Le cahier de punissement

ST_ CahierPunissement_v0.0.1

Spécification Technique

Table des matières

1 PORTEE

1.1 IDENTIFICATION

Nom du projet	:	Le cahier de punissement
Nom du document	:	Spécification Technique
Doc. Anagramme	:	ST
Révision date	:	24/05/2019
Destinataires	:	Développeurs groupe 2

1.2 INTRODUCTION

Ce document a pour objectif de guider les développeurs lors de la phase de codage.

Ce projet a pour but de développer une application Android qui permet de punir un stagiaire ou un groupe de stagiaire et d'afficher toutes les punitions.

Le projet à une durée de 4 jours ouvrés et est réalisé par le groupe 2 qui est constitué de :

- Axel DUCUING (chef de projet)
- Loïc FERRANDEZ
- Saïd HATTITI
- Joël MORO
- Daniel GORIOUNOV

2 EXIGENCES TECHNIQUE

Ce chapitre contient toutes les exigences techniques du système, c'est-à-dire la structure et les règles à respecter dans l'élaboration des différents fichiers qui composeront le projet.

Le chapitre est divisé en sections regroupant les directives en fonction de leur catégorie:

- Spécification logicielles,
- Spécification données.

2.1 SPECIFICATION LOGICIELLES

2.1.1 REGLE DE NOMMAGE

Pour toute la durée du projet la règle de nommage concernant tous les fichiers sera le « *camelCase* ».

Ce qui implique que tous les noms des fichiers ou packages seront liés sans espace ni ponctuation et en mettant en capital la première lettre de chaque mot sauf au premier mot qui commence par un minuscule.

Exceptions :

- les classes doivent impérativement commencer par une capitale et ensuite utiliser le « *camelCase* ».
- Les ressources auront un « *underscore* » qui sépare les mots qui seront écrit tout en minuscule.

Package *beans*, contient les classes objets qui seront nommés :

- AdresseObject,
- GroupeObject,
- StagiaireObject,
- PunitionObject,
- TypeObject.

Package *activity*, contient les classes activités qui seront nommées :

- FicheStagiaireActivity,
- FormulairePunitionActivity,
- FormulaireStagiaireActivity,
- AccueilActivity,
- ListePunitionsActivity,
- ListeStagiaireActivity.

Package dao, contient les classes qui ont un accès direct à la base de données :

- DAO,
- AdresseDAO,
- DBHelper,
- GroupeDAO,
- PunitionDAO,
- StagiaireDAO,
- TypeDAO

Package utilitaires, contient les classes annexes :

- MyAdapterGroupe,
- MyAdapterStagiaire,
- MyViewHolderGroupe,
- MyViewHolderStagiaire,
- RecyclerTouchListener.

Dossier Ressources :

Package layout, contient les vues des différentes activités :

- activity_fiche_stagiaire,
- activity_formulaire_punition,
- activity_formulaire_stagiaire,
- activity_accueil,
- activity_liste_punitions,
- activity_liste_stagiaires,
- cell_card_groupe,
- cell_card_stagiaire.

Package menu, contient la vue du menu :

- my_menu

Package values :

Ajout d'un jeu de test nommé :

- jeu_de_test

2.1.2 STRUCTURE DES REPERTOIRE

Le projet contiendra dans le dossier « *src.main.java.com.oim.punissement* » 4 packages.

- activity,
- beans,
- dao,
- utilitaires,

Dont leurs compositions sont détaillées plus haut.

2.1.3 INFORMATION COMPLEMENTAIRE

Les beans sont une représentation des tables SQL de la base de données.

Les dao représentent des classes pouvant se connecter à la base de données. Ils sont constituaient :

- d'une classe mère abstraite DAO,
- de cinq classes concrètes pouvant manipuler les informations de la base de données, dont chacun étant associer à un bean,
- d'une classe DBHelper qui permet la connexion.

Les tests seront réalisés avec un « jeu de test » qui est composé de 30 stagiaires, 3 groupes, 3 punitions, 30 adresses, 3 types de punition.

2.2 SPECIFICATION DES DONNEES

L'objectif de ce chapitre consiste à décrire les données qui seront utilisées.

