du projet https://github.com/OCA/OCB
2. Première version de ERPLibre : https://github.com/ERPLibre/ERPLibre/releases/tag/v0.1.0.
3. Logiciel de gestion de versions décentralisé

Tableau 1.1 Tableau des dates de lancement du logiciel Odoo

Légende	Version actuelle	Anciennes versions avec maintenance étendu	Anciennes versions ou fin de maintenance
Odoo version	Date de lancement	Commentaire	
6.0/6.1	octobre 2009	Première publication sous `AGPL`, premier client web	
7.0	décembre 2012		
8.0	septembre 2014	Changement de nom pour Odoo, anciennement OpenERP	
9.0	novembre 2015	Première publication des éditions «Community» sous licence `LGPLV3` et «Enterprise» sous licence propriétaire.	
10.0	octobre 2016		
11.0	octobre 2017		
12.0	octobre 2018	Version utilisé dans ERPLibre 1.5.0	
13.0	octobre 2019		
14.0	octobre 2020		
15.0	octobre 2021		
16.0	octobre 2022		

qu'il est installable, à date précédente du 1 janvier de chaque année.

Des fois, la quantité de module diminue d'une année à l'autre. Il y a création d'une nouvelle branche lors d'une nouvelle version qui est la suite de la version précédente. Par exemple, dans le tableau 1.2, la version 10.0 entre 2017 et 2018, il y a une réduction de 171 modules dans les répertoires d'entreprise, mais il y a eu seulement 4 mois pour faire le nettoyage, les méthodes de mises à jours ont évolués depuis.

De plus, les chiffres du tableau 1.2 semblent démontrer que les versions paires d'Odoo sont plus populaire que les versions impaires. Cependant la communauté d'Odoo est bien plus grosse que 133 répertoires.

Dans la section total du tableau 1.2, la section unique signifie que la somme va ignorer les doublons. En date du 1 janvier 2023, il y a au total 17 309 modules, mais 6 063 modules uniques. Ça signifie qu'il y a 11 246 modules en doublon. Hors, le code diffère d'une version à l'autre même si c'est un doublon, ils peuvent avoir des bogues ou des fonctionnalités différentes entre eux.

Tableau 1.2 Nombre de modules par version Odoo à partir du 1 janvier minuit par année sur la plateforme ERPLibre 1.5.0.

Légende	Total OCA Entreprise	Version actuelle	Anciennes versions avec maintenance étendu	Anciennes versions ou fin de maintenance

17309/10728/6581 modules à supporter le 1 janvier 2023
17465/10952/6513 modules le 15 février 2023
156/132/24 nouveaux modules en 31 jours durant janvier 2023

Odoo version	2016	2017	2018	2019	2020	2021	2022	2023
6.1	295 269 26	299 270 29	299 270 29	299 270 36	299 270 36	299 270 36	299 270 36	299 270 36
7.0	637 597 40	633 592 41	634 593 41	635 594 41	669 619 50	669 619 50	669 619 50	671 619 52
8.0	741 597 144	1092 907 185	1215 996 219	1265 1027 238	1290 1036 254	1297 1043 254	1340 1049 291	1341 1050 291
9.0	135 46 89	456 346 110	725 602 123	776 643 133	796 659 137	803 666 137	844 666 178	850 669 181
10.0		523 111 412	954 713 241	1 537 953 584	1 647 1 047 600	1 685 1 085 600	1 754 1 109 645	1 765 1 120 645
11.0			288 77 211	1 398 658 740	1 710 929 781	1 797 1 000 797	1 860 1 023 837	1 869 1 032 864
12.0				**784** **137** **647**	**1 837** **993** **844**	**2 503** **1 464** **1 039**	**2 851** **1 633** **1 218**	**2 977** **1 693** **1 284**
13.0					617 115 502	1 445 844 601	2 024 1 310 714	2 241 1 506 735
14.0						906 129 777	2 150 1 143 1 007	2 648 1 698 950
15.0							786 96 690	1 669 865 804
16.0								972 206 766
Total								
Somme	1 808 1 509 299	3 003 2 226 777	4 115 3 251 864	6 701 4 282 2 419	8 872 5 668 3 204	11 411 7 120 4 291	14 584 8 918 5 666	17 309 10 728 6 581
Support Odoo	1 808 1 509 299	2 704 1 956 748	3 182 2 388 794	4 495 2 391 2 104	5 811 3 084 2 727	6 651 3 437 3 214	7 811 4 182 3 629	7 530 4 275 3 255
Unique	1 244 1 047 197	1 995 1 435 560	2 260 1 845 415	3 214 2 172 1 042	3 927 2 610 1 317	4 676 3 080 1 596	5 452 3 572 1 880	6 063 3 980 2 083

133/72/61 répertoires de modules dans ERPLibre 1.5.0

1.1.2 Introduction Accorderie

L'Accorderie, un réseau d'entraide Québécois, était à la recherche d'une plateforme améliorée, avec des technologies plus récentes pour contrer l'effet des réseaux sociaux qui est devenu un intermédiaire intéressant pour échanger entre les membres, ainsi que d'automatiser les processus d'échange de temps, qui demande actuellement une gestion manuelle.

«Le 3 juin 2002, l'Accorderie de Québec est officiellement constituée en tant qu'organisme à but non lucratif. Sa mission : lutter contre la pauvreté et l'exclusion sociale ainsi que favoriser la mixité sociale» [5].

Nous avons décelé que le présent projet pourrait répondre à leur besoin avec l'automate programmeur qui permet de facilité, entre autres la maintenance dans le temps. De plus, la plateforme a le potentiel de leur éviter des coûts, du développement redondant et leur donner des fonctionnalités personnalisées. Ainsi, nous avons obtenu accès au code source `PHP` de la plateforme Espace Membre dont le copyright mentionne l'année 2007 par la compagnie GRF Ressource Informatique. De plus, nous avons aussi eu accès à la base de données et selon les archives, le premier échange tracé est le 1 janvier 2003. La plateforme aurait eu plusieurs mises à jour au fil du temps. Cela nous donnait un cas réel empirique, avec des données et des utilisateurs réels, pour rendre concret une plateforme d'échange de temps dont le présent mémoire fait objet.

1.1.3 Introduction CEPPP

Le CEPPP, Centre d'excellence sur le partenariat avec les patients et le public, était à la recherche d'une plateforme pour la gestion du partenariat avec les patients et le public. Le Portail des partenaires ("Portail") du Centre d'excellence sur le partenariat avec les patients et le public (CEPPP) est issu de la fusion de communautés de patients partenaires, entre autres de la Direction collaboration et partenariat patient (DCPP) de la Faculté de médecine de l'Université de Montréal, et celle du CEPPP du Centre de recherche du Centre Hospitalier de l'Université de Montréal (CR-CHUM). Le Portail est un outil qui vient soutenir les activités de recrutement et de recherche sur les pratiques de partenariat. Ils avaient donc déjà une solution [6], mais elle était incomplète, il manquait la gestion de l'anonymisation et l'interface ne répondait pas à des critères esthétiques.

Le présent projet venait pallier aux problèmes rencontrés en facilitant le développement, en accélérant ce dernier, en permettant une personnalisation et en développant l'anonymisation des données. À l'instar de l'Accorderie, cela permettait aussi un cas réel d'utilisation empirique du projet afin d'améliorer et de comprendre ce dont le générateur de code a besoin.

1.2 Définitions et concepts

1.2.1 Caractéristique de la plateforme Odoo

Internationalization et localisation

Dans un réseau d'entraide, il est nécessaire de supporter plusieurs langues pour faciliter la compréhension de l'outil informatique à l'utilisation. Pour y arriver, il existe un standard `i18n`, qui a été référencé [7], qui permet d'adapter les logiciels informatiques à plusieurs langues sur plusieurs localisations [8]. Odoo rend accessible plusieurs bibliothèques pour permettre l'extraction de chaînes de caractères au moment de l'exécution, que ce soit dans du Python, Javascript ou `XML`. Le système permet de générer un fichier «pot» qui contient ces chaînes de caractères. Pour supporter une nouvelle langue et une localisation, on copie le fichier «pot» pour faire un fichier «po» sous la nomenclature `i18n` et on peut faire la traduction ou l'adaptation linguistique. Le système peut aussi générer la langue existante pour faire un fichier «po» avec les traductions déjà connu par le système, il suffit de le mettre à jour.

Architecture `MVC`

Pour générer des modules Odoo, le générateur de code a besoin de se reposer sur l'architecture `MVC` pour permettre une séparation claire des responsabilités et une structure de code facile à maintenir. Le générateur doit ainsi générer chacune des sections de cette architecture pour faire un tout qui permet d'échanger les données entre la base de données et l'interface utilisateur.

L'architecture `MVC` permet de séparer les composantes logicielles comme suit :

1. Le modèle : représente les données et les règles de l'application. Le modèle est responsable de la manipulation des données, de leur stockage et de leur récupération.
2. La vue : représente la présentation de la donnée. La vue est responsable de l'affichage des données stockées dans le modèle à l'utilisateur final.
3. Le contrôleur : gère les interactions entre le modèle et la vue. Le contrôleur est responsable de la réception des demandes de l'utilisateur et de leur transmission au modèle, puis de la récupération des données du modèle pour les transmettre à la vue.

Il est possible de modifier une de ces composantes sans affecter les autres composantes, ce qui facilite sa maintenance.

Website builder

Le «Website builder» est un outil nécessaire dans un réseau d'entraide pour permettre une autonomie aux utilisateurs lambda [4] (augmenter l'accessibilité) dans la création et mise à jour de contenus des pages web sur le site vitrine de l'organisation. C'est un mécanisme `LCNC` dans Odoo 12 pour créer et modifier un site web multi-page par un mécanisme de «drag and drop» avec des «snippets» paramétrables. Il permet de modifier une page web en réduisant l'intervention d'un expert technique réduisant ainsi les coûts de développement.

Architecture ORM

L'utilisation de requête SQL pour communiquer avec des bases de données demandent un temps considérable au développeur à implémenter et il nécessite la programmation d'interface avec la base de données. L'utilisation d'un `ORM` permet d'augmenter la productivité, de faciliter la maintenance et augmente la sécurité d'un logiciel. L'objectif du `ORM` est de faciliter la manipulation des données stockées dans une base de données relationnelle en les représentant sous forme d'objets dans un langage de programmation orienté objet [5]. Cela permet la simplification de l'architecture en l'écrivant en langage haut niveau au lieu de directement en `SQL`, réduisant le nombre d'erreurs et facilite la maintenance.

Architecture modulaire par héritage

Le développement de système `ERP` est complexe par l'ensemble des fonctionnalités nécessaire à la gestion des procédures et ressources des organisations. Pour réduire la complexité du développement logiciel, utiliser une architecture modulaire permet la réutilisation de code et créer des relations fonctionnelles personnalisables aux contextes des organisations.

L'héritage dans Odoo 12 peut se faire de deux manières :

1. L'héritage par extension : Cela permet d'ajouter des fonctionnalités ou de modifier le comportement d'un module existant sans toucher au code source d'origine. Les nouvelles fonctionnalités peuvent être ajoutées en créant un nouveau module qui hérite du module d'origine et en y ajoutant des vues, des modèles ou des contrôleurs supplémentaires.
2. L'héritage par substitution : Cela permet de remplacer complètement le comportement d'un module existant en créant un nouveau module qui hérite du module d'origine et

4. Utilisateur semblable à la majorité dans son comportement. `https://fr.wiktionary.org/wiki/utilisateur_lambda`

5. `https://fr.wikipedia.org/wiki/Mapping_objet-relationnel`

en y modifiant les vues, les modèles ou les contrôleurs existants.

En utilisant l'architecture modulaire avec héritage dans Odoo 12, les développeurs peuvent facilement personnaliser l'application pour répondre aux besoins spécifiques de leur organisation, sans toucher au code source d'origine et sans compromettre la compatibilité avec les mises à jour futures.

Fonctionnalité du hook lors de l'installation d'un module

Au moment de l'installation d'un module Odoo, il y a des opérations qui peuvent être effectué, comme par exemple migrer des données pour les adapter à un nouveau modèle de données. Ainsi, il y a une fonctionnalité qui se nomme le hook pour «pre-install», «post-install» et «uninstall». Ce sont des méthodes qui sont exécutés soit au moment de l'installation, avant ou après l'initialisation de la plateforme, puis à la désinstallation. En «post-install», il devient possible d'exécuter du code et accéder à la plupart des fonctionnalités du `ORM` au moment d'installer un module. C'est une manière pour exécuter des scripts dans la plateforme au moment de l'installation d'un module, que ce soit en ligne de commande ou via l'application «Application» dans Odoo.

ERPLibre

Depuis la version 9.0 d'Odoo, une version communautaire et entreprise ont été créés causant une divergence sur les fonctionnalités créant le modèle d'affaires «Open Core» [6]. L'adaptation d'un `ERP` est complexe et nécessite l'intervention d'un ou des experts pour bien répondre aux besoins de l'utilisateur pour la personnalisation de la plateforme au réalité de l'organisation. Bien que l'`OCA` travaille pour rendre accessible librement ces fonctionnalités, cela vient limiter les réseau d'entraides à pouvoir se débrouiller de manière souveraine.

La plateforme ERPLibre a ainsi été créé en début 2020 encapsulant la version Odoo 12.0 sous licence AGPLv3 en offrant une alternative 100% libre, et dans le but de faciliter le déploiement et le développement pour une organisation en permettant la gestion des dépendances avec Poetry, automatisation du déploiement avec les Docker, la gestion de tous les répertoires de module 1.2 avec Git Repo [7], et plusieurs scripts pour le développement et une documentation propre pour son utilisation. Cela permet de rendre accessible à la même endroit toutes l'information nécessaire à la gestion de son ERP pour une organisation.

6. `https://fr.wikipedia.org/wiki/Open_core`
7. `https://gerrit.googlesource.com/git-repo`

1.2.2 Cadre conceptuel

Nous proposons un modèle formel de la génération de code qui guide le travail proposé dans ce mémoire. Soit C, un code informatique exécutable (qui pourrait aussi être un script interprétable), soit μ_C les méta-données du code, et soit M, une machine disposant de deux modes d'opération : M^d le mode direct, et M^i le mode inverse, voir figure 1.1. Lorsque M opère en mode direct sur μ_C, on doit obtenir C ; en opérant en mode inverse sur C, on doit obtenir μ_C.

On peut représenter symbolique ces deux processus par les équations $M^d(\mu C)$=C et M^i(C)=μC. La machine peut alors être représentée par M = $\{M^d, M^i\}$.

L'ingénierie de génération est le mode direct. La rétro-ingénierie, le mode inverse, est le processus qui consiste à examiner et à analyser un système existant pour en comprendre le fonctionnement et les spécifications. Cette boucle va permettre d'intégrer des concepts d'amélioration continue sur l'évolution du code.

