Rapport Projet Atelier Logiciel

I. Cahier des charges :

Ce projet se déroulant dans le contexte de l'atelier GL (Génie Logiciel) de la faculté ESPRIT de Tunis, a pour but l'introduction des étudiant au méthodes et technique de gestion et de suivie de projet.

Dans le cas échéant, notre équipe se composant de:

- Adel Attia
- Chedi Toueiti
- Emir Hamraoui

Le sujet traité est le n°4 « Gestion des produits et composants » dont le cahier des charges est le suivant :

- Administrateur de l'application crée les comptes Responsable projets
- Le Responsable crée les composants logiciels : Titre, version, type (open source, acheté, développé en interne), Nature (exe, lib, dll, src) sa licence (GPL, LGPL,BSD,...) son coût (en Euro)
- Un composant peut avoir plusieurs versions différentes
- Le Responsable crée aussi les produits logiciels en indiquant: Titre, le client, la version, état du produit (en cours, en maintenance), liste des composants logiciels embarqués (en spécifiant à chaque fois la version correspondante)
- Un produit logiciel peut avoir plusieurs versions différentes et chacune embarque des composants différents
- Le Responsable projet peut consulter les produits logiciels en synthétisant sur leurs coûts et sur les autres critères (client, license, ...)
- Le système permet des rendus pour le Responsable sur la cartographie des produits logiciels et les composants utilisés

II. Backlog produit :

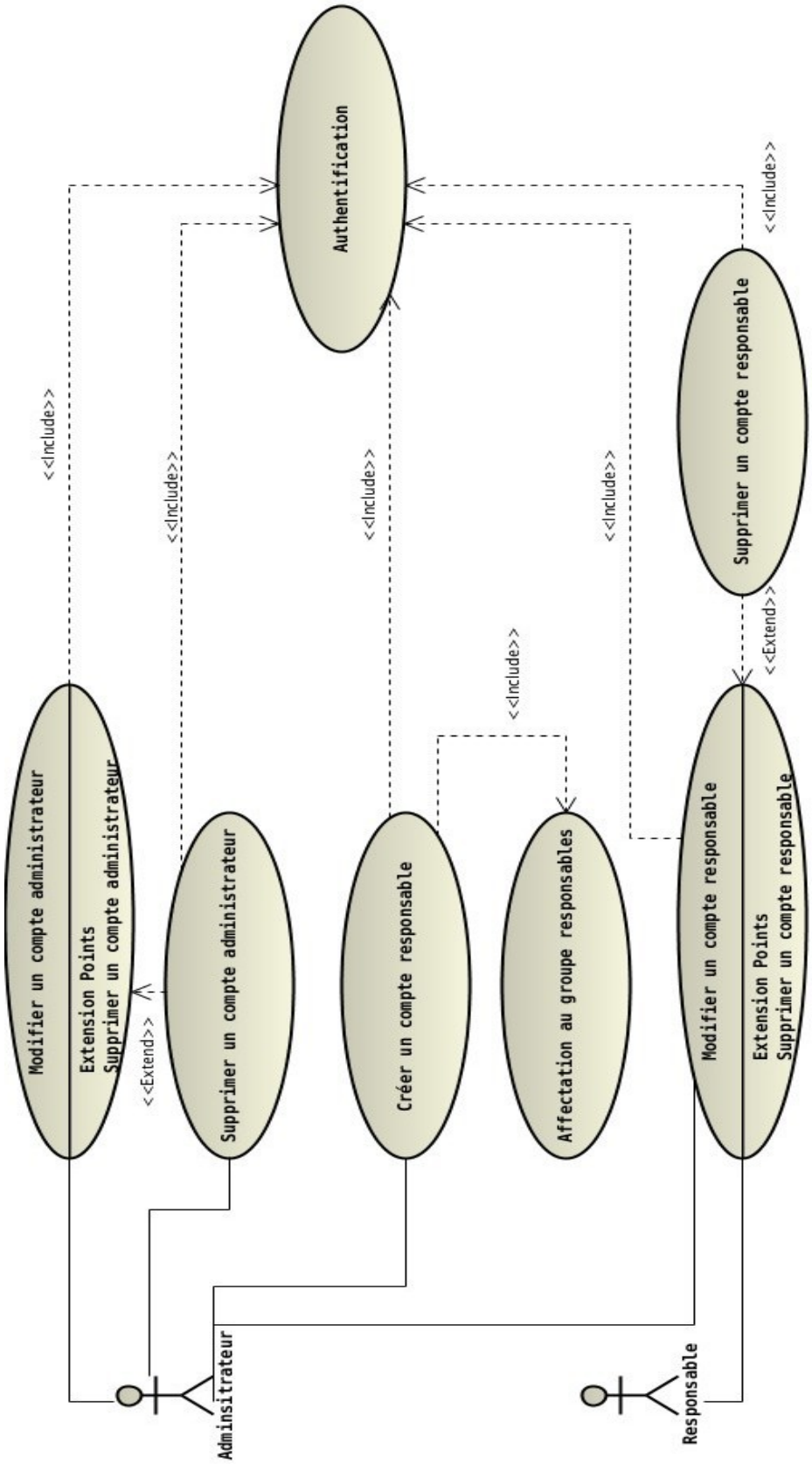
Sprint n° 1 : Analyse et modélisation du projet		
User story	**SP**	**ES**
Analyse du projet	6	4
Diagramme UC	3	2
Diagramme classes	3	2
Total	**12**	**8**

