Intégration Odoo

« Formation à l'ERP Odoo et conception de la structure de données personnalisée pour les besoins de SabSystem. »

Par Vasco Valadares Semana – En 2024 – Projet en Entreprise

Table des matières

1. Introduction 3
2. Mes Objectifs 3
2.1 Contexte de Développement 3
2.2 Mes Objectifs Principaux 3
3. Les Concepts Clés d'Odoo 4
3.1. Qu'est-ce qu'Odoo ? 4
3.2. Modèles et Champs : La Base de la Structure 4
3.3. Relations entre Modèles (One2many, Many2one) 4
3.4. Vues XML et Odoo Studio 5
4. Mon Travail de Modélisation des Données dans Odoo pour SabSystem 5
4.1. Analyse des Besoins Spécifiques de SabSystem 5
4.2. Conception du Modèle "Contact" (res.partner) Étendu 5
4.3. Définition d'Autres Modèles Clés pour Notre Activité 6
4.4. Mon Fichier odoo_model.py : Référence pour l'Intégration 7
4.5. Visualisation de la Structure 7
5. Compétences Odoo Acquises et Mises en Pratique (via les Exercices de Formation) 7
5.1. Création d'Applications et de Modèles Personnalisés 7
5.2. Définition des Champs et des Relations Complexes 8
5.3. Création de Vues Odoo (Listes, Kanban) 8
6. Mon Rôle dans l'Intégration avec le Projet de mon Collègue de Travail 8
7. Bilan Personnel de cette Expérience Odoo 9
8. Perspectives d'Évolution pour Notre Structure Odoo 9
9. Conclusion 10

1. Introduction

Ce projet représente ma montée en compétence sur l'ERP Odoo, acquise au travers d'une formation interne de quelques heures et d'exercices pratiques. L'objectif principal de cette formation était de me doter des compétences nécessaires pour concevoir et modéliser la structure de "base de données" (les modèles Odoo) pour notre société, SabSystem. Ce travail de modélisation a été crucial et a servi de fondation pour d'autres projets internes, notamment un projet réalisé par un de mes collègues, pour lequel j'ai fourni la structure Odoo cible.

Note : Je ne peux pas fournir de captures d'écran ou morceaux de code pour cette documentation.

2. Mes Objectifs

2.1 Contexte de Développement

SabSystem avait besoin de structurer et de centraliser ses données, en particulier celles relatives à nos contacts, aux établissements hôteliers que nous suivons, et aux informations que nous collectons (par exemple via les données SIREN). Ma formation sur Odoo a été initiée pour répondre à ce besoin stratégique.

2.2 Mes Objectifs Principaux

Au travers de cette formation et du travail de modélisation qui a suivi, mes objectifs étaient de :

- **Acquérir les Compétences Fondamentales sur Odoo :** Comprendre le fonctionnement de l'ERP, son ORM, la manière de définir des modèles, des champs, des relations et des vues ;
- **Modéliser la Structure de Données pour SabSystem :** Concevoir une structure de données dans Odoo qui réponde précisément à nos besoins métier, en particulier pour notre CRM et la gestion des informations sur les établissements ;
- **Définir les Champs Personnalisés :** Identifier et créer les champs spécifiques (x_studio_*) nécessaires pour stocker des informations non couvertes par les modèles Odoo standards ;
- **Intégrer les Données Externes :** Préparer la structure Odoo pour recevoir et organiser efficacement les données issues de sources externes (informations sur les hôtels/entreprises telles que : classement, nombre d'avis, URL, etc.) et les informations légales des entreprises (SIREN, catégorie juridique, etc.) ;
- **Fournir un Modèle de Données Clair :** Produire une définition claire de notre structure Odoo (comme illustré par mon fichier odoo_model.py) pour faciliter son utilisation par d'autres systèmes ou collègues, notamment pour le projet mentionné plus tôt.

3. Les Concepts Clés d’Odoo

Ma formation m'a permis de me familiariser avec Odoo, un Progiciel de Gestion Intégré (ERP) open source très puissant et flexible.