2.2.1 DESCRITPION DES DONNEES

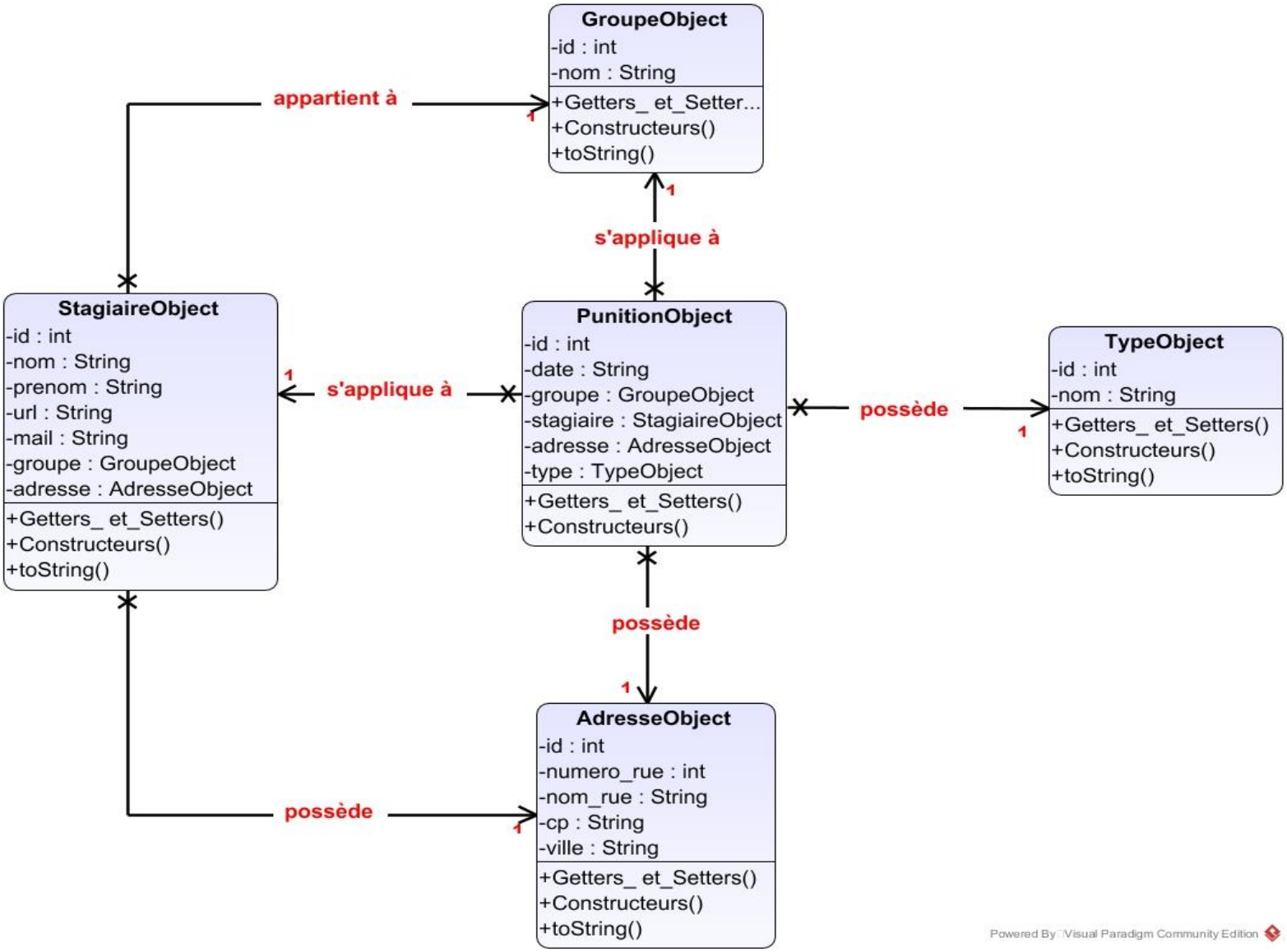
Ici sont présentées les différentes données qui constitue la base de données.

La base de données s'appellera « punissement ».

