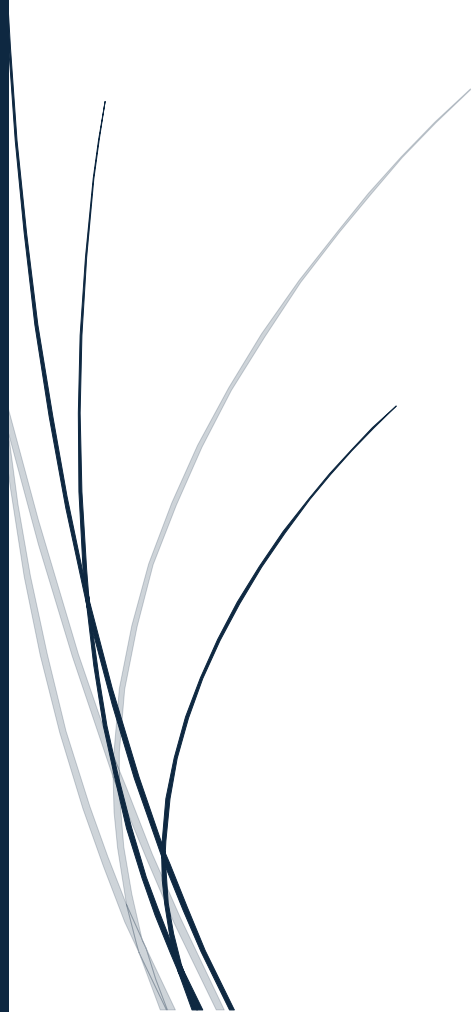
09/09/2024

Rapport de stage

Travail au sein de l'entreprise Xev



RYCKAERT Valentin
INSTITUTION DES CHARTREUX – BTS SIO 1ERE ANNEE (SLAM)

Remerciements

Avant de commencer, je tiens à remercier chaleureusement Antoine Hervet, mon maître de stage et PDG de l'entreprise Xev, qui m'a permis de travailler dans son entreprise et de découvrir le monde d'une entreprise informatique. Ce stage fut pour moi une expérience à la fois agréable, enrichissante et formatrice. Il a renforcé chez moi le goût du travail, de la discipline personnelle, et du perfectionnement.

Je souhaite également remercier l'ensemble de l'équipe de Xev, qui pendant tout ce stage m'a soutenu, accompagné, conseillé, aidé lorsque j'étais en difficulté. Ils m'ont permis de vivre une expérience unique au sein de leur équipe.

Table des matières

Remerciements .....
Introduction ..... - 1 -
Contexte ..... - 1 -
L'entreprise Xev ..... - 1 -
Mon travail ..... - 1 -
Projet et activités ..... - 2 -
La technique ..... - 2 -
Conduite de projet ..... - 3 -
Exemple de projet (Annexes 3 à 7) ..... - 4 -
Conclusion ..... - 5 -
Annexes ..... - 6 -
Annexe 1 : Algorithme en python de traitement de fichiers CSV ..... - 6 -
Annexe 3 : Capture d'écran du plan comptable dans l'ERP ..... - 8 -
Annexe 4 : Capture d'écran des sociétés dans l'ERP ..... - 9 -
Annexe 5 : Capture d'écran du formulaire d'ajout d'une écriture comptable dans l'ERP ..... - 9 -
Annexe 6 : extrait de la documentation utilisateur du module « comptabilité » de l'ERP ..... - 10 -
Annexe 7 : Documentation technique de l'ERP ..... - 11 -
Ressources ..... - 14 -