Sprint n° 2 :Conception et implémentation des modèle de données		
User story	**SP**	**ES**
Modification des paramètres du compte administrateur	2	2
Modification des paramètre des comptes responsable	2	2
Gestion des paramètres d'un composant logiciel	2	2
Suppression d'un compte responsable	2	2
Suppression d'un composant logiciel	2	2
Suppression d'un composant logiciel	2	2
Création de compte responsable projet	3	3
Authentification	5	3
Ajout des composant logiciels	3	2
Gestion des versions des composant logiciel	3	2
Total	**26**	**22**

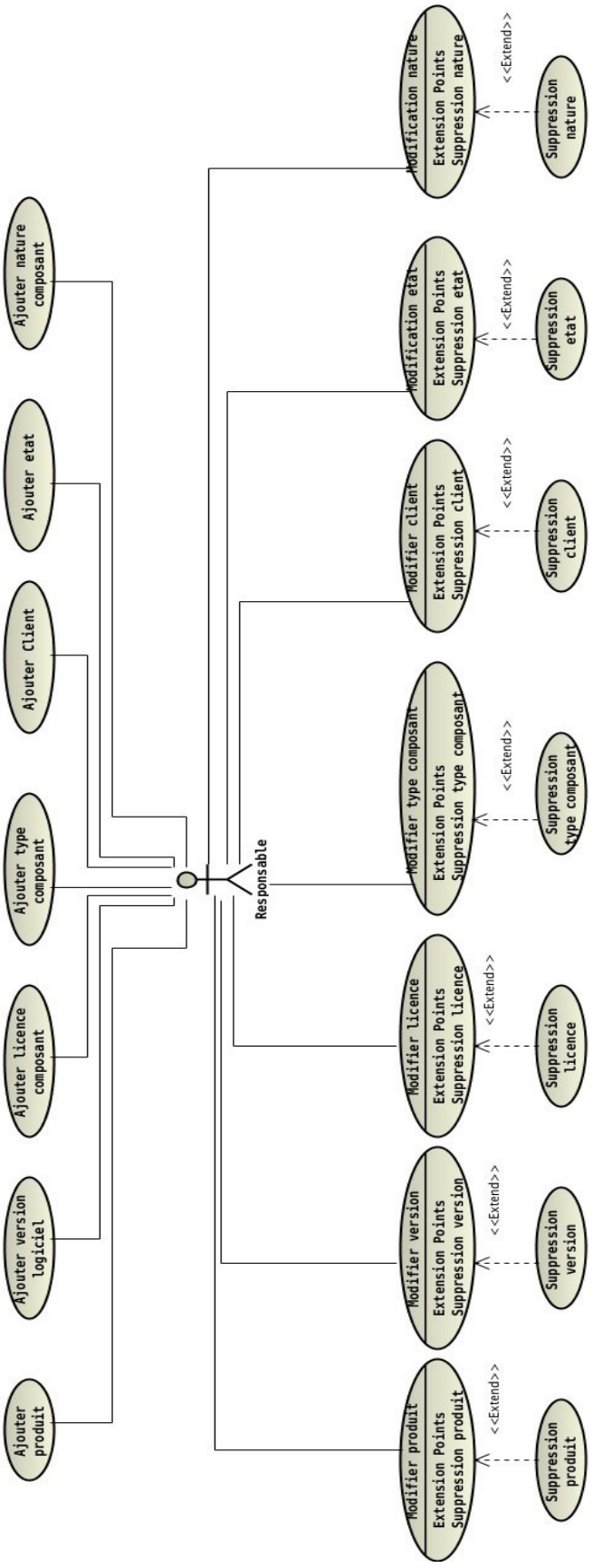
Sprint n° 3 :Conception et implémentation des modèle de données		
User story	**SP**	**ES**
Ajout d'un produit logiciel	2	2
Modification des paramètre d'un produit logiciel	2	2
Suppression d'un produit logiciel	2	2
Consultation des produit logiciels	2	2
Génération rapport produit	4	4
Génération rapport composant	4	4
Total	**16**	**16**