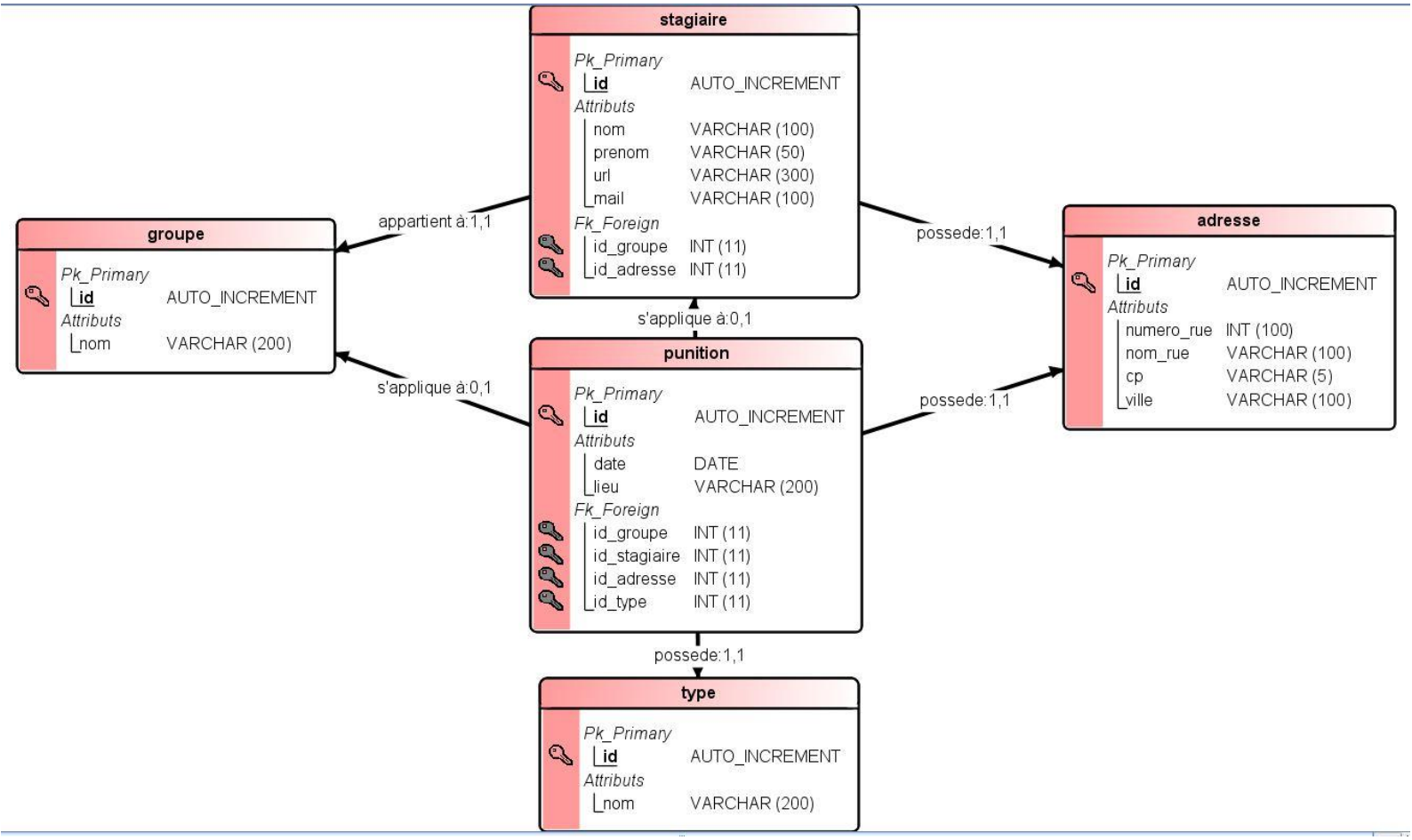
Elle sera constituée de 5 tables :