Introduction

Lorsque j'ai débuté la première année de BTS, je possédais déjà des connaissances en développement et en algorithmie. Je savais programmer en Python, en Ruby et en Java. Je n'avais cependant que peu de connaissance en systèmes et réseaux. Mon ressenti de début de première année fut assez différent selon les matières. Bien que les premiers cours orientés développement (SLAM) me parussent simples ou déjà acquis, les cours orientés Systèmes/Réseaux (SISR) me mirent rapidement en difficulté. Petit à petit, j'ai compris que les cours de développement et base de données m'intéressaient plus que les autres cours (certainement en partie grâce à mes facilités). Ainsi, j'ai pris l'option SLAM pour acquérir des connaissances dans ces matières où je portais un intérêt certain. Pour la recherche de mon stage, j'avais donc pour objectif de trouver une entreprise qui travaillait dans le développement. Antoine est un contact à moi, et apprenant qu'il commandait une petite entreprise de développement web, c'est tout naturellement que je lui ai demandé un stage. Il accepta, et me voilà à présent dans cette équipe de Xev, à travailler avec les autres développeurs sur des projets concrets.

Contexte

L'entreprise Xev

L'entreprise Xev est une entreprise de développement web fondée en 2018. Elle est aujourd'hui menée par Antoine Hervet. Elle possède à ce jour sept employés, et a travaillé pour des entreprises diverses, comme la société Dalbe ou encore la startup Mon Livret C.

Dans l'entreprise, l'organisation se fait de manière assez classique, avec un chef de projet qui s'occupe de connaître et rapporter les besoins du client, tester les fonctionnalités des produits et demander les modifications nécessaires. Ces demandes de modifications sont gérées par l'équipe de développeurs, menée par un tech-lead. C'est dans cette équipe de développement que je j'évoluai.

Mon travail

L'ensemble des projets sur lesquels j'ai travaillé appartiennent au SI de l'entreprise Dalbe. Celle-ci a souhaité confier à Xev la reconstruction de son système. En effet, l'entreprise possédait des logiciels obsolètes, comme un logiciel de management, l'AS400, remplacé aujourd'hui par les ERP.

L'équipe de Xev s'attèle donc à refaire ce SI, qui comporte à ce jour une boutique vitrine, un extranet pour les magasins, ainsi que d'un lien vers les progiciels Pongo

(fidélisation de clientèle) et ProGmag (encaissement). Les outils utilisés sont Strapi et Angular pour l'extranet, Shopify pour la boutique, et le logiciel open-source Axelor, basé sur Spring et PostgreSQL, pour l'ERP (Axelor sert de base modifiable selon les besoins). Pour l'hébergement de la base de données, Xev Google Cloud.

Ainsi je n'ai pas eu à proposer d'outils. Ils me furent naturellement imposés, arrivant sur des projet en cours. Concernant Angular, je fus assez étonné de le voir encore utilisé, sachant que de nombreux et puissants concurrents existent sur le marché (Vue, React, Svelte, Solid…). Angular est le plus complexe des frameworks javascript frontend, mais également le plus complet. Il utilise par ailleurs Typescript par défaut, contrairement aux autres, ce qui permet de limiter les erreurs de typage. C'est pour ces raisons que Xev l'utilise. Quant à Strapi, bien sûr il y a comme concurrent principal les frameworks JS backend comme Express.js, mais également des CMS back-office comme Directus. L'avantage de Strapi est cependant pratique pour la modification de son code pour l'adapter à des besoins spécifiques.

Ainsi, mon stage pourrait se résumer à la participation à des projets de développement web.

Projet et activités

La technique

Traitement du CSV avec Python (annexe 1)

Dans plusieurs projets au cours de mon stage, j'eus à comparer et trier une quantité importante de données. Pour optimiser ces tâches, j'ai écrit des scripts Python pour trier, comparer et rassembler des données.

La plupart des données étaient sous forme de fichiers CSV. Ayant une bonne connaissance de la manipulation de CSV en Python, j'ai conçu des scripts pour automatiser le tri. Je pense que la principale difficulté n'était cependant pas le script en lui-même, mais de comprendre ce que l'on nous demande de faire.