III. Diagramme Use Case :

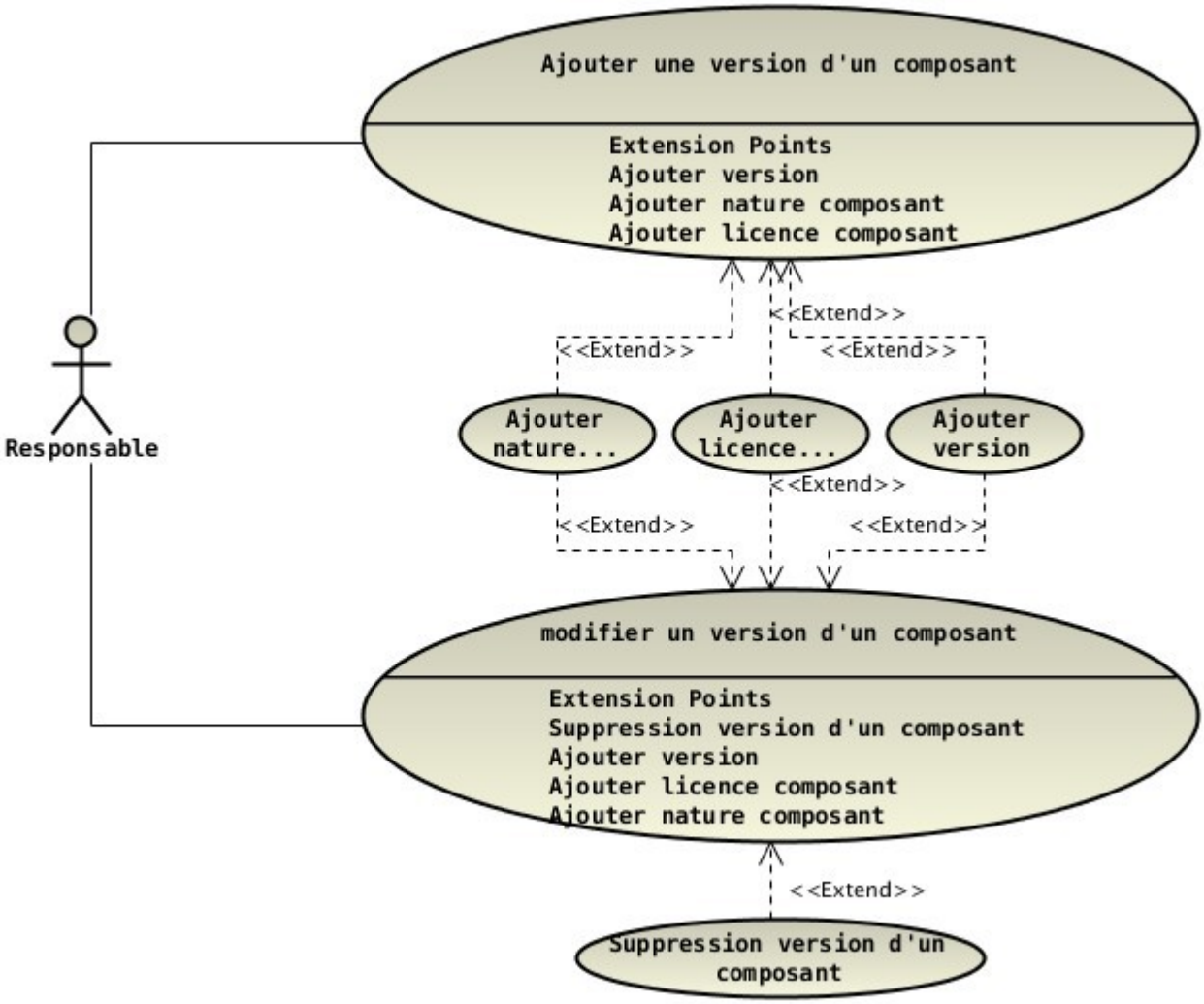
1. Authentification :