3.1. Qu'est-ce qu'Odoo ?

J'ai appris qu'Odoo est construit sur une architecture modulaire, où chaque fonctionnalité (CRM, Ventes, Inventaire, etc.) est un "module" ou une "application" qui peut être installée indépendamment. Il est principalement développé en Python et s'appuie sur une base de données PostgreSQL. Un des aspects les plus importants que j'ai appréhendés est son **ORM (Object-Relational Mapper)**. Cet ORM permet de manipuler la base de données en utilisant des objets Python, ce qui simplifie grandement le développement et la personnalisation.

3.2. Modèles et Champs : La Base de la Structure

Dans Odoo, les "tables" d'une base de données traditionnelle sont appelées des Modèles (ou objets). J'ai appris à définir ces modèles en Python en héritant de la classe models.Model.

Chaque modèle est composé de Champs, qui sont l'équivalent des colonnes d'une table. J'ai travaillé avec différents types de champs Odoo :

- **fields.Char :** Pour les chaînes de caractères ;
- **fields.Text :** Pour les textes longs ;
- **fields.Integer, fields.Float :** Pour les nombres ;
- **fields.Boolean :** Pour les valeurs vrai/faux ;
- **fields.Date, fields.Datetime :** Pour les dates et dates/heures ;
- **fields.Selection :** Pour les listes de choix prédéfinis, comme un enum ;
- Et les champs relationnels que je détaille ci-dessous.

3.3. Relations entre Modèles (One2many, Many2one)

La puissance d'Odoo réside en grande partie dans sa capacité à gérer des relations complexes entre les modèles. J'ai particulièrement travaillé avec :

- **fields.Many2one(comodel_name, string) :** Ce type de champ crée une relation "plusieurs-à-un". C'est l'équivalent d'une clé étrangère. Par exemple, dans mon modèle Etablissement, j'ai un champ res_partner_id = fields.Many2one('res.partner', string='Contact Principal') pour lier un établissement à un contact principal. De même, un établissement peut avoir un type_etablissement_id qui est un Many2one vers un modèle etablissement.type. Les champs se terminant par _id dans Odoo sont typiquement des Many2one ;
- **fields.One2many(comodel_name, inverse_name, string) :** Ce type de champ crée une relation "un-à-plusieurs". Il représente une liste d'enregistrements d'un autre modèle qui sont liés à l'enregistrement courant. Par exemple, un contact (res.partner) peut avoir plusieurs notes. J'ai modélisé cela avec un champ x_studio_notes = fields.One2many('note.etablissement', 'contact_id', string='Notes

Établissement') dans mon modèle Contact étendu. Le paramètre inverse_name est le champ Many2one dans l'autre modèle qui pointe vers le modèle courant. Les champs se terminant par _ids dans Odoo sont souvent des One2many ;

- J'ai aussi compris le concept des relations Many2many, bien que moins directement utilisées pour ce projet spécifique.

3.4. Vues XML et Odoo Studio

J'ai appris que l'interface utilisateur d'Odoo est définie à l'aide de fichiers XML. Ces vues décrivent comment les données des modèles doivent être présentées (formulaires, listes, vues kanban, graphiques, etc.).

Lors de ma formation, j'ai créé des vues listes et kanban pour mes modèles personnalisés.

De plus, une grande partie des champs personnalisés que j'ai définis pour SabSystem ont été créés ou modifiés à l'aide d'Odoo Studio. C'est un outil intégré à Odoo qui permet de personnaliser l'application (modèles, vues, rapports, automatisations) directement depuis l'interface web, sans nécessairement écrire de code Python ou XML, ce qui est très pratique pour des ajustements rapides ou pour des utilisateurs moins techniques.

4. Mon Travail de Modélisation des Données dans Odoo pour SabSystem

Ma tâche principale suite à la formation a été de concevoir la structure de données dans Odoo pour répondre aux besoins spécifiques de SabSystem.