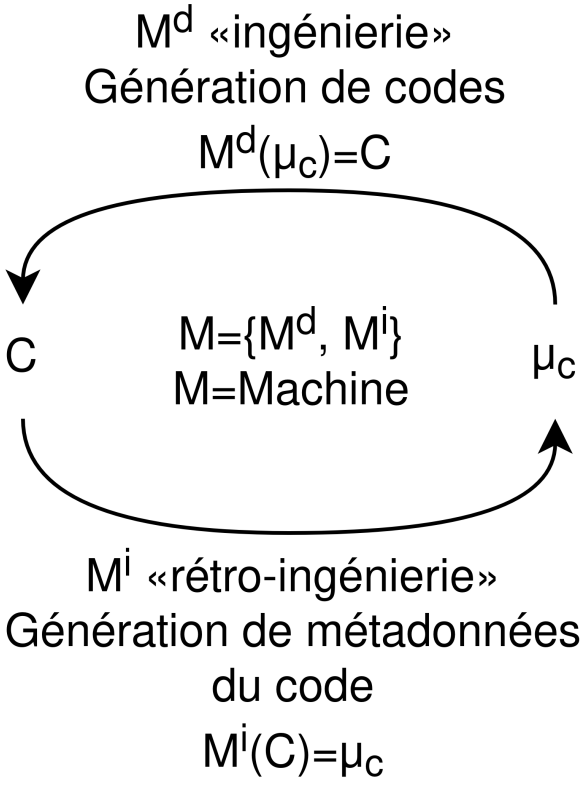


Figure 1.1 Mode direct et inverse

Pour concrétiser le sens de ce modèle formel, nous allons proposer quelques exemples simples. Ayant pour but de faciliter la compréhension, ces exemples sont triviaux et ne présentent pas

le plein potentiel de notre approche. L'interpréteur Python 3.6 et + est utilisé pour les exemples de codes.

Pour la plupart des exemples, le C, représenté par «C.py», est le code «Hello, World !», voir Code 1.1.

```
1 print("Hello, World!")
```

Code 1.1 Exemple de code Hello, World!

1.2.3 Exemples illustratifs d'auto-reproducteur

Le Quine

«Un quine [9] (ou programme auto-reproducteur, self-reproducing en anglais) est un programme informatique qui imprime son propre code source» [8] sans se lire lui-même. Il doit contenir une logique d'écriture de code et contenir ses méta-données de génération. Il est ainsi un générateur de premier niveau.

Voir l'exemple de Code 1.2, la sortie textuelle dans le console lors de l'exécution est le même que son code.

```
1 a='a=%r;print(a%%a)';print(a%a)
```

Code 1.2 Exemple de code Quine

Cependant, le Quine ne sait rien faire d'autre que de s'auto-générer. Ce qui pourrait apporter une contribution serait de faire un auto-reproducteur qui permet de dériver vers d'autres fonctionnalités et ainsi intégrer l'amélioration continue sur son propre développement.

1.2.4 Exemples illustratifs de générateur de code

La génération de code est un des moyens pour supporter le développeur dans son développement d'un logiciel. C'est la partie «création de code» de la méthodologie DevOps.

Technique de génération de code basique

Dans le code 1.3, 1.4, 1.5, la fonction «eval» en Python est dynamique, c'est-à-dire qu'il permet l'exécution à partir d'une chaîne de caractère, ce type de génération de code ne permet pas une évolution efficace ou une simplification du développement. Ça revient à lire un fichier et à l'imprimer, il n'a pas de capacité dynamique d'adaptation.

8. https://fr.wikipedia.org/wiki/Quine_(informatique)

```
print("Hello, World!")
```

Code 1.3 C

```
print('print("Hello, World!")')
```

Code 1.4 μ_C

```
eval("""print('print("Hello, World!")')""")
```

Code 1.5 M(μ_C)

Dans le code 1.6, 1.7, 1.8, cette technique basique est paramétrable, contrairement à la première exemple 1.5. Le f-strings fait office de template et il y a la capacité dynamique d'adaptation en ajoutant des itérations et des conditions sans utiliser de bibliothèque externe.

```
print("Hello, World!")
```

Code 1.6 C - fichier C.py

```
{
 "fonction": "print",
 "argument": "\"Hello, World!\""
}
```

Code 1.7 μ_C - fichier uC.json

```
import json

with open("uC.json") as f:
    metadata = json.load(f)

result = f"{metadata.get('fonction')}({metadata.get('argument')})\n"

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.8 M(μ_C)

Technique de génération de code par template

Un moteur de template est un outil de modèle structurel qui simplifie la syntaxe pour assurer une bonne maintenabilité et qui est généralement utilisé pour le développement de projet Web.

La génération de code par template est une technique de développement de logiciels qui permet de produire du code source à partir de modèles prédéfinis appelés templates.

Le code 1.9, 1.10, 1.11, 1.12 utilise la bibliothèque «Jinjer2». C'est un mécanisme similaire à code 1.8, cependant la logique est intégré directement dans le fichier template.

```
print("Hello, World!")
```

Code 1.9 C - fichier C.py

```
{
 "fonction": "print",
 "argument": "\"Hello, World!\""
}
```

Code 1.10 μ_C - fichier uC.json

```
{{ fonction }}({{ argument }})
```

Code 1.11 template - fichier template

```
import json

from jinja2 import Template

with open("template") as f:
    template = Template(f.read())

with open("uC.json") as f:
    metadata = json.load(f)

result = template.render(
    fonction=metadata.get("fonction"), argument=metadata.get("argument")
)

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.12 M(μ_C)

Les avantages du template 1.13 pour cette approche sont : une productivité accrue, une réduction des erreurs de codage, une meilleure cohérence du code et une réduction du temps de développement. Cette technique peut également faciliter la maintenance du code, puisque les modifications apportées aux templates sont automatiquement propagées à tout le code généré.

```
{% for student in students %}
  <li>
  {% if student.score > 80 %}:){% else %}:({% endif %}
```

```
  <em>{{ student.name }}:</em> {{ student.score }}/{{ max_score }}
  </li>
{% endfor %}
```

Code 1.13 Exemple de template avec logique

Générateur de code par template avec Odoo 12

La technique utilisée par Odoo 12 est le «scaffold», il gère 2 types de template : un module de base «default» et un module thème «theme».

Dans le code 1.14, tout est presque commenté, rien n'est utilisable, mais nous avons la structure `MVC`. Le gain d'accélération au développement est minime.

```
> ./odoo/odoo-bin scaffold module_default ./
> tree module_default/
controllers
    controllers.py
    __init__.py
demo
    demo.xml
__init__.py
__manifest__.py
models
    __init__.py
    models.py
security
    ir.model.access.csv
views
    templates.xml
  views.xml
```

Code 1.14 Commande Odoo pour générer un module avec le Scaffold

Cette fois-ci, dans le code 1.15, tout est vide, le module «theme_module» fait une erreur à l'installation. Il est plus efficace copier et modifier un thème existant que d'utiliser le «scaffold».

```
> /odoo/odoo-bin scaffold -t theme theme_module ./
> tree theme_module/
demo
    pages.xml
__init__.py
__manifest__.py
static
```

```
    src
        scss
            custom.scss
views
    options.xml
    snippets.xml
```

Code 1.15 Commande Odoo pour générer un thème avec le Scaffold

Le gain d'accélération au développement est minime, tellement que c'est négligeable. Copier un module fonctionnel et le modifier en enlevant ce qu'on ne veut pas est plus efficace que d'utiliser cette technique. Disons que cette technique est utile pour un débutant pour comprendre à quoi ressemble une architecture vide, mais il va apprendre plus vite en regardant le fonctionnement de module fonctionnel.

Technique de génération de code par rétro-ingénierie

Lorsqu'on veut faire de la rétro-ingénierie [10] avec un générateur le code, l'objectif est de faire de la ré-ingénierie sur la partie génération, ainsi on altère le code, voir figure 1.2.

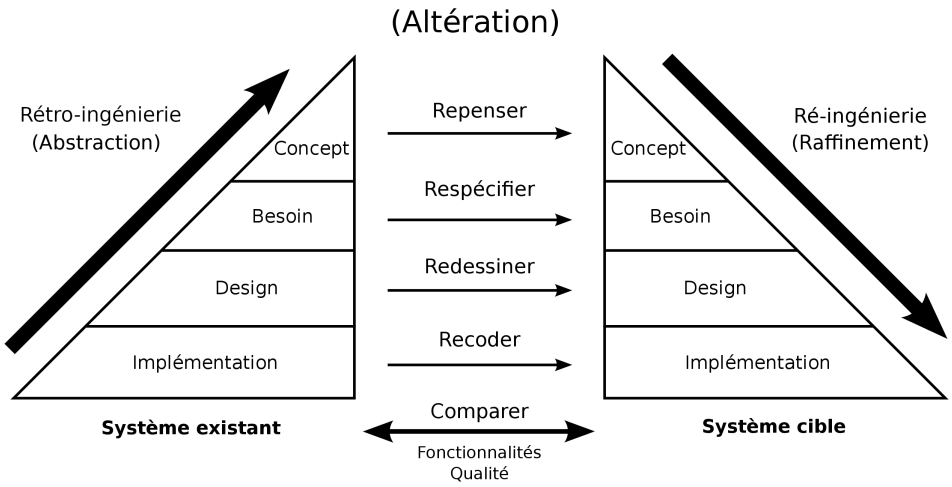


Figure 1.2 Altération du code avec la rétro-ingénierie et la ré-ingénierie. Image modifiée [11]

Dans le code 1.17, la bibliothèque AST est utilisée pour analyser l'arbre de syntaxe abstraite du code source. La difficulté avec la rétro-ingénierie est de savoir précisément ce qu'on cherche à extraire ; il faut avoir un cas d'utilisation spécifique. Ici, le cas d'utilisation est la recherche d'un nœud de type expression qui contient des paramètres. L'avantage est de permettre de corriger des problèmes de qualité logiciel entre l'extraction d'information et la génération

du code. Dans ce contexte, le résultat 1.18 du code source original 1.16 devient formaté en `PEP8 [12]`.

```
print(                        "Hello, World!"                              )
```

Code 1.16 C mal formaté - fichier C.py

```
import ast

with open("C.py", "r") as f:
    code = f.read()

# Extraction du AST
tree = ast.parse(code)
node = tree.body[0]

if (
    isinstance(node, ast.Expr)
    and isinstance(node.value, ast.Call)
    and isinstance(node.value.func, ast.Name)
):
    # Si une expression executable de type function est trouve
    fct_arg = ""
    for arg in node.value.args:
        if isinstance(arg, ast.Str):
            # Cherche un parametre
            fct_arg = arg.s
            break
    # Template
    result = f"""{node.value.func.id}("{fct_arg}")\n"""

with open("C.py", "w") as f:
    f.write(result)
```

Code 1.17 M qui extrait μ_C pour générer C.py

```
print("Hello, World!")
```

Code 1.18 C corrigé - fichier C.py

1.2.5 Tester un générateur de code

Il existe le test «Output comparison testing» [13] qui est le principe que le générateur de code crée du code (une sortie en texte lisible par l'humain) et que l'humain valide cette sortie textuel.

Pour valider si le générateur utilise sa pleine capacité, il suffit de faire des tests de toutes les combinaisons de ses techniques de génération et d'utiliser un outil de couverture de code pour déterminer les lignes qui sont opérés.

CHAPITRE 2 REVUE DE LITTÉRATURE

CHAPITRE 3 MÉTHODE

L'objectif général de ce projet a été de créer et de valider le fonctionnement d'un automate générateur de code qui sert à l'implantation d'un ERP libre. Plus spécifiquement, les travaux ont été concentré sur les cinq sous-objectifs suivants :

SO-1 développer un générateur de code de module sur Odoo,

SO-2 développer une logique d'amélioration continue sur l'écriture du code,

SO-3 développer une interface permettant de paramétrer la génération de code,

SO-4 développer un système de distribution,

SO-5 développer un système de gestion de communauté.

La méthodologie Agile a été adopté pour le développement de ces composantes d'un automate générateur de code. Les fonctionnalités de génération de code ont été validées par des tests de comparaison entre les codes générés et les codes révisés par le développeur.

3.1 SO-1 - Générateur

Développer une logique d'écriture de module sur une architecture de `MVC` avec support de plateforme web. Mise en place d'un concept de gabarit de code qui génère du code. Mise en place de la génération de code à partir de données. Générer un module à partir d'une base de données externes.

3.2 SO-2 - Rétro-ingénierie

Développer la capacité de comprendre une structure de code et de la reproduire. Définir ce que c'est de l'amélioration continue et son application dans un contexte d'automatisation. Mise en place de test de validation de code sur des critères de qualité mesurables. Intégration de règles de codage standardisées pour favoriser le réseau d'entraide.

3.3 SO-3 - Interface

Proposer une classification des techniques que l'automate peut réaliser en programmation. Développer un interface permettant le contrôle de l'automate pour l'orienter dans la programmation de fonctionnalités. Rendre disponible une interface `LCNC` pour permettre aux utilisateurs de programmer leurs fonctionnalités.

3.4 SO-4 - Déploiement

Développer un système de distribution de l'automate.

3.5 SO-5 - Réseau d'entraide

Documenter les processus de développement pour amener les utilisateurs à contribuer et les faire participer à la maintenance. Mettre des guides pour permettre le sentiment d'appartenance. Mettre en place une politique tolérance zéro avec un système de communication non violente et créer un lieu de discussion public. Élaboration du prototype pour les spécifications du réseau de l'Accorderie du Canada. Élaboration du prototype pour les spécifications de l'organisme CEPPP du Canada.

3.6 Environnement informatique

La plateforme ERPLibre 1.5.0 contenant Odoo 12 communauté, qui utilise les langages Python 3.7, `XML`, Javascript et `SCSS`. Nos tests et développement ont été effectués sur le système d'exploitation Ubuntu 20.04. Pour les temps d'exécution des tests, ils ont été effectués sur une machine avec le `CPU AMD` Ryzen 9 5950X, mémoire `ram` 2x«32GiB DIMM DDR4 Synchronous Unbuffered (Unregistered) 2667 MHz (0.4 ns)», et disque 1To wd-black-sn850-nvme. Toutes les commandes utilisées sont faites à partir de la racine du projet ERPLibre.

3.7 Méthodologie de test

Les tests ont été codés directement dans ERPLibre dans un script Python avec un ensemble de script pour valider les différences dans le Git après exécution, et cacher des différences.

Le tout a été programmé avec la technique parallélisme avec la bibliothèque Asyncio et des nouveaux processus et non des «thread» [1]. Seul le test de nouveau projet est exécuté après le parallélisme ajouter plus de temps à l'exécution.

Puisque le test d'un générateur de code consiste à valider ce qu'il a généré, ainsi vérifier la couverture de code est un bon indicateur pour déterminer le code utilisé et validé ce qui est déprécié dans le générateur.

1. Processus légé

3.7.1 Couverture du code

La couverture du code a été faite avec la bibliothèque «Coverage» version 7.0.1 en Python et configurée sur le répertoire qui contient le générateur de code pour faire le suivi des lignes exécuter dans la machine. Ça devient une métrique de la performance d'utilisation sur la quantité de fonctionnalité généré.

CHAPITRE 4 RÉSULTAT

Les résultats sont présentés en commençant d'abord par décrire l'implémentation du modèle conceptuel d'automate présenté dans la sous-section 1.2.2. Ensuite, ce sera présenté successivement les résultats propres à chacun des sous-objectifs décrits dans le chapitre 3.

4.1 Implémentation : du modèle conceptuel au modèle opérationnel

L'implémentation de la machine (conceptuelle) est passée par la programmation manuelle d'une première interface et d'un premier noyau, en mettant à jour une version préliminaire de générateur basé sur une GUI [14] en lui intégrant la capacité de générer du code à partir d'un module externe via son «hook» au moment de l'installation. La version à ce jour, ainsi que ses guides d'utilisation et ses scripts associés, sont disponibles sur le site de ERPLibre [1].