Le premier de ces algorithmes (annexe 1) que j'eu à écrire concernait mon troisième ticket : L'énoncé parlait notamment d'un fichier CSV à créer, dans lequel il fallait insérer des numéros de registre comptable associés à une société. Il y avait déjà des fichiers similaires pour les deux autres sociétés du projet. Mais je m'aperçus que ces fichiers possédaient plus d'information que ce dont je disposais. Entre autres, chaque registre est associé à un registre parent, permettant d'organiser les registres dans l'application. J'ai dû ainsi utiliser les fichiers existants pour voir ceux qui correspondaient

au fichier que l'on m'avait fourni pour ainsi récupérer leurs informations (et donc leur parent – certains n'étaient pas présents, je les ai laissés tels quels). Enfin, j'ai rajouté les registres parents dans le fichier (et les parents de ces parents, etc.).

La difficulté ici réside également dans la compréhension de la tâche. Heureusement, je savais ce qu'était un registre comptable et qu'il y avait des catégories. Dans le cas contraire, je pense que la tâche m'aurait demandé plus de temps pour bien comprendre l'objectif.

Extranet Strapi-Angular

L'une de mes activités fut de travailler avec Strapi. Strapi est un CMS pour la gestion de données écrit en TypeScript. Open-source, son code est fait pour être adapté aux besoins des développeurs (il s'agit d'un CMS low-code). Xev l'utilise pour des grosses applications, ou pour permettre à différents logiciels de se partager les données. Dans le cas où je l'ai utilisé, il s'agissait d'un extranet composé d'un frontend Angular relié à un backend Strapi. Ce dernier est lui-même relié au progiciel Pongo, qui permet de gérer la fidélisation client. Quant à la base de données, il s'agissait d'une base PostgreSQL, stockée sur Google Cloud, accessible via un proxy. Autant de technologies que j'ai dû comprendre et manipuler pour mener mes missions à bien.

Une de mes activités fut de configurer une erreur particulière dans l'application : chaque client présent dans la base de données de Strapi est relié à un compte Pongo. Cependant, l'identifiant d'un compte est le numéro de téléphone du client. Or certains clients n'avaient pas donné leur numéro lors de la création de leur compte client. Ainsi, ces clients sont à relier un à un à un compte Pongo unique. Problème : on peut relier plusieurs utilisateurs au même compte Pongo. Le ticket que je dois compléter a pour objectif de régler ce problème.

Ici, deux éléments furent complexes : le premier était de bien configurer les liens entre les différentes applications. Je me suis embrouillé plus d'une fois et mon maître de stage me vint en aide pour m'aider à prendre du recul. L'une des manières qu'il me commanda de faire était en premier lieu un schéma, avec les informations de connexions de chaque élément de l'extranet. Finalement, cette méthode fonctionna et mes idées furent plus claires. Mais je dû cependant demander de l'aide à une des développeuses qui connaissait bien Strapi. Car seul l'expérience et la pratique d'un outil permet d'avoir une "intuition", très utile dans les domaines techniques comme le développement.

Conduite de projet

L'équipe de Xev fonctionne avec une méthode agile, où différents sprints sont planifiés ; ces sprints sont divisés en plusieurs User Stories ("tickets"). Ces tickets sont généralement énoncés par le chef de projet, qui sont ensuite répartis entre les

développeurs. Pour gérer cette méthode, nous utilisons le site Jira, qui propose une interface Kanban.

Concernant l'intégration des modifications au projet principal, nous utilisons GitHub. J'appris notamment ce qu'était une "Pull Request" (ou "PR") et la méthode à suivre pour la faire valider. A chaque correction, avant de valider le ticket, il faut qu'il soit testé par le chef de projet pour vérifier que cela corresponde à ce qui était demandé par le client. De la compréhension à la validation du ticket, le temps de travail peut varier de quelques heures à une semaine.

Tous les matins, une réunion avec le chef de projet et les développeurs est organisée, pour que chacun explique ce qu'il a fait la veille et ce qu'il prévoit de faire aujourd'hui. Celle du lundi est particulière, car cette fois-ci il s'agit de résumer ce qui a été fait la semaine dernière et de prévoir la semaine qui arrive.

Il y a régulièrement une réunion technique : le chef de projet, qui répertorie les tickets et organise les sprints, répond aux différentes questions des développeurs à propos des consignes de tickets. On m'expliqua que, de manière générale, ces consignes sont vagues et nécessitent des précisions pour mieux comprendre ce que l'on doit faire.