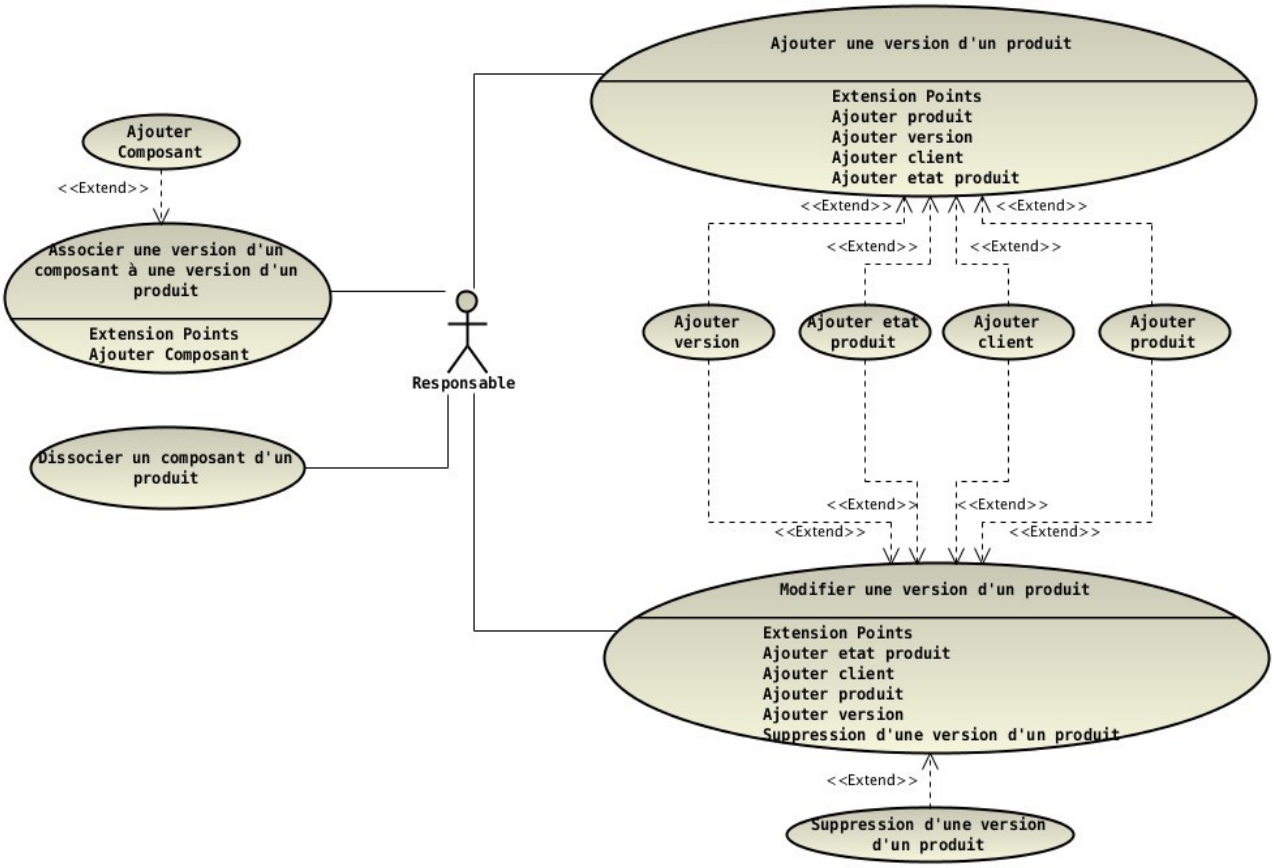
2. Configuration des listes :