L'interface permet l'interactivité avec un utilisateur ou le système-cible. L'interface est découplée en deux ensembles de méta-données : μ_C^A et μ_C^B. μ_C^B est l'ensemble qui paramétrise le passage de méta-données au code pour un module spécifique. μ_C^A est l'ensemble qui paramétrise le passage de code aux méta-données tout en préparant un μ_C^B adapté à ce module spécifique. Ce découplage a permis l'adaptation de l'interface au contexte de l'installation de modules sur une instance Odoo via des «hooks». Par la suite, un ensemble supplémentaire de méta-données, noté μ_C^0, a été dégagé de la programmation manuelle de versions successives de μ_C^A. μ_C^0 sert à initialiser une version de départ de μ_C^A.

Le noyau prend les paramètres issus de l'interface pour créer des méta-données, générer l'ensemble des fichiers des modules désirés (mode direct) et faire de la rétro-ingénierie (mode indirect) sur des modules existants.

4.1.1 Développement et amélioration continue

Dans la figure 4.1, μ_C^0, μ_C^A, μ_C^B, C et M sont tous des modules installables sur Odoo. M^i et M^d sont des sections de code dans le module M. μ_C^0, μ_C^A et μ_C^B dépendent de M. μ_C^A, c'est les macro-méta-données, que μ_C^B, c'est les micro-méta-données.

Au départ d'un nouveau module code, μ_C^0 génère μ_C^A qui génère μ_C^B qui génère C. Il existe un script qui automatise un nouveau code, le développeur peut paramétrer le nom des modules et leurs emplacements. Ensuite, le développement commence en itération agile, les actions de

1. `https://erplibre.ca`

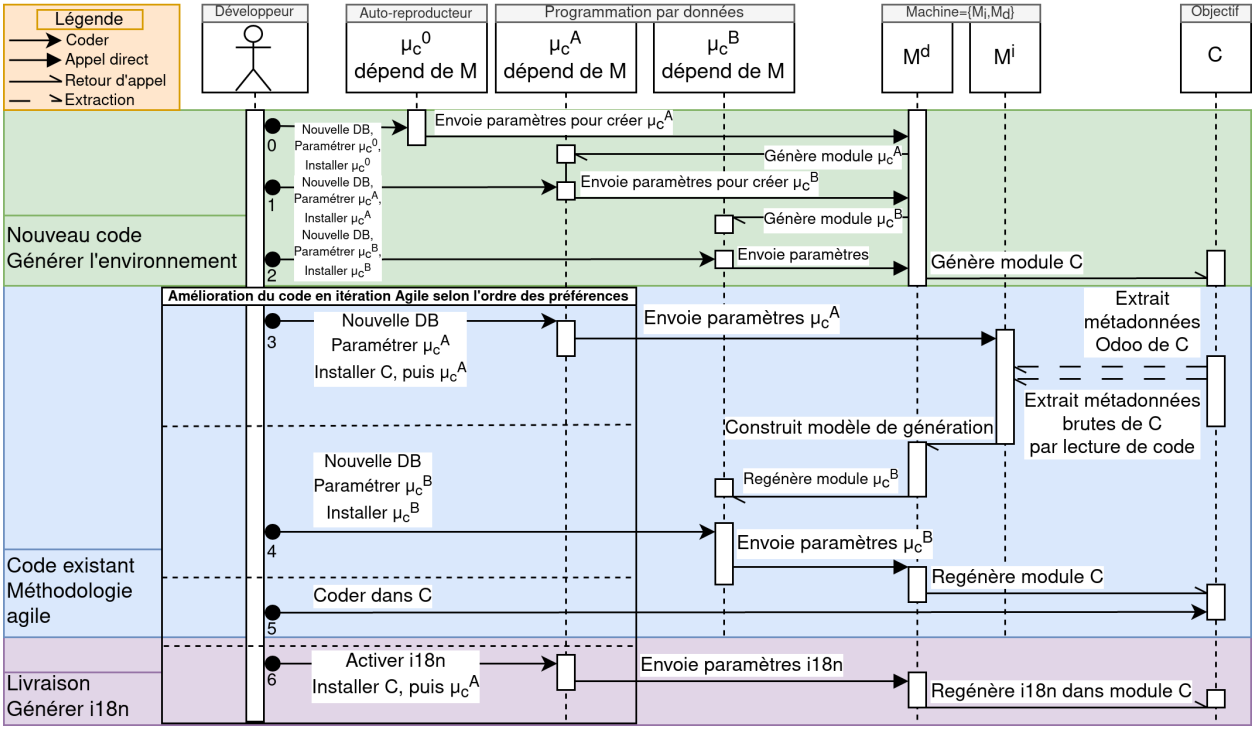


Figure 4.1 Interaction du développeur avec le générateur de code

3 à 6 peuvent être exécutées dans l'ordre du choix du développeur.

Passer par l'étape 3 permet de mettre à jour l'étape 4 selon l'état du code via la rétro-ingénierie. Passer par l'étape 4 permet de mettre à jour le code selon le générateur. Il est possible de générer de nouvelles sections, comme la vue portail. Passer à l'étape 5 permet de personnaliser le code directement. Passer par l'étape 6 permet de mettre à jour le i18n de manière automatique.

La livraison sert à générer le i18n. C'est Odoo qui le génère, mais le générateur envoie les commandes, la liste des langues désirées à supporter et place les fichiers aux bons endroits dans le module. La raison pour laquelle c'est μ_C^A qui doit le générer, c'est parce que le module doit être fini d'être généré et rechargé pour ensuite générer les langues, sinon ils sont corrompus par les traces de μ_C^B.

4.1.2 Architecture

La figure 4.2, elle démontre un développeur qui utilise l'interface de la machine qui opère dans le noyau de la machine.

1. L'interface machine permet à l'utilisateur de créer un modèle de données pour indi-

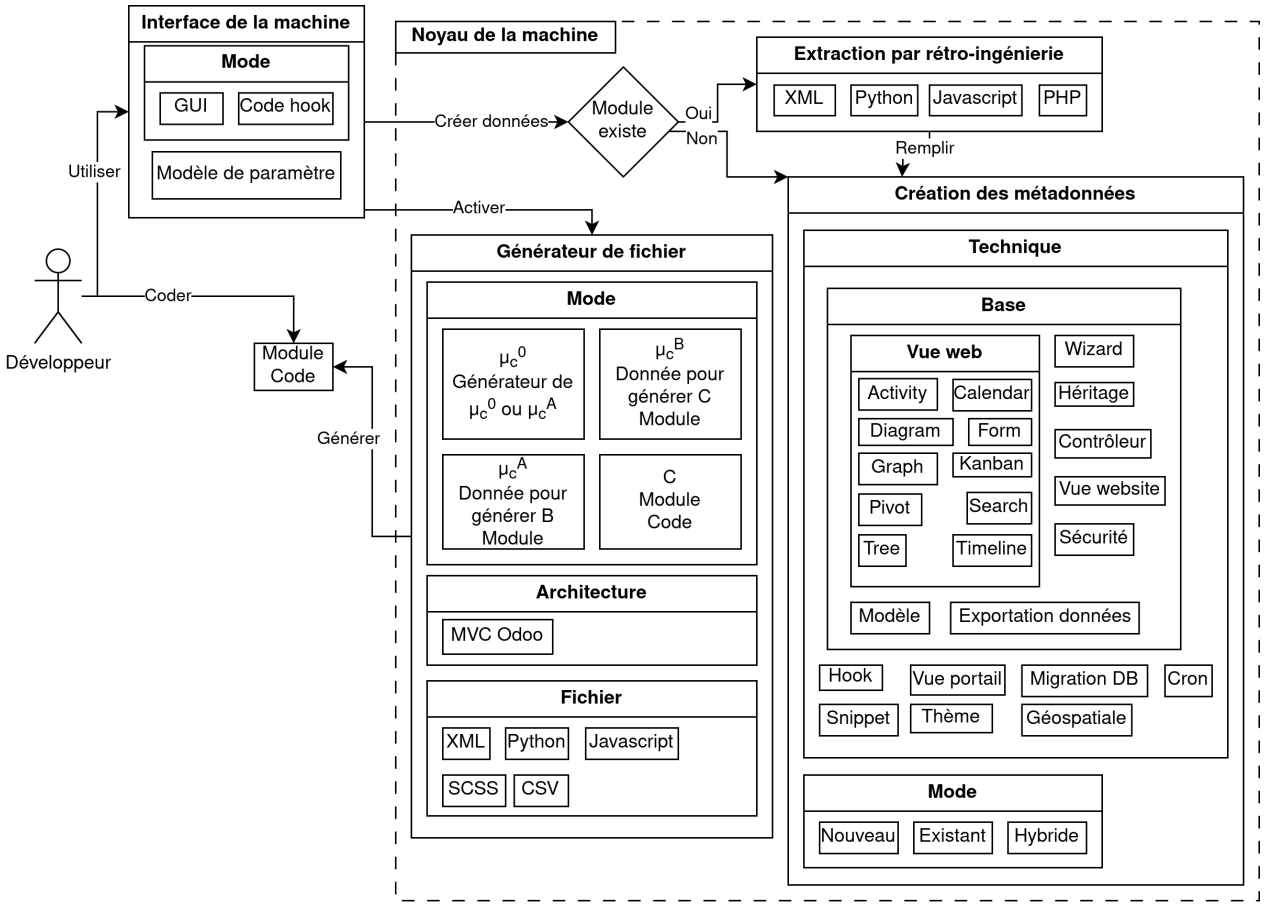


Figure 4.2 Architecture de l'automate

quer à la machine quelle opération effectuée avec leurs données associées. Plusieurs combinaisons, héritables, sont possibles selon les différentes techniques. De plus, c'est ici qu'on vient activer la génération de code.

2. L'extraction par rétro-ingénierie permet de remplir le modèle de méta-données sans passer par la paramétrisation. Il extrait des informations qui ne sont pas accessibles dans le modèle de données d'Odoo sur le module.

3. La création de méta-données se fait soit par l'utilisateur via la GUI ou le «Code Hook», il gère à la fois plusieurs techniques qui sont dans des modules. Le mode «nouveau» permet de créer de nouvelles données. Une fois qu'ils sont créés, c'est le mode «existant» qui est utilisé. Cependant le mode «hybride» permet d'écraser les données existantes en réactivant le mode «nouveau».

4. Le générateur de fichier se fait activer par l'interface, mais il prend les méta-données pour faire sa génération selon le mode qu'il doit générer et l'architecture qu'il connaît. Il fait des liaisons entre les modèles et les vues en référence aux noms des champs de

chaque modèle de données.

Chacun de ces blocs de l'architecture est modulaire, chaque technique est héritable pour modifier le comportement et ajouter des liaisons pour permettre une génération de code au final.

La sécurité dépend du modèle. Le contrôleur dépend du modèle. La vue dépend du contrôleur et du modèle.

4.1.3 Auto-générateur

Représenté par μ_C^0, voir Annexe L, c'est un auto-générateur ! Au moment de son installation, il génère le même code que lui-même au même endroit dans le système de fichier. Une légère modification va créer une autre entité qui sera une déviation dans l'objectif de démarrer une chaîne.

C'est le module `M` qui contient les méta-données de μ_C^0. Ainsi, exécuter μ_C^0 devient un test de fonctionnalité et c'est un succès lorsqu'il n'y a pas de différence. Cependant sa programmation est actuellement spécifique à sa génération, aucun autre module n'a besoin de cette fonctionnalité unique.

L'auto-générateur est utilisé pour générer des μ_C^A avec une légère modification dans les paramètres. Même s'il a la capacité de générer un μ_C^B, mieux vaut créer la chaîne proposé pour faire de l'amélioration continue.

4.2 Résultats propres à SO-1

4.2.1 Génération par gabarit

La génération par gabarit était déjà supportée dans la version initiale [14], de plus, il y a eu des améliorations :

1. Utilisation des f-strings au lieu d'utiliser la fonction «format» de String.
2. Utilisation de la bibliothèque Code-writer en Python [2]
3. Utilisation de la bibliothèque lxml [3]

2. `https://pypi.org/project/code-writer/`
3. `https://pypi.org/project/lxml/`

4.2.2 MVC

L'architecture MVC était déjà supportée dans la version initiale [14], de plus, il y a eu des améliorations :

1. Ajout de bouton qui ouvre des «Wizard» pour générer les «Views», les «Models» et les «Controllers». Ainsi le développeur peut les configurer et demander de générer les méta-données associées.

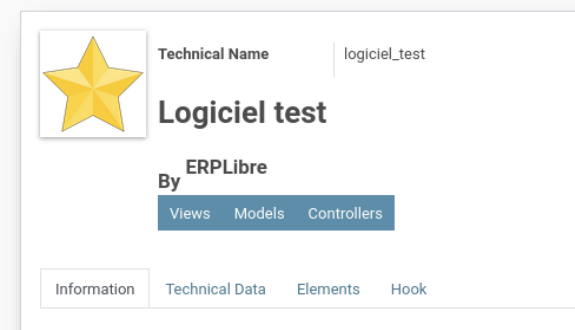


Figure 4.3 Architecture de l'automate

2. Les règles de sécurité sont ajustées selon les configurations et personnalisables par la suite.

4.2.3 Générer un module à partir d'une base de données externes

La migration de données à partir de `SQL` était déjà supporté dans la version initiale [14], cependant il y a eu des améliorations :

1. Ajout de type de données, dont ceux utilisé par le projet Accorderie
2. Ajout d'association entre les types de données et les différentes personnalisations de l'architecture Odoo
3. Interface représentant la base de données avec les contrôleurs permettant la configuration de la migration sur le modèle de données désirés
4. Gestion des interdépendances (A de B et B de A)

4.2.4 Génération de code par des données

La capacité de prendre les données via les interfaces utilisateurs telles que la GUI et le «Code hook» est de la génération de code par des données. Cela permet de personnaliser pour obtenir un logiciel adapté à ce que l'utilisateur est capable d'exprimer.

L'automate est une machine qui grâce à son interface «code hook», il se fait commander par deux couches de méta-données paramétrable par l'humain. Les deux couches s'interfèrent entre eux pour permettre l'évolution de la fonctionnalité désirée.

4.3 Résultats propres à SO-2

4.3.1 Extraction de code et reproduction

De la macro et micro extraction a été réalisé avec plusieurs techniques combinées.

Pour pouvoir faire de la reproduction, il a suffit de faire de la macro extraction, c'est-à -dire faire une recherche dans toutes les classes pour copier le contenu de chaque méthode pour le transformer en méta-données et pouvoir faire l'opération direct de générer le code qui a été copié. L'utilisation de l'`AST` a servi à déterminer quelle ligne de code était à découper pour les recopier.

Cependant il était nécessaire dans certain contexte de faire de la micro extraction, comme :

1. L'extraction des noms des constantes, ils sont transformés en valeur lors de l'exécution ;
2. L'extraction des commentaires, qui n'est pas supporté dans la bibliothèque `AST` de Python 3.7 ;
3. L'extraction des décorateurs ;
4. L'extraction des paramètres et nom des méthodes/fonctions ;
5. Les méta-informations d'un modèle (description, nom, etc.)

De plus, les vues ont été extraites dans le but d'obtenir des méta-données spécifiques qui caractérise la reconstruction de la vue, ces données n'étaient pas accessibles dans les données d'Odoo.

Certaines informations ont été extraites dans le Javascript à l'aide de la bibliothèque «pyjs-parser».