Exemple de projet (Annexes 3 à 7)

Durant la première semaine de mon stage, on m'assigna plusieurs tickets concernant un projet d'ERP (gros logiciel de gestion). Destiné à l'entreprise Dalbe, le projet est basé sur l'application open-source Axelor, développé avec Spring, Gradle et PostgreSQL. Je souhaite détailler ce projet, car il m'a permis de bien comprendre les compétences essentielles dans le monde du développement.

Les ERP (Entreprise Ressources Planning) ont la particularité d'être des applications très lourdes, car ils doivent permettre de gérer toute l'activité interne d'une entreprise (gestion des stocks, RH, ventes, factures, personnel, contrats...). Ainsi, développer de bout en bout ce type de logiciel nécessite de gros moyens en termes de main d'œuvre et d'expertise. Xev utilise donc comme base l'ERP open-source d'Axelor. Cela leur permet de s'adapter aux besoins du client (en modifiant les données et les fonctionnalités) tout en ayant une application fonctionnelle dès le départ.

Ces travaux consistent plus en de la recherche que du code (ces premiers tickets ne consistaient pas en la modification d'un algorithme, l'ajout d'un contrôleur Spring, ou encore de la modification d'une page HTML. La principale difficulté est de trouver où est l'information à changer. Pour cela, j'observais l'application, repérait les variables à changer, puis cherchais leur nom dans le projet. IntelIJ, IDE avec lequel j'ai travaillé sur le projet, permet d'effectuer des recherches rapides et précises dans tout le projet.

L’annexe 6 propose une documentation utilisateur pour le module « comptabilité » du projet, celui sur lequel j’ai principalement travaillé. L’annexe 7 présente une documentation technique globale de l’application.

Conclusion

Ce stage m’a permis de découvrir le monde de l’entreprise. J’ai pu approfondir mes connaissances, mon expérience et ma capacité à trouver et résoudre des problèmes. J’ai pu aussi expérimenter de nouvelles méthodes de travail pour être plus productif. Je pense que ce que j’ai appris me servira pendant la suite de ma formation et dans le monde professionnel. Ces méthodes de travail, il faudra que je les peaufine et que je les intègre. Cela implique une meilleure organisation, une meilleure gestion du temps. Pour de prochains projets, je compte réutiliser les technologies que j’ai découvert, notamment PostgreSQL et Strapi, qui sont des outils très puissants.

Annexes

Annexe 1 : Algorithme en python de traitement de fichiers CSV

Contexte (anonymisé) :

La société Lopard possède un logiciel de gestion. Dans ce logiciel, deux sociétés y sont enrgistrées : Lopard et Lopard-SCI (SCI : Société Civile Immobilière, pour la gestion des bâtiments). A chacune de ces sociétés est rattaché un certain nombre de registres comptables (numéro enregistrant une certaine transaction de bien en comptabilité).

L'ensemble de ces registres sont enregistrés dans les fichiers CSV suivants.

- registres_lopard.csv
- registres_lopardSCI.csv

Chaque ligne contient le code du registre, le code de son registre parent (ces registres parents peuvent être vus comme des catégories de registre). Ces parents servent à organiser les comptes dans le logiciel.

Exemple :

code ; codeParent ; nomRegistre

0 ; ; REGISTRE RACINE
16 ; 0 ; INTERETS COURUS
163 ; 16 ; AGENCEMENTS INSTALLATIONS
163005 ; 163 ; AGENCEMENT BURO LONRAI

Cependant le fichier de lopardSCI est incomplet : on n'y trouve pas les codes parents.

Objectif :

Ici, nous allons supposer que tous les registres de lopard-SCI se trouvent dans ceux de lopard.

L'objectif est de compléter le fichier CSV de lopard-SCI. Pour cela, nous créeons un troisième fichier final.csv que nous allons remplir. Il y a deux étapes :

- Pour chaque registre de registres_lopardSCI.csv, retrouver le registre dans registres_lopard.csv et insérer la ligne dans final.csv.
- Pour chaque registre de final.csv, retrouver le registre parent (et le parent du parent etc.) dans registres_lopard.csv et l'insérer dans final.csv.