- adresse :
  - id : Identifiant unique <int>
  - numero_rue : numéro de la voie de résidence <int>
  - nom_rue : nom de la voie de résidence <String>
  - cp : code postale de résidence <String>
  - ville : nom de la ville de résidence <String>
- groupe :
  - id : Identifiant unique <int>
  - nom : nom du groupe <String>
- stagiaire :
  - id : Identifiant unique <int>
  - nom : nom du stagiaire <String>
  - prenom : prénom du stagiaire <String>
  - url : photo du stagiaire <String>
  - mail : adresse email du stagiaire <String>
  - groupe# : ID du groupe dans lequel il appartient <groupe>
  - adresse# : ID de l'adresse du stagiaire <adresse>
- type :
  - id : Identifiant unique <int>
  - nom : nom du type <String>
- punition :
  - id : Identifiant unique <int>
  - date : date de la punition (format : yyyy-mm-dd) <String>
  - groupe# : ID du groupe qui sera puni <groupe>
  - stagiaire# : ID du stagiaire qui sera puni <stagiaire>
  - adresse# : ID de l'adresse de la punition <adresse>
  - type# : ID du type de la punition <type>

Diagramme de classe qui découle de cette structure :



Model Logique de Données (MLD) par JMerise :



2.2.2 CAHIER DE FONCTION

Grâce au cahier des charges nous avons les besoins du client et nous pouvons y répondre.

Voici un tableau correspondant aux fonctions qui résulte des besoins client.

Besoins du client	Fonctionnalité résultante
Ajouter un Stagiaire	Création d'un objet stagiaire via un formulaire, tous les champs seront soumis à vérification avant la création du stagiaire. Création d'une méthode qui permettra intégrer le dit stagiaire dans la base de données. Utilisation de <*INSERT INTO*>
Consultation d'un stagiaire	Création d'une méthode qui permettra de créer un objet stagiaire avec les données récolté depuis la base de données et de l'afficher, grâce à une recherche par ID. Utilisation de <*SELECT*>
Modification d'un stagiaire	Création d'une méthode qui permettra de modifier un stagiaire dans la base de données grâce à son ID. Utilisation de <*UPDATE*>
Ajouter une punition	Création d'un objet punition via un formulaire, tous les champs seront soumis à vérification avant la création de la punition. Création d'une méthode qui permettra de récupérer l'ID du groupe ou du stagiaire pour lui appliquer la dite punition. Utilisation de <*INSERT INTO*>
Gestion des groupes	Lors de la création d'un stagiaire via le formulaire, l'utilisateur devra sélectionner le groupe d'appartenance depuis un spinner qui affichera tous les groupes présent dans la base de données. Récupération du nom du groupe dans un String, une recherche dans la base de données sera lancée avec le nom en paramètre pour récupérer l'objet groupe et ainsi l'affecter au stagiaire Utilisation de <*SELECT*>
Gestion des types	Lors de la création d'une punition via le formulaire, l'utilisateur devra sélectionner le type depuis un spinner qui affichera tous les types présent dans la base de données. Récupération du nom du type dans un String, une recherche dans la base de données sera lancée avec le nom en paramètre pour récupérer l'objet type et ainsi l'affecter à la punition. Utilisation de <*SELECT*>

Besoins du client	Fonctionnalité résultante
Gestion des adresses	Dans les formulaires de création stagiaire et punition, l'utilisateur devra indiquer l'adresse. 4 champs seront mis en place pour créer un objet adresse et ainsi recueillir : le numéro de rue, le nom de rue, le code postal et le nom de la ville. Après vérification des champs, il faudra vérifier si l'adresse n'existe pas déjà. - Si elle existe ont la récupère pour avoir un objet complet (avec ID) et l'affecter soit dans stagiaire soit dans punition, - Si elle n'existe pas, on l'ajoute à la base de données puis on la récupère pour avoir un objet complet (avec ID) et l'affecter soit dans stagiaire soit dans punition. Utilisation de <*INSERT INTO*>, <*SELECT*> et <*UPDATE*> dans le cas d'une modification d'un stagiaire
Consultation des punitions	Création d'une méthode qui renvoie une liste de punition en faisant une recherche dans la base de données. Récupération de tous les éléments dans l'ordre croissant des dates puis afficher cette liste dans un layout. Utilisation de <*SELECT* *> et <ORDER BY>