Pour l'extraction d'un projet externe d'une autre technologie, un extracteur de PHP a été développé via un «parser» de la communauté [4].

C'est en développent les techniques de génération de code qu'on réalise la reproduction. Un script a été développé pour accélérer l'écriture du générateur de code, ainsi un générateur de générateur de code. Ensuite le développeur peut le transformer légèrement pour prendre les paramètres des méta-données.

4. `https://github.com/JameelNabbo/PHP-Parsers`

4.3.2 Amélioration continue sur la génération

Grâce à l'extraction des méta-données, dès que la technique de génération est bien développé avec les méta-données, le code est automatiquement généré avec des bonnes pratiques logicielles corrigeant automatiquement les problèmes.

L'intérêt de passer par Odoo pour lire le module valide déjà certains fonctionnements (par exemple : pas d'erreur de syntaxe dans le Python ou les `XML` étaient bien construits).

Ainsi, pour un module désiré, nous utilisons les outils pour générer μ_C^A et μ_C^B, puis en avançant dans le développement, nous bouclons entre le 3 et 4 sur Figure 4.1.

4.3.3 Test de validation de génération de codes

Pour tester ce générateur de code, la technique du test de comparaisons des sorties de la génération a été utilisé. Pour procéder, un développeur valide via l'outil Git ce qui est commité [5]. Ainsi, un script a été développé pour lancer en parallèle les tests et valider les différences de génération avec ce qui a été commité précédemment. Aucune différence est un succès.

Il y a plusieurs types de test :

1. Valider l'installation du module généré ;
2. Valider que μ_C^B génère bien le module cible sans différence dans le code ;
3. Valider que μ_C^A génère bien μ_C^B sans différence dans le code ;
4. L'extraction des paramètres et nom des méthodes/fonctions ;
5. Valider que la migration d'une base de données `SQL` se fait sans différence dans le code.

En exécutant tous les tests, voir annexe E, une couverture de 84% est obtenu, tous les tests présents sont un succès, sauf ceux sur l'auto-générateur.

4.3.4 Règles de codage standardisées

Au moment de générer les fichiers, toutes les sorties textes sont traitées par des outils de mise en forme suivant des règles de codage standardisées.

Pour le Python, l'outil «black» est utilisé pour la mise en forme, suivant le standard `PEP8` avec «isort» pour réordonner les importations. «black» donne une mise en forme non naturel comparé à l'écriture de code pour un humain, cependant son résultat facilite la lecture et le suivi des différences pour les futurs ajouts.

Le Javascript, le HTML et le `XML` sont mises en forme avec l'outil «prettier».

5. Un terme dans l'outil Git pour valider le code en créant un état dans l'historique.

De plus, le générateur force le déplacement des classes dans leur propre fichier respectif pour faire une classe par fichier.

Les champs pour chaque modèle sont déplacés en ordre alphabétique, mais le premier est celui qui est utilisé pour représenter le modèle [6].

4.4 Résultats propres à SO-3

4.4.1 Classification des techniques développés

En référence à la figure 4.2, les techniques «Modèle», «Form», «Tree», «Contrôleur» et «Migration DB» étaient déjà implémentés dans la version initiale [14], mais ils ont reçu des améliorations pour s'agencer aux autres techniques.

Les techniques :

1. Contrôleur ;
2. Cron ;
3. Exportation des données ;
4. Géospatiale (dépend de Modèle) ;
5. Héritage ;
6. Hook ;
7. Migration DB [7] ;
8. Modèle ;
9. Portal ;
10. Sécurité ;
11. Snippet ;
12. Thème ;
13. Vue web ;
    (a) Activity ;
    (b) Calendar ;
    (c) Diagram ;
    (d) Form ;
    (e) Graph ;

6. Référence à l'attribut «_rec_name»
7. importation des données par DB

   (f) Kanban ;
   (g) Pivot ;
   (h) Search ;
   (i) Timeline ;
   (j) Tree ;
14. «website_leaflet» (dépend de Snippet et Géospatiale) ;
15. Wizard ;

4.4.2 Interface du générateur de code

L'interface graphique existait déjà dans la version initiale [14], elle a été améliorée pour afficher plus d'informations par rapport aux développements. Elle sert à faciliter la paramétrisation du générateur de code. Elle n'a pas été priorisée et elle manque de fonctionnalité comparé à ce qui peut être supporté via la technique «code hooks» avec μ_C^A et μ_C^B.

Ce qui fonctionne :

1. Créer module ;
2. Renommer un module ;
3. Ajouter des modèles, voir Annexe A, et des champs, voir Annexe B ;
4. Ajouter des menus ;
5. Ajouter la sécurité ;
6. Changer les icônes ;
7. Changer les informations sur les propriétés «manifest» du module ;
8. Ajouter du code, voir Annexe C ;
9. Modification des «hooks», voir Annexe D ;
10. Etc.

L'interface «code hook» permettent d'accéder à la totalité des fonctionnalités de l'automate via μ_C^A et μ_C^B, elles ont été utilisés pour toutes les démonstrations qui servent de test, elles contiennent la paramétrisation pour des modules désirés.

4.5 Résultats propres à SO-4

4.5.1 Utilisation d'un conteneur Docker

Puisque l'automate fait partie de ERPLibre, la version 1.5.0 contient les modules de génération de code. Le déploiement se fait rapidement en utilisant le logiciel Docker, le générateur de code permet l'utilisation de l'interface graphique pour générer des modules Odoo.

4.6 Résultats propres à SO-5

4.6.1 Guide créer une communauté autour d'une technologie pour un réseau d'entraide libre

Un guide hybride a été produit pour comprendre les aspects cités du démarrage d'un projet, de gestion d'une communauté autour d'un projet libre et des règles d'hébergement libres.

4.6.2 Démarrage d'un projet

Le guide du tableau 4.1 permet de démarrer rapidement un projet et s'assurer que les membres impliqués du réseau d'entraide comprennent les mêmes enjeux et s'aligne dans la même direction.

Tableau 4.1 Les 7 étapes pour démarrer un projet dans un réseau d'entraide

Étape	Description
Mission	Trouver votre mission, vos indicateurs et les objectifs associés.
Processus	Déterminer les étapes pour du développement informatique, de l'assemblage des travaux, des méthodes pour faire des services et de l'amélioration continue. Avoir conscience des gaspillages
Connexion	Mécanisme d'animation du suivi des tâches, des services.
Valeurs	Trouver les valeurs qui vont guider la façon de gérer l'équipe, la communauté, sans être exclusifs ou figées dans le temps. Ils sont un point de repère et aide pour prendre les grandes orientations.
Vision	Détailler un plan stratégique pour la communauté. C'est une projection dans le futur pour permettre de comprendre la direction sur la longue durée.
Prochaines étapes	Passer à l'action en mode itératif avec des méthodologies agiles.

4.6.3 Intégration d'un membre

1. Amener les utilisateurs à faire des contributions pour ensuite qu'ils participent à la maintenance en facilitant chaque étape ;
2. Rendre disponible des tâches pour les nouveaux membres ;
   (a) Permettre d'utiliser des étiquettes de classement adaptés sur des initiatives proposé par des nouveaux, tels que «suggestion», «problème» ou «question».
3. Remercier la personne pour son intérêt qui veut participer au projet ;
4. Répondre en moins de 24 heures pour accueillir le membre ;
5. Définir les types de contributions nécessaires et la manière qu'on examine une contribution ;
6. Mettre en place un sentiment d'appartenance :
   (a) Lorsqu'un problème est reporté, demander gentiment s'il peut avoir une contribution ;
   (b) Mettre la liste des contributeurs dans un fichier du projet ;
   (c) Remercier les contributeurs dans une infolettre.
7. Émettre des dates de rencontres officielles pour parler du projet par vidéo-conférence pour des communautés locales, qui ont la même langue.

4.6.4 Comportement en communauté

1. Proposer un guide sur les comportements désirés ;
2. Réagir publiquement pour chaque message sur la plateforme ;
3. Encourager de publier les notes de réunions et même le menus commandés des repas pour promouvoir la transparence ;
4. Développer une culture de développement ouvert.

4.6.5 Outils de développement public

1. Mettre en place un site web de développement en lien avec l'organisation créé ;
2. Documenter publiquement le processus de développement ;
3. Permettre de voir l'avancement des tâches en lien avec les processus ;
   (a) Permettre de proposer des changements avec un système d'acceptation par les pairs.

4. Montrer la feuille de route du projet, des livrables prévues ;
5. Encourager la publication du travail brouillon avec un état de travail en progression ;
6. Déployer un moyen de discussion public et éviter de répondre en privé.

4.6.6 Résolution de problème

1. Mise en place d'un arbre décisionnel avec description des décisions sur un type de problème ;
2. Documenter la résolution d'un problème de développement logiciel ;
3. Permettre aux développeurs de prendre des décisions sur des choix impopulaires basés sur leur ressentiment ;
4. Éviter les débats réguliers sur des aspects triviaux ;
5. Concentrer les discussions vers la résolution d'un problème qui mène vers une action.
   (a) Quel serait la prochaine étape à prendre ?
   (b) Suggérer des conditions pour de nouveaux progrès, offrir un itinéraire, un chemin à suivre pour obtenir les résultats désirés.

4.6.7 Documentation

1. Rendre accessible les documentations :
   (a) Fonctionnelles pour les utilisateurs ;
   (b) Technique pour le développement ;
   (c) Hébergement pour le déploiement ;
   (d) Fichier «README» pour l'utilisation rapide du logiciel ;
   (e) Fichier «CONTRIBUTE» pour comment faire de la contribution ;
   (f) Fichier «GOVERNANCE» pour le départage décisionnel.

4.6.8 Sécurité

1. Montrer le niveau de sécurité de l'application ;
2. Mettre en place un système de communication des mises à jours nécessaires ;
3. Informer comment sécuriser les clés d'authentification, les données et les configurations personnelles.

4.6.9 Développement libre

1. Suivre les règles d'hébergement de logiciel libre pour permettre l'inclusion ;
2. Expliquer l'importance du choix de la licence libre ainsi que les différences. Expliquer pourquoi d'autres choix ne sont pas proposés et restreindre l'utilisation de logiciel libre si la licence n'est pas acceptée ;
3. Rendre accessible la documentation sur le logiciel libre en lien avec la localité administrative de l'organisation [8] ;
4. Toujours proposer des licences libres avec un guide explicatif, ne pas permettre d'utiliser des licences non compatible au libre :
   (a) AGPL [9] ;
   (b) CC0 [10] ;
   (c) LiLiQ-R+ [11] ;

4.6.10 Communication

1. Faire un suivi des émotions/sentiments des membres lors des réunions (hebdomadaires) par rapport au projet ;
2. Mettre en place des outils favorisant la communication non violente.

4.7 Projet diverse

4.7.1 Projet module «auto_backup»

Le module «auto_backup» est le premier module de la communauté à avoir été testé dans ce projet, un μ_C^A et μ_C^B a été généré et de la qualité a été appliqué causant une différence avec la version communautaire dans l'organisation `OCA` et leur répertoire «server-tools».

Ainsi, la technique de gestion des «Cron», puisqu'il lance des sauvegardes par `SSH` ou en local à des moments spécifiques dans le temps.

8. Exemple, un document du Québec : `https://www.tresor.gouv.qc.ca/fileadmin/PDF/ressources_informationnelles/logiciels_libres/ll.pdf`
9. `https://www.gnu.org/licenses/agpl-3.0.en.html`
10. `https://creativecommons.org/share-your-work/public-domain/cc0/`
11. Licence libre restrictive au Québec : `https://forge.gouv.qc.ca/licence/`

4.7.2 Projet module workflow design

C'est un module de gestion de projet généré par l'automate qui permet de faire le suivi sur les opportunités, les menaces, les forces, les faiblesses et les objectifs. Il a été utilisé dans le projet Accorderie.

4.7.3 Projet module STARS

Le module STARS dépend de l'application Projet, il permet de configurer une procédure associable à un nouveau projet pour suivre les étapes de STARS, voir Figure 4.4.

Ainsi, on peut créer un nouveau projet et suivre cette procédure en ajoutant des tâches d'amélioration continue de son organisation, voir Figure 4.5.

4.7.4 Projet module SRS

C'est un module de gestion de projet généré par le générateur de code qui permet de faire l'analyse des besoins pour ensuite passer à l'analyse fonctionnelle et finalement définir les requis fonctionnelles d'un projet. Il a été utilisé en autre pour le projet Accorderie et le projet Portail CEPPP.

4.8 Projet espace Accorderie

Le projet a débuté par l'élaboration d'une analyse des besoins et fonctionnelles, puis un ensemble de requis logicielles ont été rédigé avec un membre du Réseau de l'Accorderie.

Le générateur de code a permis créer un module Odoo 12.0 avec leurs modèles de données basé sur leur base de données en `SQL` de MariadB, voir Annexe F.

Plusieurs corrections ont été effectué avant la migration : correction des noms des champs pour les uniformisés ; correction des types de champs (exemple le «True» était exprimé par la valeur «-1» dans un type «int», ainsi ce type a été transformé en booléen) ; enlever les

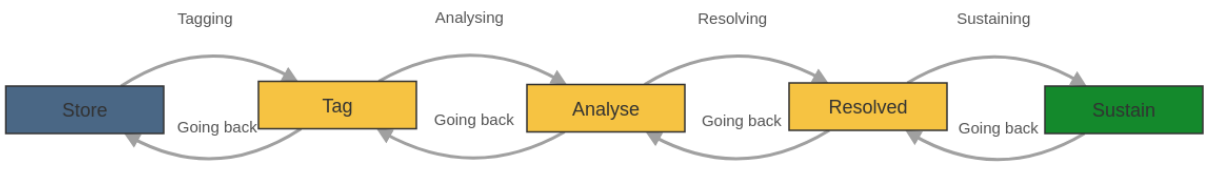


Figure 4.4 Procédure STARS dans l'application Projet vue Diagramme

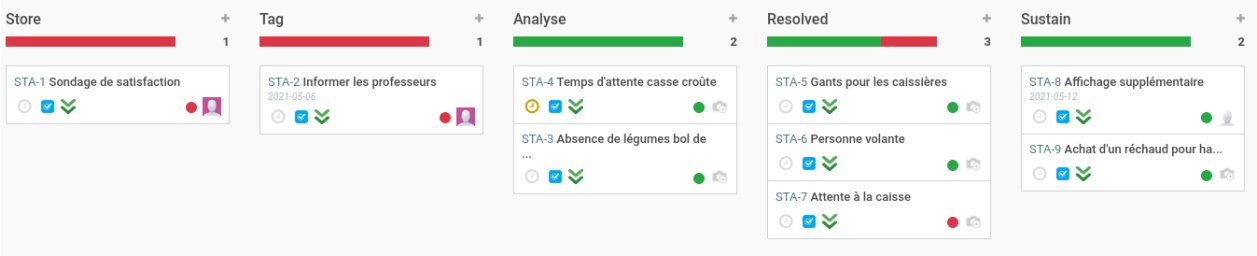


Figure 4.5 Suivi des tâches de projet avec procédure STARS en vue Kanban

double dépendances par changement de l'architecture ; correction des données erronés (un champs est requis, mais il manque des données pour certaines entrées). De plus, le modèle de données n'a pas été conçue pour de l'automatisation, mais plutôt pour que les échanges de service soient validés par des membres de la communauté.