3. Gestion composants :

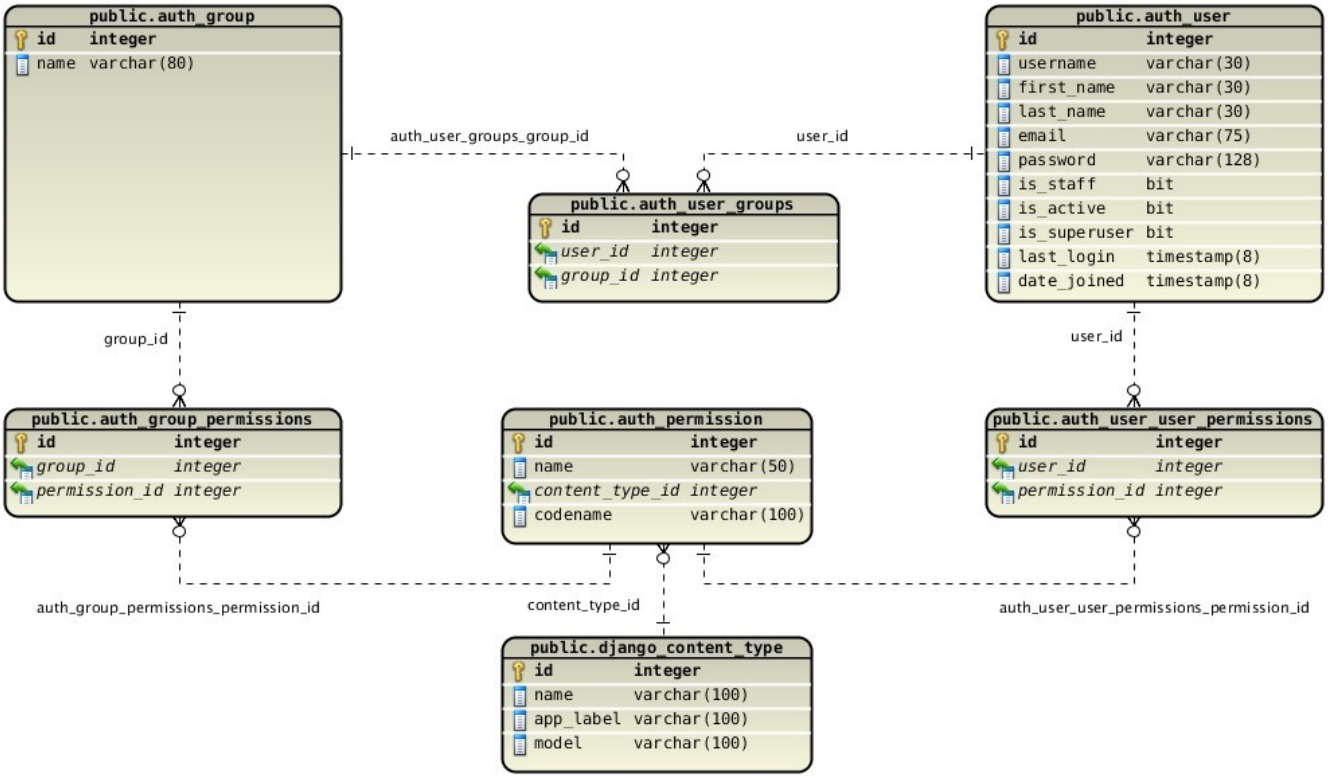


4. Gestion produits :

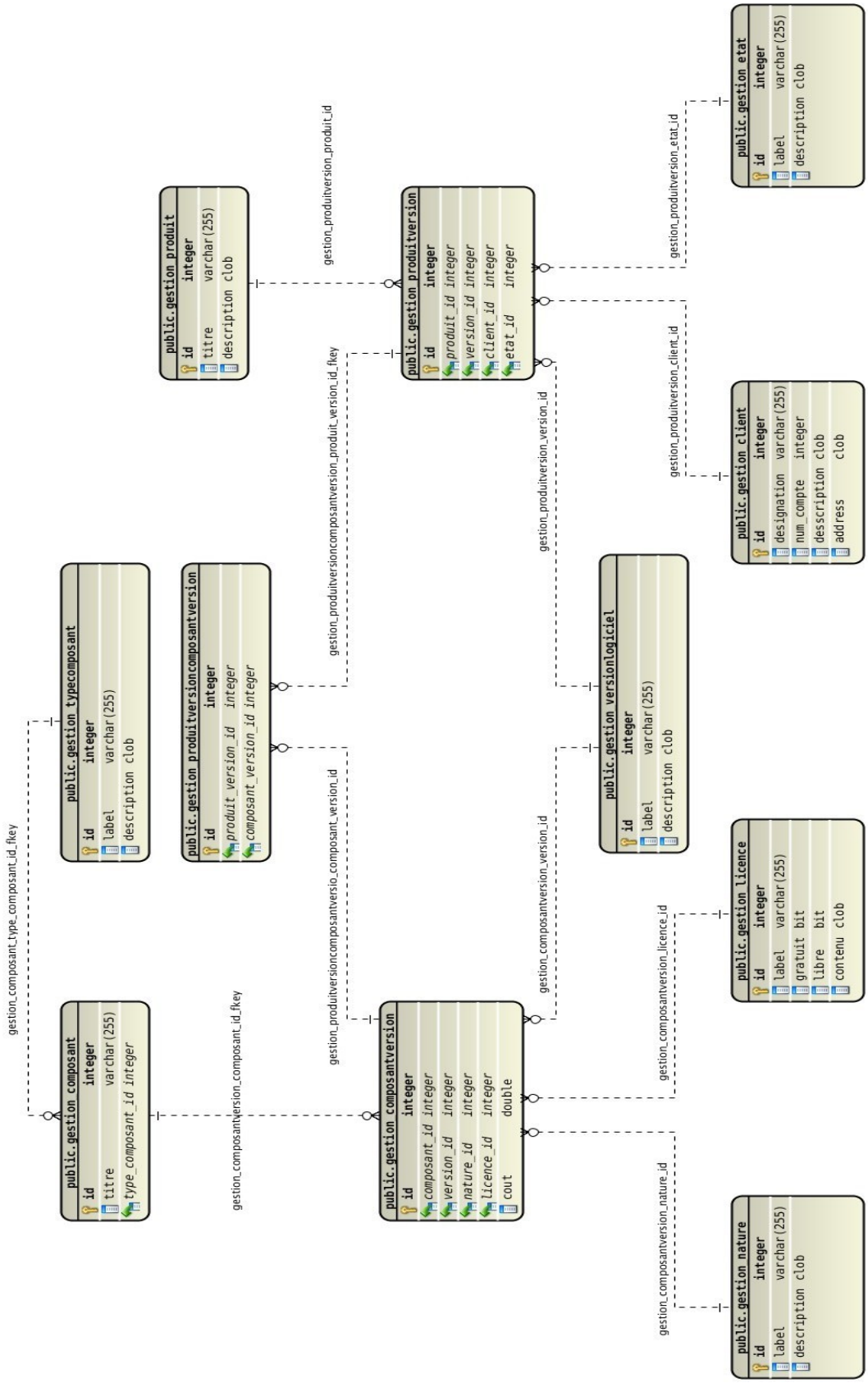


IV. Diagramme de classe global :

1. Authentification :



2. Gestion :


public.gestion produit
id integer
titre varchar(255)
description clob
public.gestion produituersion
id integer
produit_id integer
version_id integer
client_id integer
etat_id integer
public.gestion etat
id integer
label varchar(255)
description clob
public.gestion client
id integer
designation varchar(255)
num_compte integer
desscription clob
address clob
public.gestion typecomposant
id integer
label varchar(255)
description clob
public.gestion produituersioncomposantversion
id integer
produit_version_id integer
composant_version_id integer
public.gestion versionlogiciel
id integer
label varchar(255)
description clob
public.gestion composant
id integer
titre varchar(255)
type_composant_id integer
public.gestion composantversion
id integer
composant_id integer
version_id integer
nature_id integer
licence_id integer
cout double
public.gestion licence
id integer
label varchar(255)
gratuit bit
libre bit
contenu clob
public.gestion nature
id integer
label varchar(255)
description clob
gestion_produituersion_produit_id
gestion_produituersion_etat_id
gestion_produituersion_client_id
gestion_produituersion_version_id
gestion_produituersioncomposantversion_produit_version_id_fkey