Pour des questions de simplicité, on supprime les doublons à la fin de l'algorithme.

Préparation :

```
import csv

# ouvrir les fichiers
lopard = open('registres_lopard','r')
lopardSCI = open('registres_lopard.csv','r')
finalCSV = open('final.csv','w')

# récupération des informations
lopard_reader = csv.reader(dalbe, delimiter=';')
lopard_list_lines = []
for line in lopard_reader:
  lopard_list_lines.append(line)

lopard_SCI_reader = csv.reader(lopardSCI, delimiter=';')
lopard_SCI_list_lines = []
for line in lopard_SCI_reader:
  lopard_SCI_list_lines.append(line)

writer_account = csv.writer(finalCSV) # création du module d'écriture

#############
# algorithme #
#############

finalCSV.close()
lopard.close()
lopardSCI.close()
```

Première Etape :

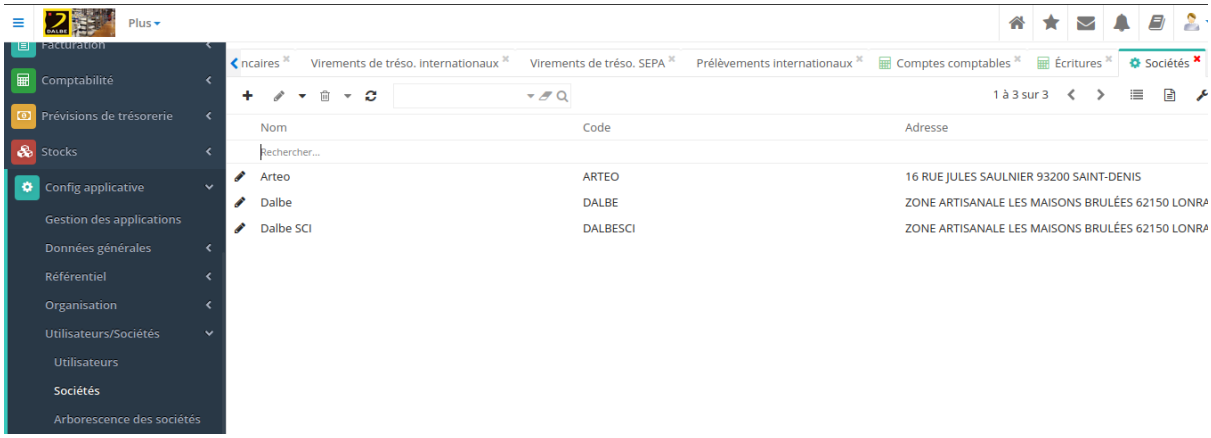
```
for line_sci in lopard_SCI_list_lines[1:]: # on saute la première ligne de description
  found = False
  for line_lopard in lopard_list_lines[1:]:
    if line_lopard[0] == line_sci[0]:
      writer_account.writerow(line_lopard)
      found = True
      break
  if not found : writer_account.writerow(line_sci)
```

Seconde Etape :