Dans l'Annexe G, on peut observer les adaptations des champs. Par exemple, avec la table «accorderie_echange_service», il y a l'ajout des champs : «nb_heure_estime» pour avoir une prévision des heures à effectuer, «nb_heure_dure_trajet» pour reconnaître le temps de déplacement, «distance_trajet» pour connaître la distance qui sera calculé avec le projet libre `OSRM`, etc...

La migration du modèle de données a été faite dans un module qui dépend de la technique «Migration DB» qui permet d'importer le modèle. Certaines données nécessitent la création d'un fichier de données `XML`. Pour les autres données, un autre module a été créé pour ajouter les données directement dans une base de données qui sera migré vers une mise en production.

Un portail a été généré, pour remplacer les formulaires utilisés par l'ancienne plateforme `PHP` et ainsi visualiser les entrées. Cependant, cette fonctionnalité a été abandonné puisque cette technologie ne plaisait pas

Une maquette a été conçue pour un nouvel espace membre. Ainsi, on a utilisé la technique «website_snippet» pour afficher des données sur le site web et créer des formulaires. Cette base a permis d'accélérer la création de code de communication entre le client et le serveur. À force de faire l'intégration et la personnalisation de cette maquette, il n'y a plus vraiment de code qui provient du générateur de code.

Le générateur de code a permis d'aider à créer un diagramme pour afficher le processus d'échange de temps, voir Annexe H, puis une application en Javascript avec AngularJS a été développé pour afficher ce processus à l'utilisateur, une machine à état qui permet de revenir selon des paramètres à un état du processus.

Dû à des limitations humaines et de temps, tout le reste du projet a dû être fait manuellement

puisqu'on a changé de technologie pour faciliter le développement de l'interface.

4.9 Projet Portail CEPPP

L'objectif est de faire une section portail pour les patients et une section administratif pour les recruteurs, les partenaires et les administrateurs de la plateforme, rendre accessible des formulaires et anonymiser les données. Le mandat était de migrer les fonctionnalités de la plateforme qui a été développé sur SuiteCRM en `PHP`.

Un module d'extraction de `PHP` a été développé, mais il n'est pas accessible dans les techniques du générateur de code. Le modèle de données était directement dans le code puisqu'il est dynamique, il n'est pas dans la base de données. La base de données n'a pas été extraite, les données ont été exporté en `CSV` et un module d'exportation des données a été développé.

Extraction :

1. 23 fichiers analysés ;
2. 2851 données extraites ;
3. Capacité de faire la traduction anglais-français sur les données, mais cette fonctionnalité annulé puisque les données ont été modifié dans la ré-ingénierie.

Voici les statistiques 4.2 du code après ré-ingénierie et adaptation des fonctionnalités, livraison de la plateforme en début septembre 2023 [12].

Tableau 4.2 L'évolution entre la génération et la ré-ingénierie des statistiques sur les langages du portail CEPPP

Langage	# Ligne extrait	# Ligne personnalisé	# Diff
`XML`	6 861	3 856	- 3 005
Python	567	1 564	+ 997
Javascript	0	68	+ 68
`CSV`	25	51	+ 26

L'anonymisation n'est pas supporté par le générateur de code, puis la personnalisation enlève beaucoup de champs mis de manière générique dans les fichier `XML`.

Le modèle de données du portail CEPPP dans Odoo 12 contient 24 modèles I. L'interface administrateur J contient la fiche du patient, que les partenaires ont accès seulement qu'à la partie anonymisée K.

12. `https://portailppp.ca`

CHAPITRE 5 DISCUSSION

5.1 Interprétation des résultats

Les résultats obtenus ont permis d'atteindre en tout ou en partie l'ensemble des sous-objectifs énoncés dans le chapitre 3.

SO-1 Accomplissements Simplification de l'écriture de technique dans le générateur de code avec l'utilisation de «f-string», de la bibliothèque «Code-writer» et de la bibliothèque «lxml».
Ajout de «Wizard» pour configurer le `MVC` selon des paramètres.
Amélioration de l'importation des bases de données externes.
Support de la génération de code par des données.
Augmentation de technique de génération de code, support des templates «Qweb», ajout du type de données géospatiale.

SO-1 Feuille de route Implémenter les fonctionnalités manquantes dans la GUI pour générer les MVC d'un module.
Augmenter le nombre de technologies à supporter l'importation des données.
Ajouter le support de génération sur différentes architectures.
Il manque des techniques de sécurité à personnaliser, comme l'anonymisation des données.
Générer automatiquement une documentation sur l'utilisation d'une technique.
Supporter la génération sur d'autres systèmes `ERP` libres tel que Tryton [1].

SO-2 Accomplissements Extraction du code via l'utilisation d'un `AST` et extraction des méta-données dans les fichiers `XML`.
Amélioration continue sur la génération de code grâce à la reproduction à l'aide de l'extraction du code.
Un outil pour aider à la création de technique de génération à l'aide d'un générateur de générateur de code.
Le générateur de code est accompagné de tests de validation en reproduisant l'ensemble des techniques en démonstration.
La génération de code applique des règles de codage standardisées.

1. `https://www.tryton.org`

SO-2 Feuille de route Finaliser l'implémentation de l'auto génération sur l'automate.
Mise à jour des tests pour atteindre une couverture de code à 100%.
Ajout de test sur les techniques d'extractions de code tel que le `PHP`.

SO-3 Accomplissements Ajout de nouvelles techniques et une classification de ceux-ci.
Rendre accessible une interface graphique pour paramétrer la génération de code.
Rendre accessible une interface de programmation pour utiliser toutes les fonctionnalités de l'automate.

SO-3 Feuille de route Ajout de paramètres pour faire plus de personnalisation sur les techniques et les séparer une par module.
Supporter les fonctionnalités manquantes sur toutes les techniques pour l'interface graphique.
Support l'accès à la création de méta-données par la rétro-ingénierie via l'interface graphique.

SO-4 Accomplissements Intégration dans un système de distribution via un Docker.

SO-4 Feuille de route Développer une synchronisation entre les instances pour permettre de la redondance.
Développer une gestion de son infrastructure via le générateur de code.
Faire participer l'automate à la maintenance de l'infrastructure de déploiement.
Utiliser d'autres systèmes de conteneur en distribution qui sont libres comme «Pod» [2].
L'automate doit avoir la capacité de valider techniquement si le logiciel est `AGPLv3` au moment de l'exécution

SO-5 Accomplissements Test de la génération sur un module existant de la communauté nommé «auto_backup».
Des modules de gestion de projet ont été générés pour faire le suivi de la conception fonctionnelle et de l'amélioration continue.
Le projet Accorderie a bénéficié du générateur de code pour la migration de la base de données vers Odoo.
Le projet Portail CEPPP a bénéficié du générateur de code pour la migration du code `PHP` vers Odoo, ainsi que de l'aide au développement de la section Portail.

SO-5 Feuille de route L'automate doit supporter la demande de «Pull Request» sur les projets Git respectif lorsqu'il y a une amélioration. Il doit faire le suivi et s'assurer de suivre

2. `https://podman.io/`

les règles de contributions de la communauté.
Développer d'autres modules de gestion de projet pour l'accompagnement dans le développement de projet client.
Développer des modules de gestion de communauté sur des projets logiciels libres.
Développer le suivi du développement des modules communautaires avec une traçabilité sur les résultats avec des métriques de génie logiciel.

5.1.1 Comment les résultats obtenus soutiennent le libre

Le réseau d'entraide a besoin d'un support technologie libre, puisque permettre aux participants de respecter leur 4 libertés vont permettre de pouvoir s'adapter à des situations d'urgence et apporter des solutions rapidement.

Étudier La rétro-ingénierie a permis à l'automate de comprendre certaines fonctionnalités pour pouvoir recréer les méta-données adéquatement pour la reproduction.

Copier L'auto-générateur a été mis en place, il reste à auto-reproduire l'automate par son module principale de générateur de code.

Modifier Rend accessible la rétro-ingénierie tout en exécutant des mises en forme de code et validation de qualité logiciel.

Utiliser L'automate a la capacité d'utiliser ses fonctionnalités générées et d'exécuter des scripts d'automatisation à des périodes de temps adaptable.

5.1.2 Couverture des tests

Les tests devraient couvrir 100% du code, cependant la couverture est de 84% pour 3 raisons :

1. Il y a du code fonctionnel non testé, il manque des tests ;
2. Il y a du code désuet qu'il faut nettoyer ou refactoriser ;
3. La gestion des erreurs n'est pas couverte, il faudrait les ignorer dans le test de couverture et faire des tests unitaires qui valide la gestion des erreurs.

5.1.3 Projet Accorderie

La migration de la base de données a été réussie, mais elle est encore à ce jour en adaptation vers un modèle Odoo plus intégré au ERP. Les efforts ont été mis pour la création d'une interface utilisateur avec des technologies qui n'étaient pas à la base supportées dans Odoo.

5.1.4 Projet Portail CEPPP

La signification que le nombre de lignes de XML aurait diminué, c'est que l'automate génère de base toutes les vues de tous les champs. Au moment de la ré-ingénierie, il y a eu beaucoup de nettoyage et de données XML effacées. Cependant, le développeur va mettre plus de code Python pour développer des logiques qui ne sont pas supportés par l'automate. Le Javascript ajouté sert à supporter les dates dans le portail. L'ajout de CSV sert pour l'ajout de permissions et rôles pour l'anonymisation.

Après la première migration par l'extraction du modèle de données par PHP, le client a pu testé la plateforme pour avoir une idée à quoi ressemblerait l'utilisation dans l'espace administration de leur modèle de données et ils ont fait des demandes de changement. Une analyse a été effectuée, nous avons utilisé le générateur de code pour générer les vues portails et fait une ré-ingénierie manuelle du modèle et des vues pour obtenir le résultat désiré. Une des fonctionnalités implémentés est l'anonymisation des données pour certains groupes d'utilisateurs, pour pouvoir visualiser des données sans avoir d'information personnelle sur le patient.

5.2 Forces et limitations

5.2.1 Génération par gabarit

L'utilisation de f-string et de la bibliothèque Code-writer permet de faciliter la lecture du développeur puisque le code du générateur se rapproche plus du résultat généré.

5.2.2 Template de «A template-based code generator for web applications»

Cette application n'est pas accessible facilement au public, il sera difficile de comparer l'efficacité du côté pratique. Ce qui peut être comparé, c'est les performances, hors ça n'a pas été testé dans notre travail, puis l'architecture, mais le fait qu'il n'utilise pas ORM explique la différence d'architecture. Sinon il n'utilise pas de rétro-ingénierie et son générateur n'est pas adapté sur une communauté existante.

5.3 Avenues futures d'amélioration ou d'utilisation

5.3.1 Support de développement de module dans la communauté Odoo

ERPLibre supporte Odoo 12, puisque c'est lui qui supporte le plus de modules. Cependant, ces données représentent seulement ceux des repos utilisés par ERPLibre, il y a en beaucoup plus dans la communauté. C'est pourquoi il faut faire une recherche de ces modules dans la communauté et entreprises, et les rendre accessible par une base de données publiques [3].

Selon l'évolution, il faudrait migrer vers la version 14.0 en 2024. Donc il faut falloir supporter la migration de modules vers des versions supérieures.

5.3.2 Amélioration du générateur code

Il faut intégrer la génération de code à l'intérieur des instances clientes dans l'objectif qu'ils soient accessibles de son gestionnaire de déploiement pour y ajouter les nouvelles fonctionnalités, démarrer la mise à jour, les tests, les améliorations, la migration, importation. Les instances clientes devraient proposer aux clients via les interfaces

Une fois que le générateur de code aura atteint 100% d'auto-génération, il restera limité à produire que les fonctionnalités qu'il utilise. Donc s'auto-générer fait office de test. Il faut faire des tests pour les fonctionnalités qu'il n'utilise pas (ou les combinaisons non utilisés) pour se reproduire. Il reste à auto-générer toutes ses techniques dans des modules qui font de l'héritage sur le générateur de code.

Amélioration de l'architecture Parallélisation de tout le code tout le temps lorsque possible.

Automatisation de la configuration pour le déverminage, automatiser la détection des anomalies, améliorer l'interface no-code pour pouvoir accomplir les mêmes étapes que le mode de paramétrisation «Code hook». Supporter de nouvelles architectures dans la génération de code comme des applications Cordova pour le support mobile natif, des extensions Javascript dans Gnome Shell pour étendre les fonctionnalités du `ERP` directement sur le bureau d'un ordinateur sans passer par un navigateur web, générer des scripts de développement dans le projet ERPLibre qui ne dépendant pas d'Odoo, ou même supporter des applications embarqués.

De plus, il serait pertinent de supporter d'autres ERP externe comme NextERP ou Tryton qui sont des solutions libres. Cela va permettre la migration entre ERP des fonctionnalités

3. Comme fait sur `https://odoo-code-search.com/`

et encourager l'utilisateur à prendre une solution entièrement `AGPLv3` avec une communauté qui le supporte dans cette philosophie.

Problème d'extraction, il était dans le générateur de code au départ dans le développement, il y a donc une extraction à deux endroits qui rend complexe la gestion du code. Au fil de la progression du développement, l'extraction de données par rétro-ingénierie s'avère plus efficace que les méta-données du module dans le système Odoo.

L'auto-ingénierie sur la machine est en cours d'exécution sur le module de base, il faudrait aussi supporter les modules hérités qui sont définis comme des techniques.

Découper les fonctionnalités d'extraction et de génération, puis les séparer dans des modules des techniques du générateur de code.

Il faudrait réduire le nombre de technique dans Base pour qu'ils soient des modules externes par technique, ça faciliterait le changement d'architecture sur la gestion des méta-données, à adapter selon la rétro/ré-ingénierie.

Les travaux d'amélioration devront être effectués après l'auto-génération.

Amélioration de la gestion de projet et statistiques Le générateur de code doit offrir des outils de gestionnaire de projet pour suivre le développement, faire la liaison entre les demandes clients et les avancements des développeurs.

De plus, puisque l'état des méta-données évoluent, il devient difficile de faire le suivi des performances du générateur de code, puisqu'il vient aider dans les boucles d'itérations qui ne nécessitent pas de faire des commits, puisqu'on commit lorsque le tout est stable. Ainsi il faudrait faire des statistiques sur ces itérations pour évaluer la contribution du générateur.

Il manque l'analyse des différences de code sur les différentes sections générées.

Suite du développement du générateur de code Il faudrait qu'il génère des tests fonctionnels, de la documentation fonctionnelle et développe la migration de données selon les changements des versions antérieurs. Un suivi des fonctionnalités selon les exigences clientes.

Une fois l'architecture mise à jour, la prochaine étape est de tester sa mise à niveau de tous les modules dans la communauté et détecter les techniques manquantes par supervision du développeur pour les implémenter. Une fois qu'il aura géré tous les modules de la communauté, on pourra implémenter la migration vers des mises à jour de la plateforme, c'est-à-dire vers Odoo 14, puis vers Odoo 16.

Une fois que l'auto-poïèse sera en place sur la gestion de la machine, la prochaine étape sera

de faire l'auto-poïèse sur tout le code Odoo pour le développement de l'architecture.

5.3.3 Projet Accorderie

Maintenant qu'une plateforme sur le site web a été développée, il faudrait poursuivre la mise à jour du générateur de code pour supporter ce type de plateforme pour des projets futurs similaires.