gestion_produituersioncomposantversio_composant_version_id
gestion_composant_type_composant_id_fkey
gestion_composantversion_composant_id_fkey
gestion_composantversion_version_id
gestion_composantversion_licence_id
gestion_composantversion_nature_id

V. Environnements :

1. Environnement de développement :

Nous avons tenu dans ce projet à utiliser le plus possible des outils libres et open source, à l'exception de l'outil de gestion scum (agilo for scrum) et de l'éditeur sublime text, tout les autres outils sont disponible en téléchargement.

1. Base de données :

Le projet ne présente aucune contrainte par rapport à une base de données relationnelle quelconque, néanmoins, les bases utilisées pour développement ont été :

- Prostgresql
- MySql

Note : Tout autre base de données relationnelle peut être utilisé avec ce projet, pour soit qu'il existe un connecteur adéquat fonctionnant sous le framework django

2. Framework Django :

Ce framework python est pièce maitresse de notre développement, il se caractérise par une très bonne flexibilité et puissance, et qui plus est permet un haut degrés de productivité qui nous a permit d'avoir un prototype regroupant la majorité des fonctions demandé dans de très court lapse de temps.

Un autre aspect qui est encouragé par ce framework est le prototypage, car en utilisant le backoffice intégrés nous avons pu examiner et affiner le modèle de données à plusieurs reprise et avoir à chaque fois une plateforme réelle de test.

La possibilité d'intégrer d'autre module django dans notre application nous a aussi permit de bénéficier de fonction avancées dont la création à partir de zéro nous aurait sans doute pris trop de temps et engendrer un nombre important de bug.
Parmi ces module externe on peut citer :

- 'django.contrib.admin' : interface administration de base
- 'django_coverage : outil de génération de rapport de couverture
- 'debug_toolbar' : outil de débogage intégré
- 'autofixture' : outil de génération de fixture pour les testes unitaires
- 'tastytools' : outil de génération de documentation pour l'api REST
- 'annoying' : module de simplification de la génération des vue
- 'tastypie' : module de gestion de l'api REST
- 'chartit' : module de création de chartes

3. Éditeur sublime text 2 :

Nous avons fait le choix arbitraire d'utiliser cet éditeur de texte au lieu d'un IDE, car il présente l'avantage de la légèreté et de la simplicité. Cet aspect est peut être aussi due à une certaine connaissance du framework, qui nous permet d'outre passer un IDE de type Eclipse et son intellicense, mais aussi au fait de la démarche de développement qui se fait de façon semi-interactive dans un shell python et qui permet de détecter en temps réel ou presque les erreurs et les corriger.