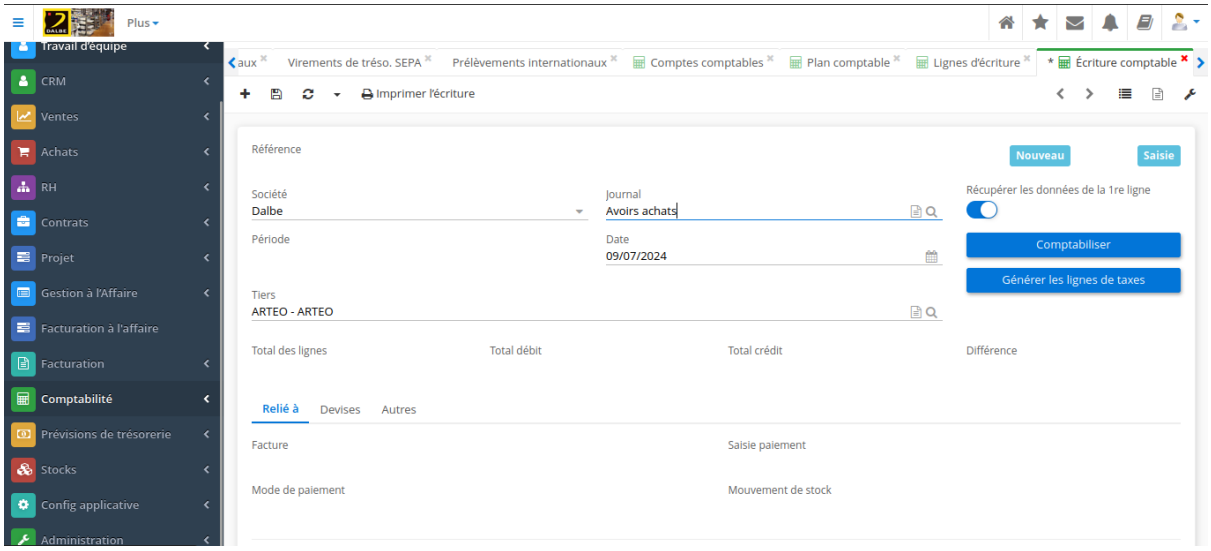
```
parentList = []
def getListParentCode(codeLine, liste_registres):
    parentList.append(codeLine[1])
    for parentCodeLine in liste_registres:
        if codeLine[1] == parentCodeLine[0]:
            getListParentCode(parentCodeLine,liste_registres)
            return parentList
    return [codeLine[1]]

# liste complète des parents
finalListParent = []
for i in range(len(dalbe_SCI_list_lines)):
    finalListParent.extend(getListParentCode(dalbe_SCI_list_lines[i],dalbe_list_lines))

# suppression des doublons
dictionary = dict.fromkeys(finalListParent)
finalListParent = list(dictionary)

# écriture dans le CSV
for line in dalbe_list_lines:
    if line[0] in finalListParent:
        writer_account.writerow(line)
```

Annexe 3 : Capture d'écran du plan comptable dans l'ERP

Annexe 4 : Capture d'écran des sociétés dans l'ERP



Annexe 5 : Capture d'écran du formulaire d'ajout d'une écriture comptable dans l'ERP



Annexe 6 : extrait de la documentation utilisateur du module « comptabilité » de l'ERP

Où trouver le module « Comptabilité »

Lorsque vous vous connectez sur l'application, le module comptabilité se trouve dans la barre de navigation à gauche, avec un logo vert.

Les fonctionnalités du module

Le module comporte quatre grandes « sections » :

- La configuration comptable
- Les écritures comptables
- Les rapports
- Le traitement des opérations comptables

Principaux onglets du module

Onglet comptes comptable : vous pouvez retrouver tous les comptes, organisés par sociétés et par familles. Si vous souhaitez ajouter un compte, cliquez sur la croix en haut à gauche de la fenêtre. Si vous souhaitez modifier un compte, cliquez dessus une fois puis cliquez sur le stylo en haut à gauche de la fenêtre.

Onglet Ecritures : vous y trouverez tous les outils pour ajouter, modifier et supprimer des écritures comptables ainsi que pour les associer à vos sociétés.

Onglet Journaux : configurez ici les journaux par défaut qui seront utilisés lors de la génération d'écritures. Lorsque vous générez une écriture depuis une facture client, elle sera générée par exemple dans le journal de Ventes.

Onglet Assistant paiements : cet onglet vous permet de gérer les différentes étapes à entreprendre lorsque vous avez un retard de paiement.

Annexe 7 : Documentation technique de l'ERP

Introduction

Cette documentation technique présente les principales technologies utilisées pour le développement de l'ERP commandé par la société Dalbe, détaille la structure du projet ainsi que les possibilités de personnalisations.

Technologies utilisées

Le projet est basé sur Axelor. Axelor est un projet open-source français codé avec le framework web Spring et Gradle. Il s'agit d'un ERP possédant plusieurs « modules » installables selon les besoins du client. Il permet une marge de manœuvre low-code pour s'adapter aux besoins des utilisateurs.