5.3.4 Projet Portail CEPPP

Le générateur de code a été utilisé en début de projet et à fait économiser du temps de développement et réduit les erreurs possibles de retranscription du langage `PHP` au langage Python, ainsi que la génération des vues admin et portail. Cependant, l'automate a arrêté d'être utilisé au moment qu'on a commencé à diverger vers des fonctionnalités personnalisées qu'il ne pouvait pas supporter. Il faudra supporter ces fonctionnalités pour les futurs projets.

5.3.5 `NLP`

La technologie `NLP` va permettre de comprendre des textes rédigés par l'utilisateur et l'associer à des techniques de programmation.

Le `NLP` est une solution alternative pour interfacer avec l'utilisateur et communiquer avec pour développer des logiciels.

Suggestion d'explorer l'outil libre : Utiliser «HuggingFace» qui contient une grande communauté autour du développement d'un réseau de neurones pour faire du `NLP` par exemple, c'est compatible avec le logiciel libre.

5.3.6 Réseau d'entraide

Il doit y avoir un responsable pour chaque localité accompagné de l'automate pour répondre à ses besoins de numérisations via une souveraineté numérique. C'est le gestionnaire de communauté.

Chaque projet d'urgence doit être capable d'avoir une équipe en charge pour accélérer la résolution de problèmes locaux.

CHAPITRE 6 CONCLUSION

RÉFÉRENCES

[1] "Les principaux erp du marché," Mordor Intelligence, 17 mars 2023. [En ligne]. Disponible : https://www.mordorintelligence.com/fr/industry-reports/enterprise-resource-planning-market

[2] "Marché de la planification des ressources d'entreprise – croissance, tendances, impact du covid-19 et prévisions (2023-2028)," Big Bang ERP inc., 17 mars 2023. [En ligne]. Disponible : https://bigbang360.com/fr/les-principaux-erp-du-marche/

[3] "What 1,384 erp projects tell us about selecting erp (2022 erp report)," Software Path Ltd, 18 janvier 2022. [En ligne]. Disponible : https://softwarepath.com/guides/erp-report

[4] M. Benoit et M.-M. Poulin. (2023, 19 mars) Erp libre v1.5.0 agplv3 contenant odoo 12.0. [En ligne]. Disponible : https://erplibre.ca

[5] M. Michaud et L. K. Audebrand, "Les paradoxes de la transformation d'une association en coopérative de solidarité : le cas de l'accorderie de québec," *Économie et Solidarités*, vol. 44, nº. 1-2, p. 152–168, 2014. [En ligne]. Disponible : https://id.erudit.org/iderudit/1041610ar

[6] D. Margulius. (2019, 27 décembre) Projet portail ceppp - suitecrm 7.10.9. [En ligne]. Disponible : https://github.com/lerenardprudent/ceppp_crm

[7] A. Lehtola *et al.*, *Current platform support for internationalization.* United States : Wiley, 1997, p. 229–287.

[8] Wikipédia. (2023, 17 février) Internationalisation et localisation - i18n. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Internationalisation_et_localisation

[9] A. Sarkar, "Quines are the fittest programs : Nesting algorithmic probability converges to constructors," 2020. [En ligne]. Disponible : https://arxiv.org/abs/2010.09646

[10] Wikipédia. (2022, 10 septembre) Rétro-ingénierie en informatique. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique

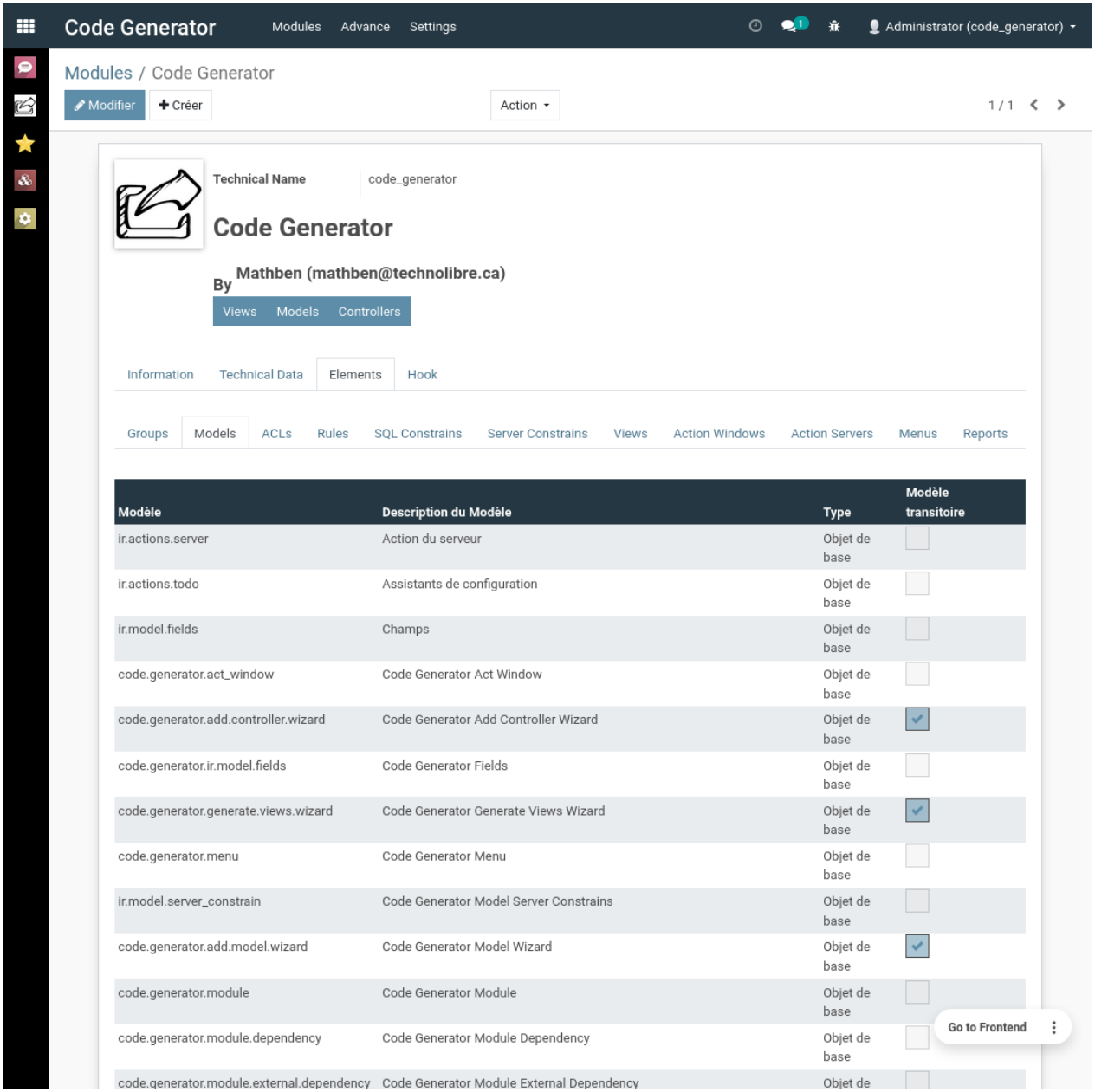
[11] (2021, 18 juin) Image altération du code via la rétro-ingénierie et la ré-ingénierie. [En ligne]. Disponible : https://fr.wikipedia.org/wiki/Rétro-ingénierie_en_informatique#/media/Fichier:Retroingenierie_-_Byrne.svg

[12] G. van Rossum, B. Warsaw et N. Coghlan. (2023, 25 février) Guide de style pour le code python. [En ligne]. Disponible : https://peps.python.org/pep-0008/

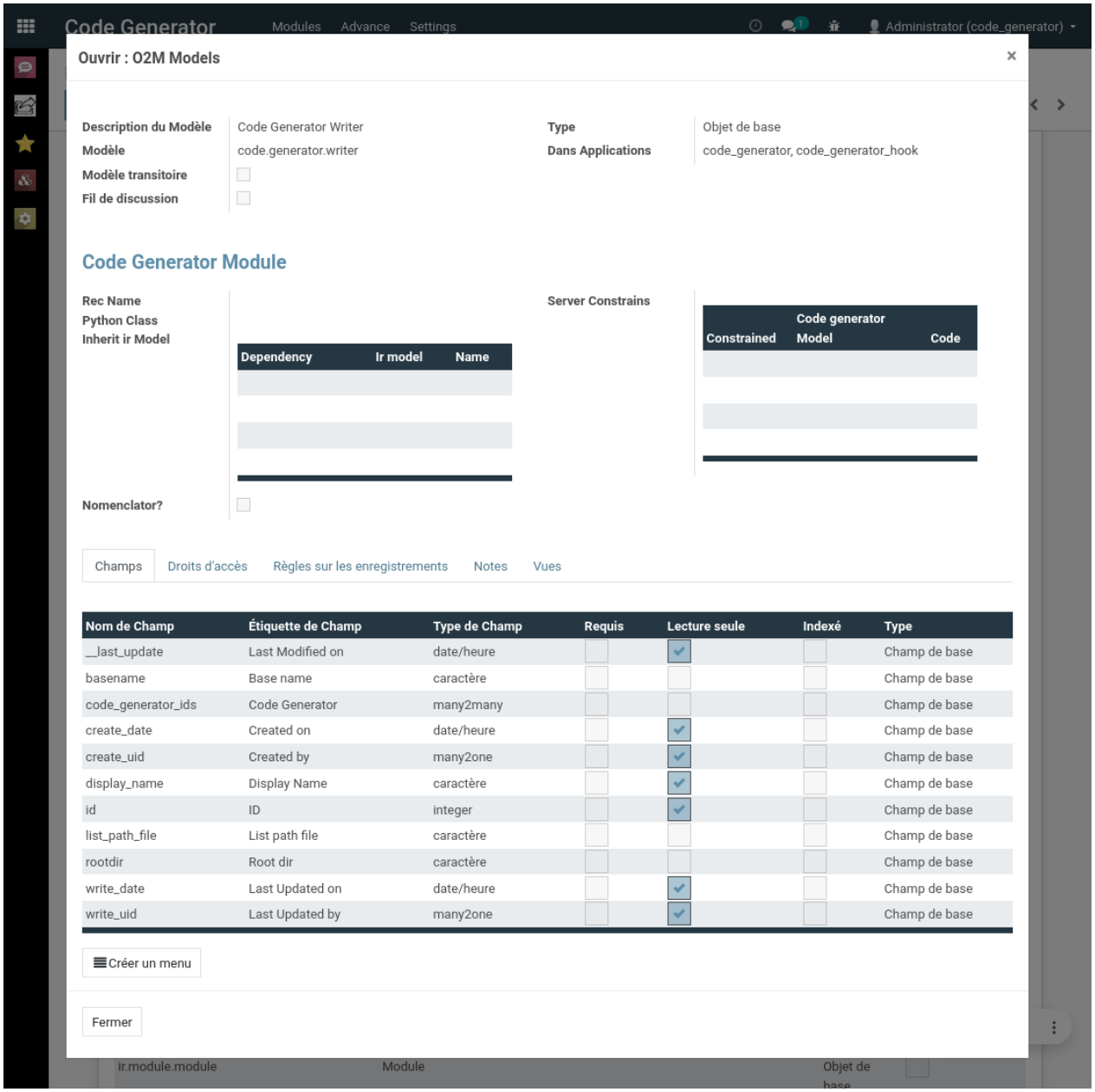
[13] Wikipédia. (2023, 13 mars) Liste de tests informatiques. [En ligne]. Disponible : https://en.wikipedia.org/wiki/Software_testing

[14] B. Luis. (2019, 27 août) Modules odoo 12 - projet initiale du générateur de code et migrateur de base de données. [En ligne]. Disponible : https://github.com/bluisknot/github_odoo_apps/tree/12.0