4.1. Analyse des Besoins Spécifiques de SabSystem

En me basant sur nos discussions internes et des documents fournis, j'ai analysé les informations que nous devions stocker et gérer. Cela incluait :

- Les informations de contact de base (entreprises, individus) ;
- Des données spécifiques aux établissements hôteliers ;
- Des informations légales et administratives (SIREN, catégorie juridique) ;
- La possibilité de lier des notes, des prestataires, et de gérer des relations capitalistiques ou de gestion ;
- La nécessité de champs pour le câblage, la classification des hôtels, etc.

4.2. Conception du Modèle "Contact" (res.partner) Étendu

Le modèle res.partner d'Odoo (qui gère les contacts, qu'ils soient des entreprises ou des individus) a été au centre de ma modélisation. J'ai étendu ce modèle en y ajoutant de nombreux champs personnalisés via Odoo Studio pour stocker toutes les informations dont nous avions besoin.

On y trouve :

- **Champs par défaut Odoo** que nous utilisons (name, street, city, email, etc.) ;

- **Champs Spécifiques Scraping (projet interne)** : Pour stocker les données issues du scraping :
    - x_studio_date_scraping (Date) ;
    - x_studio_classement (Integer, pour les étoiles) ;
    - x_studio_categorie_etablissement (Many2one vers un modèle personnalisé de catégories d'établissement) ;
    - x_studio_nombre_avis (Integer) ;
    - x_studio_url_booking (Char).
- **Champs pour les Notes d'Établissement** : J'ai conçu une relation One2many (x_studio_notes) depuis le Contact vers un modèle personnalisé note.etablissement pour stocker plusieurs notes (avec date, type de note, et valeur).
    - note.etablissement contient : x_studio_date, x_studio_type (Many2one vers un modèle categorie.note), x_studio_note (Float).
- **Champs pour les Informations SIREN** : Un ensemble complet de champs pour stocker les données légales et administratives des entreprises :
    - x_studio_identifiant_siren, x_studio_siret, x_studio_date_de_creation_etablissement, x_studio_etablissement_siege (Boolean), x_studio_enseigne_etablissement, x_studio_activite_principale (Many2one), etc.
    - J'ai prévu des champs Many2one vers des modèles dédiés pour des listes de valeurs comme x_studio_categorie_juridique, x_studio_tranche_effectifs_etablissement, etc. Ces modèles dédiés (ex: categorie.juridique) contiendraient typiquement un champ id et un champ x_name (ou name).

4.3. Définition d'Autres Modèles Clés pour Notre Activité

En plus d'étendre res.partner, ma formation et les exercices m'ont préparé à définir d'autres modèles personnalisés nécessaires pour SabSystem :

- **etablissement.hotel (ou similaire)** : J’ai travaillé sur la création d'un modèle "Etablissement" qui serait lié en Many2one à res.partner (un contact peut être un établissement). Ce modèle contiendrait des champs spécifiques aux hôtels (SIREN, URL Booking, etc., qui ont finalement été intégrés directement dans res.partner via Odoo Studio pour notre cas d'usage, mais le concept de modèle séparé a été exploré) ;
- **etablissement.type** : Un modèle simple (id, nom) pour catégoriser les types d'établissements (hôtel, restaurant, etc.), lié en Many2one depuis res.partner ou etablissement.hotel ;
- **ratings.etablissement** : Le modèle que j'ai conçu pour stocker les notes, avec une relation Many2one vers le contact/établissement (contact_id) et des champs pour la valeur de la note, le type de note (ex: Wifi, Personnel, Propreté - lui-même un Many2one vers un modèle ratings.type), et les dates ;
- **marques.hotel** : Pour lier les établissements à des marques hôtelières ;

- **Modèles de Référence pour Données SIREN** : J'ai prévu que des champs comme "Activité principale", "Catégorie juridique", "Tranche d'effectifs" dans res.partner soient des relations Many2one vers des modèles dédiés (ex: activite.principale, categorie.juridique). Ces modèles de référence contiendraient simplement un id et un x_name (ou name), comme défini dans ma dataclass OdooIdNameModel. Cela assure la cohérence des données et facilite la sélection via des listes déroulantes dans Odoo.