L'ERP développé par Xev utilise une base de données PostgreSQL hébergée sur Google Cloud, avec un encryptage en SHA-256.

Structure de base du projet

Voici la structure de base d'un projet Axelor :

```
axelor-demo
└── src
|   └── main
|       ├── java
|       └── resources
|           └── META-INF
|               ├── axelor-config.properties
|               └── persistence.xml
├── gradle
|   └── wrapper
|       ├── gradle-wrapper.jar
|       └── gradle-wrapper.properties
├── modules
├── gradlew
├── gradlew.bat
├── settings.gradle
└── build.gradle
```

Fichier « axelor-config.proprieties » : fichier de configuration du projet. C'est ici notamment que l'on fournit les informations de connexion pour la base de données.

Dossier « modules » : dossier regroupant tous les modules installables dans l'application. C'est ici principalement que l'on peut modifier les données par défaut au démarrage et contrôler l'affichage des pages via les fichiers CSV et XML.

Fichiers « gradlew.bat » « settings.gradle » « build.gradle » et le dossier « gradlew » : ensemble des fichiers de configuration de gralde, permettant de configurer et lancer le projet.

Lancer le projet

Pour lancer le projet, il faut lancer succésivement et dans l'ordre les trois commandes gradle suivantes : « generateCode », « copyWebApp », « generateLauncher » et « run ». Eventuellement, si l'on veut appliquer des changements, il parfois nécessaire de lancer au préalable la commande « clean » pour supprimer les fichiers de build.

Insertions de données par défaut

Lorsque l'on lance l'application, la plupart du code java et la base de données sont regénérés. Des modifications dans ces fichiers sont donc inutiles. Pour insérer des données, il faut se tourner vers les fichiers XML et CSV contenus dans les dossiers de modules. Ces fichiers ne sont pas générés et peuvent être modifiés pour ajouter des fonctionnalités aux pages.

Exemple

On souhaite rajouter une SCI et ses comptes comptables dans l'application au démarrage.

Dans le fichier "base_company.csv" (-> modules/axelor-opensuite/axelor-account/src/main/ressources/demo/fr/) qui contient les sociétés de l'ERP :

"importId";"name";"code";"address.importId";"partner.importId";"logo_fileName";"currency.code";defaultBankDetails.importId;"generatedMailFileName";"generatedEmailFileName";"printingSettings.importId";"partnerList.importId"

"1";"Dalbe";"DALBE";1;1;"img/axelor.png";EUR;;"courrier";"email";1;1

"2";"Arteo";ARTEO;816;1;img/axelor.png;"EUR";;"courrier";"email";1;915

On y rajoute la ligne :

"3";"SCI";"SCI";1;3;img/axelor.png;"EUR";;"courrier";email;1;3

Ensuite, on crée dans le même dossier le fichier "account_sci" et le remplit avec les comptes associés :

```
code;parent_code;name;accountType.technicalTypeSelect;company_code
0;;Racine PCG SCI;view;SCI
CL1_5;0;Comptes de bilan;view;true;SCI
CL6_7;0;Comptes de gestion;view;true;SCI
CL1;CL1_5;Comptes de capitaux;view;true;SCI
CL2;CL1_5;Comptes d'immobilisations;view;true;SCI
...
```

Enfin, on modifie le fichier "account_config.xml" pour permettre à l'application de lire le fichier "account_sci" :

```
<input file="account_account_dalbesci.csv" separator=";"
type="com.axelor.apps.account.db.Account">
  <bind to="company" search="self.code = :company_code"/>
</input>
```

Le résultat dans l'application est visible sur les annexes 3 (en bas, on peut voir que les comptes sci sont pris en compte) et 4 (la sci est bien insérée).

Ressources

Documentation fonctionnelle d'Axelor : https://docs.axelor.com/aos/accounting/ , consultée le 07/09/2024

Documentation technique d'Axelor : https://docs.axelor.com/adk/latest/dev-guide/application/index.html, consultée le 07/09/2024