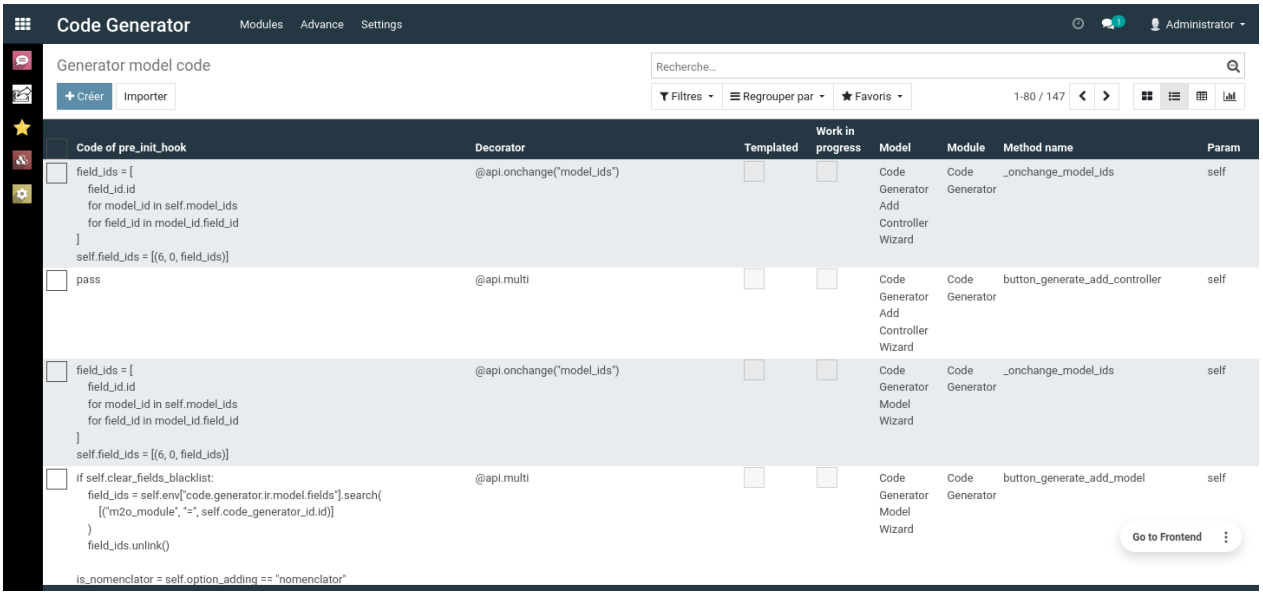
ANNEXE A GUI GÉNÉRATEUR DE CODE - LES MODÈLES



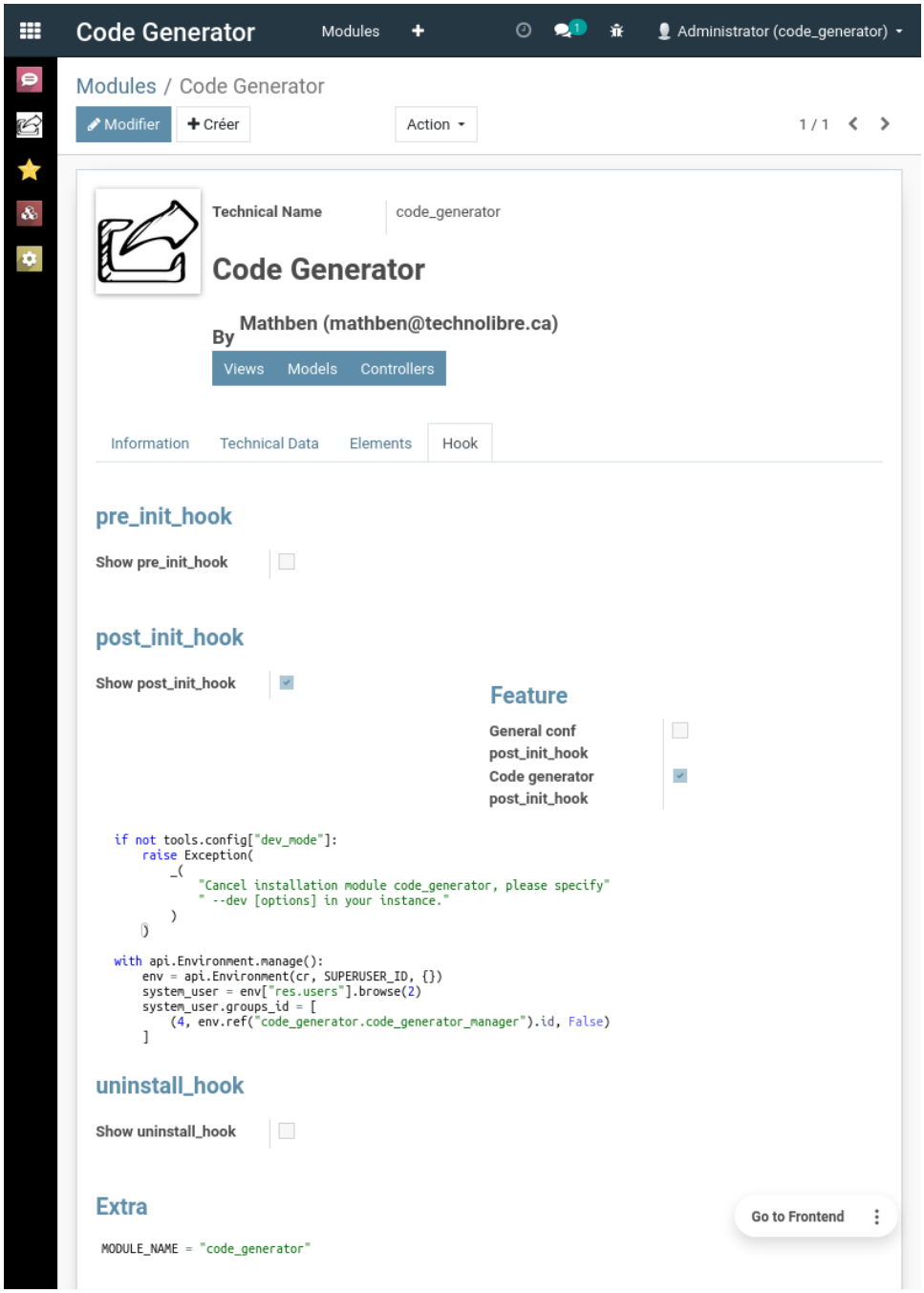
ANNEXE B GUI GÉNÉRATEUR DE CODE - LES CHAMPS



ANNEXE C GUI GÉNÉRATEUR DE CODE - LES CODES



ANNEXE D GUI GÉNÉRATEUR DE CODE - LES «HOOKS»

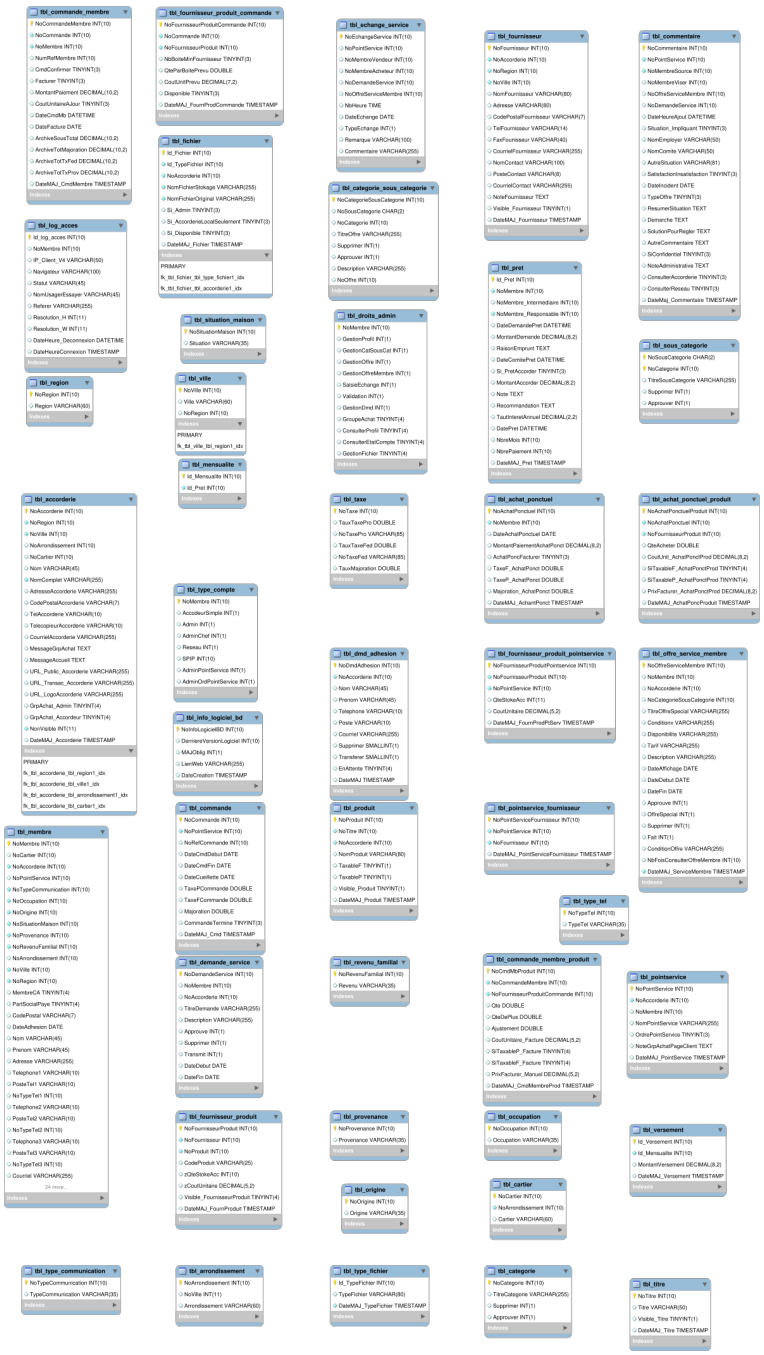


ANNEXE E TEST COUVERTURE TECHNIQUE GÉNÉRATEUR DE CODE

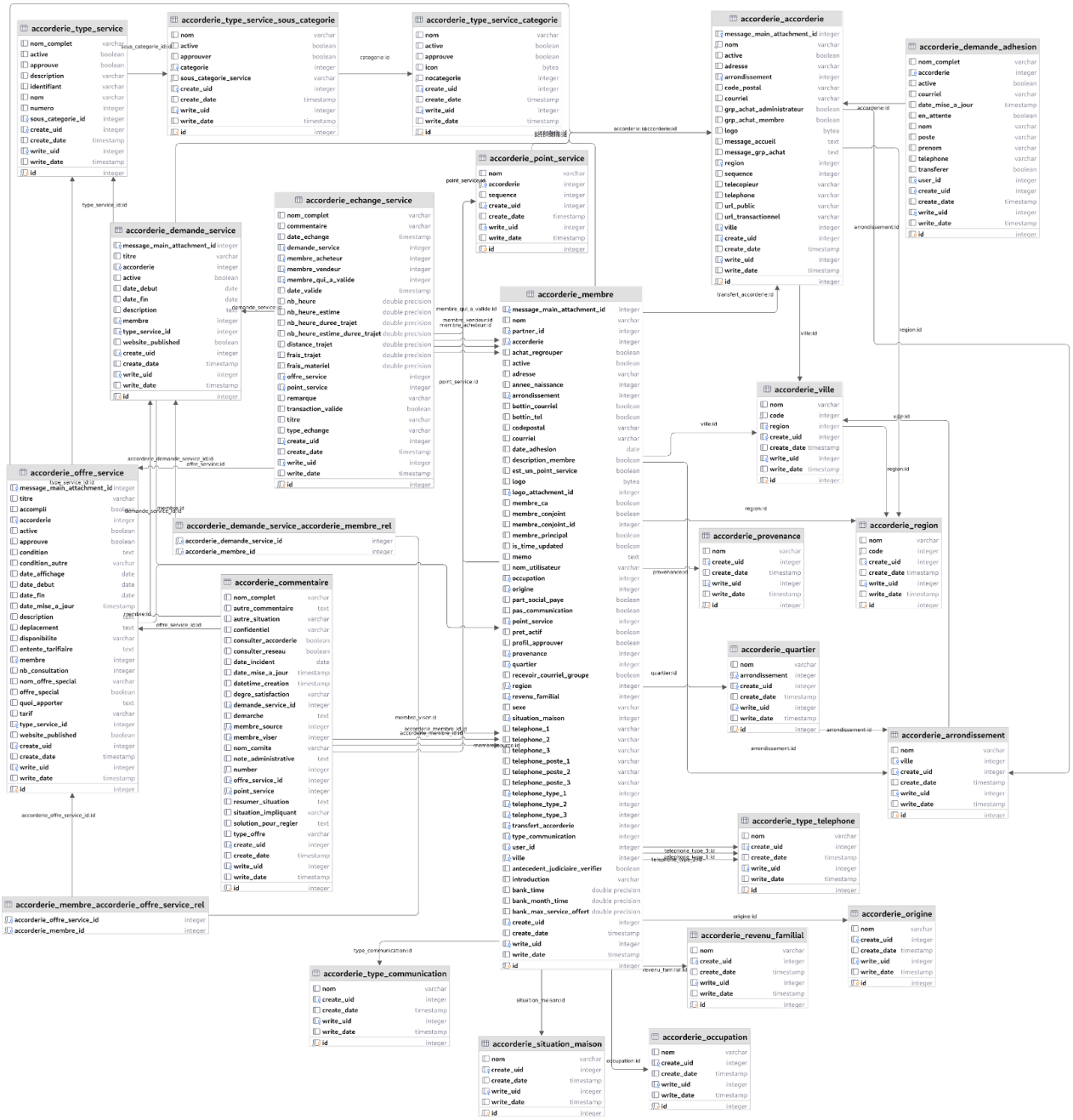
Technique | base		**# instruction**		5 085
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B modèle simple	Succès	27	2 983	41
Génération μ_C^B modèle simple avec héritage	Succès	26	3 452	32
Exportation μ_C^B données «helpdesk»	Succès	28	3 900	23
Technique | base + hook		**# instruction**		5 985
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Auto-génération μ_C^0	Succès	17	4 683	22
Nouveau projet μ_C^0 μ_C^A μ_C^B «Hello World»	Succès	39	4 610	23
Génération μ_C^A portail	Succès	44	3 638	39
Génération μ_C^A modèle simple	Succès	33	3 982	33
Génération μ_C^A modèle simple avec héritage	Succès	32	4 027	33
Exportation μ_C^B données «website»	Succès	28	3 900	23
Génération μ_C^B du générateur de code	Échec	20	3 529	41
Génération μ_C^A du générateur de code	Échec	29	3 690	38
Technique | base + cron		**# instruction**		6 153
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^A «auto_backup»	Succès	36	3 422	44
Technique | base + portal		**# instruction**		5 799
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B exemple MariaDB `SQL`	Succès	80	3 104	46
Technique | base + «theme_website»		**# instruction**		5 336
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B thème «website»	Succès	27	3 938	26
Technique | base + «website_snippet»		**# instruction**		5 615
Description du test	**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B individuel «website_snippet»	Succès	26	4 284	24

Technique	base + portal + «website_snippet»		**# instruction**		6 329
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B multiple «website_snippet»		Succès	36	2 898	54
Génération μ_C^B portail		Succès	34	3 173	50
Technique	base + portal + «Migration DB»		**# instruction**		6 559
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération migration MariaDB `SQL`		Succès	121	3 267	50
Technique	base + hook + portal		**# instruction**		6 699
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^A MariaDB `SQL`		Succès	78	4 315	36
Technique	base + hook + cron		**# instruction**		6 153
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^A «auto_backup»		Succès	36	3 422	44
Technique	base + geoengine + «website_leaflet»		**# instruction**		5 423
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Génération μ_C^B «website_snippet_leaflet»		Succès	33	3 259	40
Technique	base + cron + hook + «Migration DB» + geoengine + portal + «theme_website» + «website_snippet» + «website_leaflet»		**# instruction**		8 746
Description du test		**Status**	**Durée (s)**	**# de Miss**	**Cover (%)**
Tous les tests à succès		Succès	194	1 371	84