4.4. Mon Fichier odoo_model.py : Référence pour l'Intégration

Le fichier odoo_model.py que j'ai créé à l'aide de mon collègue sert de "contrat d'interface" et de référence pour nous et pour les scripts externes (comme celui de scraping mentionné précédemment). Il liste les **noms techniques exacts** des champs Odoo (y compris ceux créés via Odoo Studio, préfixés x_studio_) que nous utilisons.

- Le dictionnaire odoo_fields_names mappe des noms logiques que nous utilisons en interne aux noms techniques Odoo ;
- Les dataclass Python (comme OdooContactModel, OdooIdNameModel, OdooNoteEtablissementModel) fournissent une manière structurée et auto-documentée d'accéder à ces noms de champs dans du code Python externe qui interagirait avec Odoo.

C'est un élément clé que j'ai produit pour assurer la cohérence entre notre modèle Odoo et les autres systèmes ou scripts qui pourraient avoir besoin d'interagir avec.

4.5. Visualisation de la Structure

Pour mieux visualiser l'ensemble de ces modèles et leurs relations, j'ai également travaillé sur un schéma de type diagramme entité-relation. Ce type de diagramme est essentiel pour avoir une vue d'ensemble de la manière dont nos données sont interconnectées dans Odoo.

5. Compétences Odoo Acquises et Mises en Pratique (via les Exercices de Formation)

Ma formation comprenait des exercices pratiques qui m'ont permis de concrétiser ma compréhension d'Odoo.

5.1. Création d'Applications et de Modèles Personnalisés

J'ai dû créer une nouvelle application "Hôtels" et un nouveau modèle "Etablissement" dans le cadre d'exercices personnalisés durant cette formation. J'ai appris à :

- Définir la structure d'une nouvelle application Odoo ;
- Créer un modèle Python héritant de models.Model ;
- Y ajouter des champs de différents types (Char pour SIREN, Nom, Rue, Ville, Code Postal ; Many2one pour lier à res.partner et à un nouveau modèle etablissement.type) ;
- J'ai également créé le modèle ratings.etablissement avec ses propres champs (valeur, type de note, dates) et une relation One2many depuis "Etablissement"

(ratings_ids).

5.2. Définition des Champs et des Relations Complexes

À travers ces exercices, j'ai mis en pratique la création de :

- **Champs Many2one :** Par exemple, lier un Etablissement à un res.partner ou à une Marque. J'ai compris comment Odoo gère cela en stockant un ID et en permettant une sélection via une liste déroulante ou une recherche ;
- **Champs One2many :** Comme le champ ratings_ids sur le modèle Etablissement qui pointe vers plusieurs enregistrements dans ratings.etablissement. J'ai appris que cela se manifeste souvent par un onglet ou une section dans le formulaire Odoo où l'on peut ajouter/modifier les enregistrements liés ;

Lors de cette formation j'avais un peu de mal à assimiler le concept de relations Many2One et One2Many et comment les repérer dans odoo, le conseil "Les champs qui se termine par _id (Many2one) et _ids (One2many)" a été une bonne règle empirique.

5.3. Création de Vues Odoo (Listes, Kanban)

Ensuite, on m'a initié à la création de vues en XML :

- **Vues Listes :** J'ai appris à définir les colonnes à afficher pour des modèles comme Groupe, Parc installé, Emplacement, Prestataires ;
- **Vue Kanban :** J'ai créé une vue Kanban pour le modèle Etablissement, ce qui permet un affichage sous forme de cartes, utile pour une vue d'ensemble.

Il a ensuite fallu renseigner un jeu de données, ça m'a permis de voir concrètement comment mes modèles et vues fonctionnaient avec des données réelles.

6. Mon Rôle dans l'Intégration avec le Projet de mon Collègue de Travail

Une des applications directes de mon travail de modélisation Odoo a été de fournir la structure cible pour un projet de scraping, mené par un de mes collègues.