Nous recommandant néanmoins l'utilisation de l'environnement Eclipse avec le plugin

PyDev pour les développeurs débutant et même expérimenté et qui ne suivent pas la même philosophie ou qui n'ont pas accès au même outils disponible sous Linux (surtout le shell qui n'est pas à la hauteur sur la plateforme Windows)

4. **Shell ipython :**
   Ce outil est un wrapper du shell python classique qui permet d'avoir des fonctionnalités avancées tel que :
   - Syntaxe highligthing : Coloration de la syntaxe
   - Autocomplete : Découverte dynamique des éléments d'objet ou de classe
   - Historique : Gestion avancé de l'historique des commandes
   - Intégration shell : Possibilité d'invocation de commande shell et l'utilisation de variables d'environnement
   - Gestion des processus : Possibilité de lancer des commandes en tâche de fond et les recharger à souhait
   - Commandes spéciale : Un nombre intéressant de commandes très pratique

5. **Scm git/github :**
   Comme les membres de notre équipe n'était pas géographiquement proche et que nous n'avions pas de serveur de développement centrale, nous avons opté pour un modèle de développement décentralisé.

   De ce fait et vue qu'on avait une assez bonne expérience avec le scm Git, nous l'avons choisi comme outil principal de gestion du code. Il nous alors semblé naturel d'avoir un super repository dans le site github pour les raisons suivantes :
   - Infrastructure : github permet de centralisé le processus de développement via git en exposant un repository initial que les membres de l'équipe peuvent forker. Il nous décharge de même de la problématique d'avoir toujours une connexion active pour partager les dernière modification entre membre.
   - Wiki : une fonctionnalité très utile du site qui pour permet de centraliser aussi la documentation et les artefacts binaires
   - Gestion des issues : Nous n'avons pas utilisé cette fonction dans ce projet, mais c'est un moins très efficace pour la gestion des tâches et des problèmes

   Notre repository git est disponible à l'URL : https://github.com/teamaac/GL

6. **Moteur de template cheetah/jinja2 :**
   Jinja2 est un des moteurs de template les plus flexible dans le monde python, c'était donc normale que nous en avions fait le choix pour la tâche de génération de ressources pour notre api REST.

   Ce développement étant néanmoins optionnel par rapport à la demande initiale, il nous permet de mettre en évidence la simplicité avec laquelle le framework django est extensible et flexible.

   La fonction ainsi développé permet d'utilisé la réflexivité du langage python pour découvrir les entités du projet et généré à la volé un fichier ressource qui expose ces entité dans un api REST en utilisé le module tastypie.
   Une fois le fichier en place il est possible de publier cet api vers l'extérieur en vue d'une utilisation dans des client légers (JavaScript, php...) ou lourds (java,.NET,c++...)

7. **Générateur de teste fonctionnel Selenium :**
   Selenium IDE est un IDE pour les tests de l'outil de tests unitaires Selenium. Il est mise en œuvre comme une extension pour Firefox, et permet d'enregistrer, d'éditer et déboguer les tests.

   Selenium IDE n'est pas seulement un outil d'enregistrement : il s'agit d'un environnement de développement intégré (IDE). L'utilisateur peut choisir d'utiliser sa capacité d'enregistrement, ou peut modifier ses scripts à la main.

   Ces principaux avantages sont :
   - Facilité d'enregistrement et de lecture
   - Reconnaissance de sélection des identifiants, les noms, ou, au besoin XPath
   - Pas à pas à travers les tests
   - Debug et fixer l'arrêt
   - Sauvegarde des tests au format HTML, Java, Ruby scripts, ou tout autre format