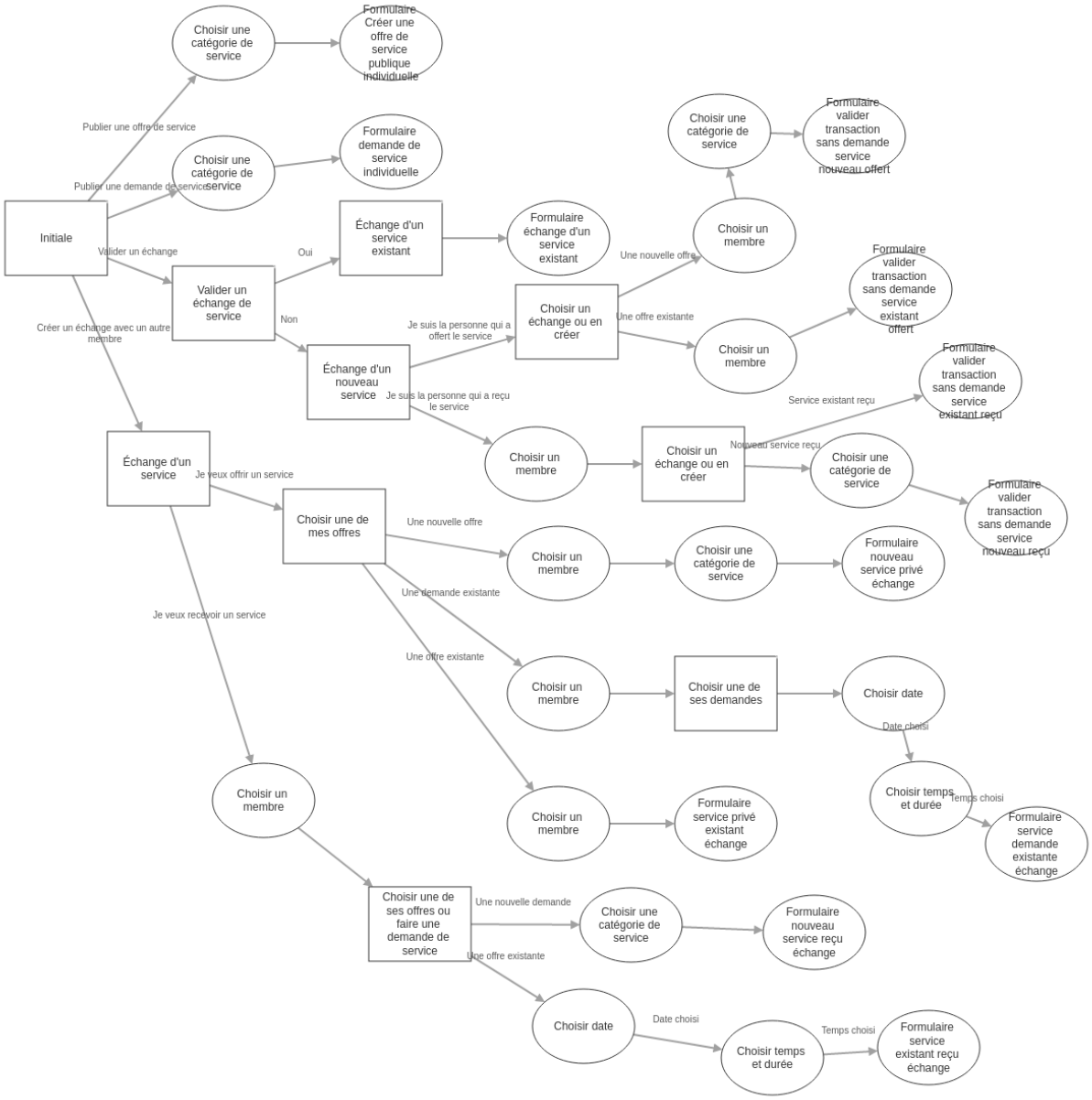
ANNEXE F DIAGRAMME MODÈLE DE DONNÉES ESPACE MEMBRE ACCORDERIE 2019



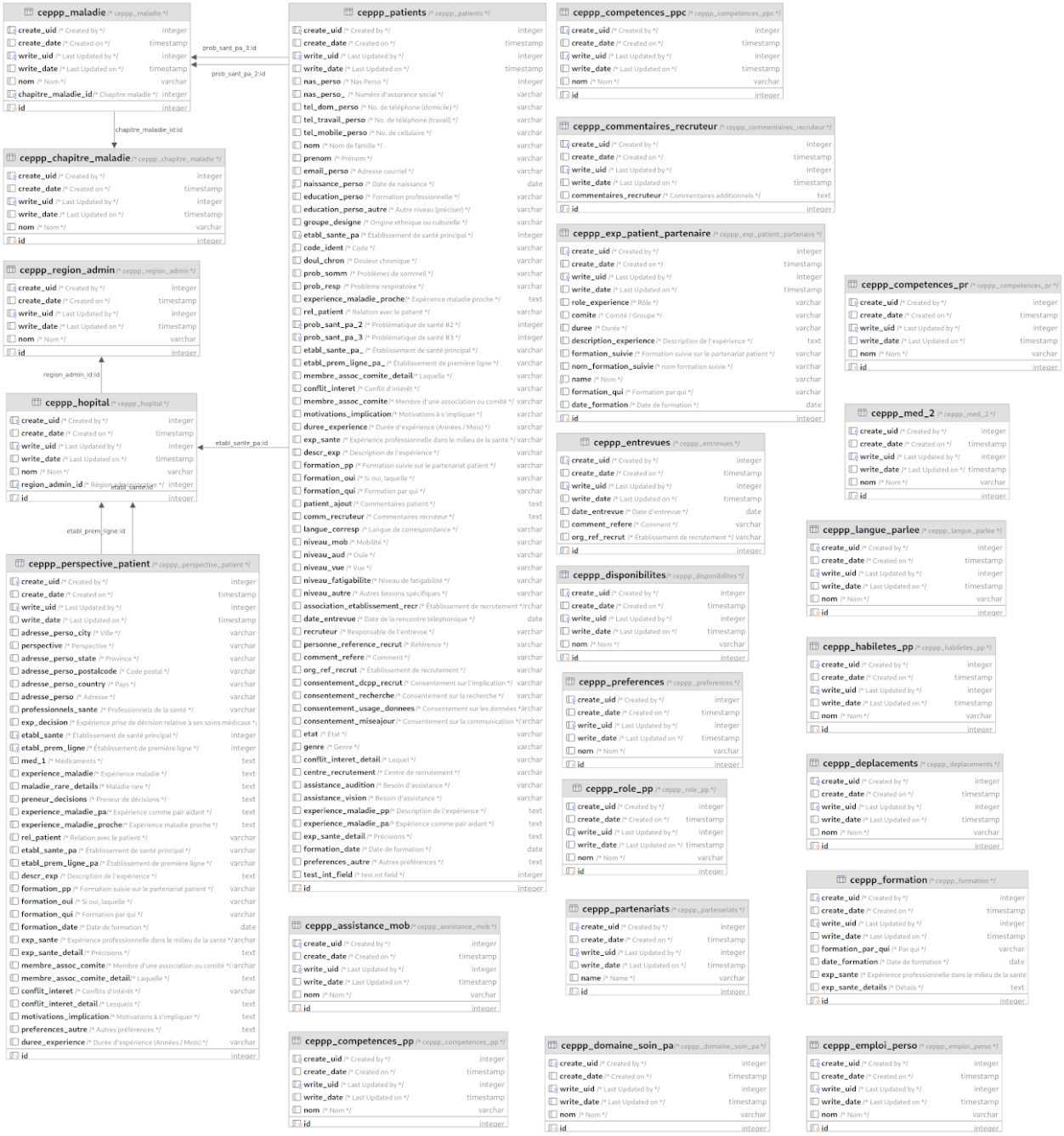
ANNEXE G DIAGRAMME NOUVEAU MODÈLE DE DONNÉES ESPACE MEMBRE ACCORDERIE 2023



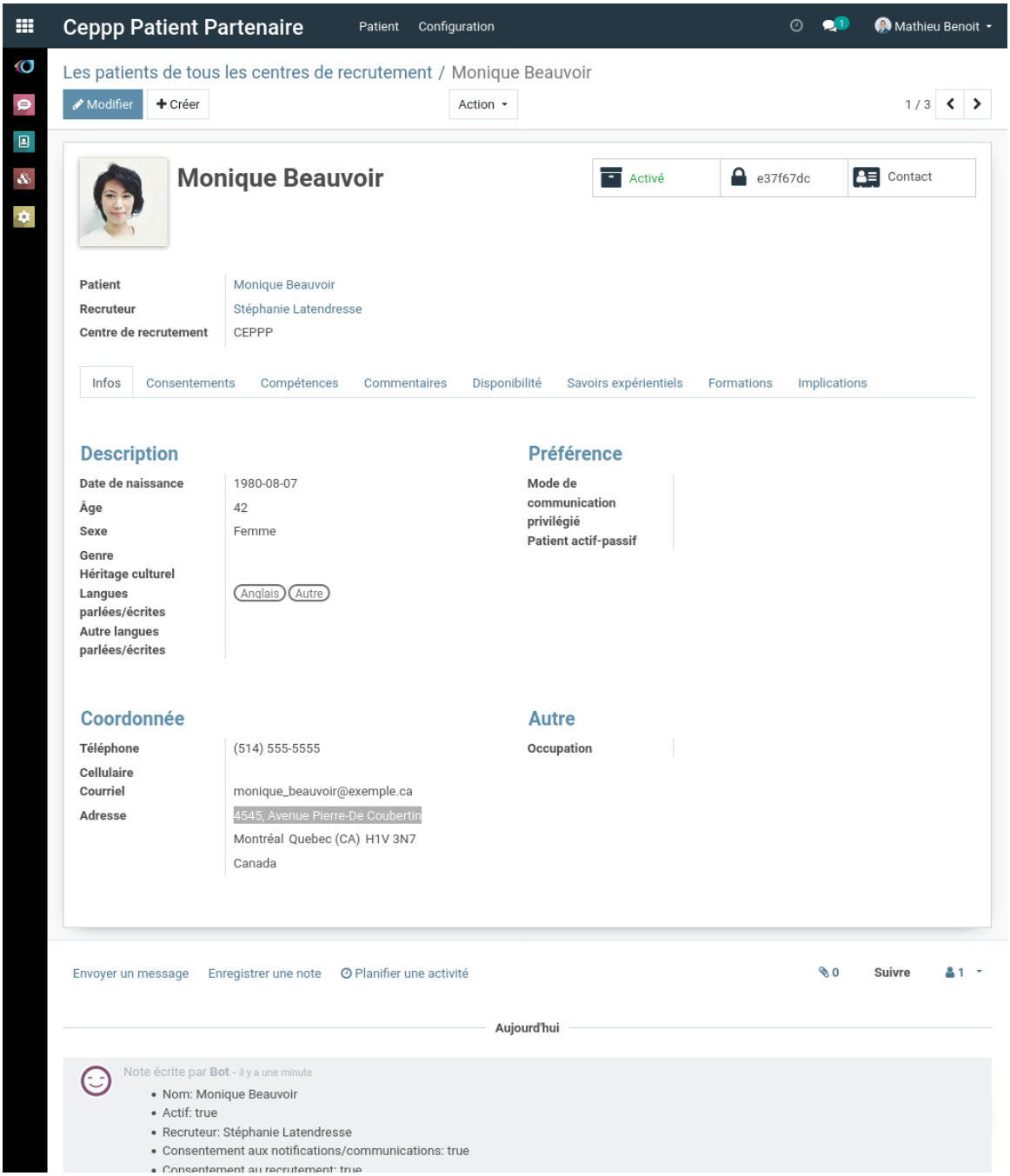
ANNEXE H DIAGRAMME PROCESSUS POUR DEMANDER, OFFRIR, ÉTABLIR UN ÉCHANGE ET LE VALIDER - ACCORDERIE 2023



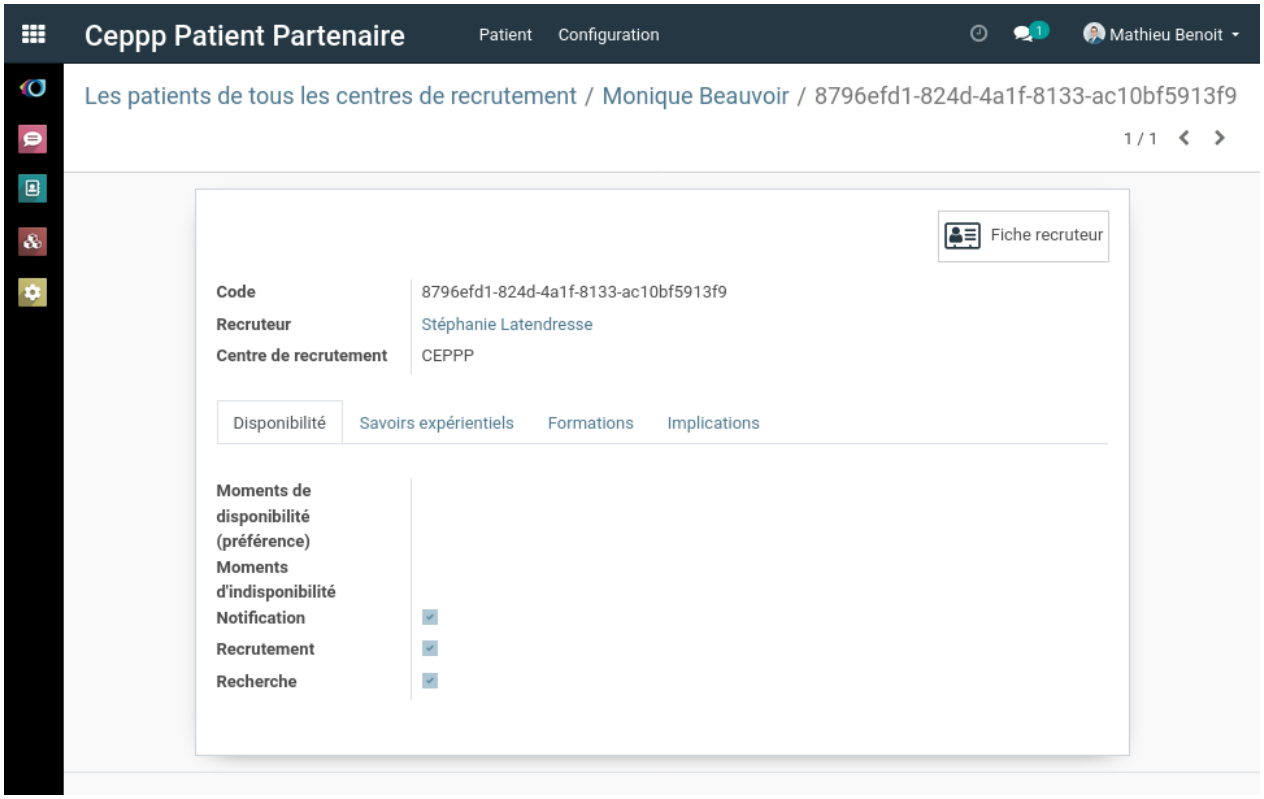
ANNEXE I DIAGRAMME MODÈLE DE DONNÉES DU PORTAIL CEPPP SEPTEMBRE 2022



ANNEXE J VUE FORMULAIRE ADMINISTRATION PORTAIL CEPPP



ANNEXE K VUE FORMULAIRE PARTENAIRE PORTAIL CEPPP



ANNEXE L CODE U_C^0 DANS LE GÉNÉRATEUR DE CODE

```python
import os

from odoo import SUPERUSER_ID, _, api, fields, models

# TODO HUMAN: change my module_name to create a specific demo
    functionality
MODULE_NAME = "code_generator_demo"


def post_init_hook(cr, e):
    with api.Environment.manage():
        env = api.Environment(cr, SUPERUSER_ID, {})

        # The path of the actual file
        # path_module_generate = os.path.normpath(os.path.join(os.path.
    dirname(__file__), '..'))

        short_name = MODULE_NAME.replace("_", " ").title()

        # Add code generator
        value = {
            "shortdesc": short_name,
            "name": MODULE_NAME,
            "license": "AGPL-3",
            "author": "TechnoLibre",
            "website": "https://technolibre.ca",
            "application": True,
            "enable_sync_code": True,
            # "path_sync_code": path_module_generate,
        }

        # TODO HUMAN: enable your functionality to generate
        value["enable_template_code_generator_demo"] = True
        value["template_model_name"] = ""
        value["template_inherit_model_name"] = ""
        # value["template_module_path_generated_extension"] = "."
        value["enable_template_wizard_view"] = False
        value["force_generic_template_wizard_view"] = False
        value["disable_generate_access"] = False
        value["enable_template_website_snippet_view"] = False
```

```
        value["enable_sync_template"] = False
        value["ignore_fields"] = ""
        value["post_init_hook_show"] = True
        value["uninstall_hook_show"] = True
        value["post_init_hook_feature_code_generator"] = True
        value["uninstall_hook_feature_code_generator"] = True

        new_module_name = MODULE_NAME
        if (
            MODULE_NAME != "code_generator_demo"
            and "code_generator_" in MODULE_NAME
        ):
            if "code_generator_template" in MODULE_NAME:
                if value["enable_template_code_generator_demo"]:
                    new_module_name = f"code_generator_{MODULE_NAME[len('
    code_generator_template_'):]}"
                else:
                    new_module_name = MODULE_NAME[
                        len("code_generator_template_") :
                    ]
            else:
                new_module_name = MODULE_NAME[len("code_generator_") :]
            value["template_module_name"] = new_module_name
        value["hook_constant_code"] = f'MODULE_NAME = "{new_module_name}"'

        code_generator_id = env["code.generator.module"].create(value)

        # Add dependencies
        lst_depend_module = ["code_generator", "code_generator_hook"]
        code_generator_id.add_module_dependency(lst_depend_module)
        code_generator_id.add_module_dependency_template(lst_depend_module)
        # Generate module
        value = {"code_generator_ids": code_generator_id.ids}
        env["code.generator.writer"].create(value)


def uninstall_hook(cr, e):
    with api.Environment.manage():
        env = api.Environment(cr, SUPERUSER_ID, {})
        code_generator_id = env["code.generator.module"].search(
            [("name", "=", MODULE_NAME)]
        )
        if code_generator_id:
            code_generator_id.unlink()
```