- **Définition de la Cible :** J'ai défini les champs spécifiques dans notre modèle Contact (étendu via Odoo Studio) pour accueillir les données issues du scraping :
    - x_studio_date_scraping
    - x_studio_classement (les étoiles de l'hôtel)
    - x_studio_categorie_etablissement
    - x_studio_nombre_avis
    - x_studio_url
- **Interface pour le Script de Scraping :** Mon fichier odoo_model.py, avec ses dictionnaires et dataclasses, a servi de référence claire pour mon collègue développant le script de scraping. Il savait ainsi exactement quels noms de champs techniques Odoo utiliser pour insérer ou mettre à jour les données dans notre système ;

- **Cohérence des Données :** En préparant la structure Odoo en amont, nous nous sommes assurés que les données de scraping seraient stockées de manière cohérente et pourraient être facilement exploitées par d'autres modules ou rapports dans Odoo.

Ce travail a été un bon exemple de collaboration où ma compréhension de la structure Odoo a été essentielle pour un autre projet technique.

7. Bilan Personnel de cette Expérience Odoo

Cette formation sur Odoo et le travail de modélisation de données qui a suivi ont été très formateurs pour moi. J'ai pu :

- **Découvrir en profondeur un ERP majeur :** Comprendre son architecture, sa philosophie modulaire et la puissance de son ORM ;
- **Acquérir des compétences pratiques en personnalisation Odoo :** Que ce soit par la création de modèles Python, la définition de vues XML, ou l'utilisation d'Odoo Studio pour des ajustements rapides ;
- **Comprendre l'importance de la modélisation des données :** J'ai réalisé à quel point une structure de données bien pensée est cruciale pour le bon fonctionnement d'un ERP et pour l'intégration de données externes ;
- **Mettre en pratique les relations complexes :** La manipulation des champs One2many et Many2one m'a donné une meilleure compréhension de la manière dont les données sont interconnectées dans un système complexe ;
- **Contribuer concrètement à un besoin de SabSystem :** Savoir que la structure que j'ai aidé à définir est utilisée pour des projets concrets comme l'intégration des données du scraping est très gratifiant.

Bien que cette formation n'ait pas abouti à une certification, les compétences que j'ai acquises sont directement applicables et précieuses pour les besoins de SabSystem.

8. Perspectives d’Évolution pour Notre Structure Odoo

- **Affiner les Relations Prestataires :** Mieux définir les liens entre les Contacts (qui peuvent être des prestataires) et les établissements, potentiellement via des modèles intermédiaires pour qualifier les types de services fournis ;
- **Modélisation du "Parc Installé" :** Développer des modèles pour détailler les équipements installés dans chaque établissement, chambre par chambre, et lier cela à un système de gestion des tickets d'incident ;
- **Gestion des Liens Capitalistiques et de Gestion :** Implémenter les relations pour tracer les propriétaires, les gestionnaires, les mandats de gestion et les appartenances à des groupes ou franchises ;
- **Historisation Poussée :** Bien que nous ayons commencé avec les notes, étendre l'historisation à d'autres types de données (ex : changements de prestataire WiFi, historique des classements) ;

- **Vues et Rapports Avancés :** Une fois les données bien structurées et peuplées, développer des vues et des rapports personnalisés dans Odoo pour exploiter ces informations (ex : rapport sur les hôtels par marque avec leurs notes) ;
- **Automatisation des Mises à Jour :** Participer au développement du scraping pour mettre à jour plus régulièrement les données SIREN ou d'autres sources externes.

9. Conclusion

Ma formation à Odoo et le travail de conception de notre structure de données interne ont été une étape importante dans mon parcours chez SabSystem. J'ai pu non seulement acquérir une compréhension solide du fonctionnement de cet ERP, mais aussi contribuer activement à la mise en place d'un modèle de données adapté à nos besoins spécifiques, notamment pour l'intégration d'informations cruciales issues de sources comme les données SIREN. Mon fichier odoo_model.py est un livrable concret de ce travail, servant de pont entre notre instance Odoo et d'autres outils. Cette expérience a renforcé mes compétences en modélisation de données et ma capacité à adapter des solutions logicielles complexes.