1. projet - Gestion des clefs d'une institution et historique des accès

2. Besoin

- un lieu peut être verrouillé par plusieurs serrures.
- une clef peut déverrouiller les serrures de plusieurs lieux.
- une personne est propriétaire d'une seul clef.
- une personne peut accéder à plusieurs lieux
- un lieu est composé de lieu.
- un historique des accès permet de connaître qui a accédé où et quand.
- un historique des modifications permet de connaître qui reçoit l'accès où et quand.

3. Livrables attendus

3.1. Document d'analyse

Le document d'analyse doit contenir:

- ☑ des maquettes d'interfaces (écran ou api) avec des données pertinentes et en nombre suffisant
- ☑ un modèle conceptuel de données
- ☑ un modèle logique de données
- ☑ un modèle de classes
- ☑ Le modèle logique de données doit être enrichi avec les champs d'audits. (voir SUNIER, 2016, *Modèle logique de données relationnel*, 14.2 Audit)

3.2. Script de création de la base de données

Le script de création doit:

- ☑ Supprimer les tables du domaine
- ☑ Créer les tables du domaine avec les contraintes
- ☑ Ajouter dans les tables un jeu de données de test
- ☑ Les champs d'audit sont mis à jour par l'intermédiaire de triggers et procédures stockées. Les commandes INSERT ou UPDATE ne peuvent pas directement modifier ces colonnes.

3.3. Implémentation

Écrire:

- ☑ Les classes du domaine et leurs classes de test
- ☑ Une classe `DemoData` retournant un `Map<UUID,Entite>` pour chaque entité indépendante du domaine. Ces différents Map doivent contenir les données de test.
- ☑ Des tests unitaires interrogeant la base de données à l'aide de la libraire JDBC et instanciant les classes du domaines. (éléments théorique voir APS - Rappel Java, JDBC)

3.4. DoD (Definition of Done)

Le DoD précise la rigueur souhaitée avant de passer une tâche en terminer.

Documentation:

- Les documents doivent être validés par tous les membres du Groupe

SQL:

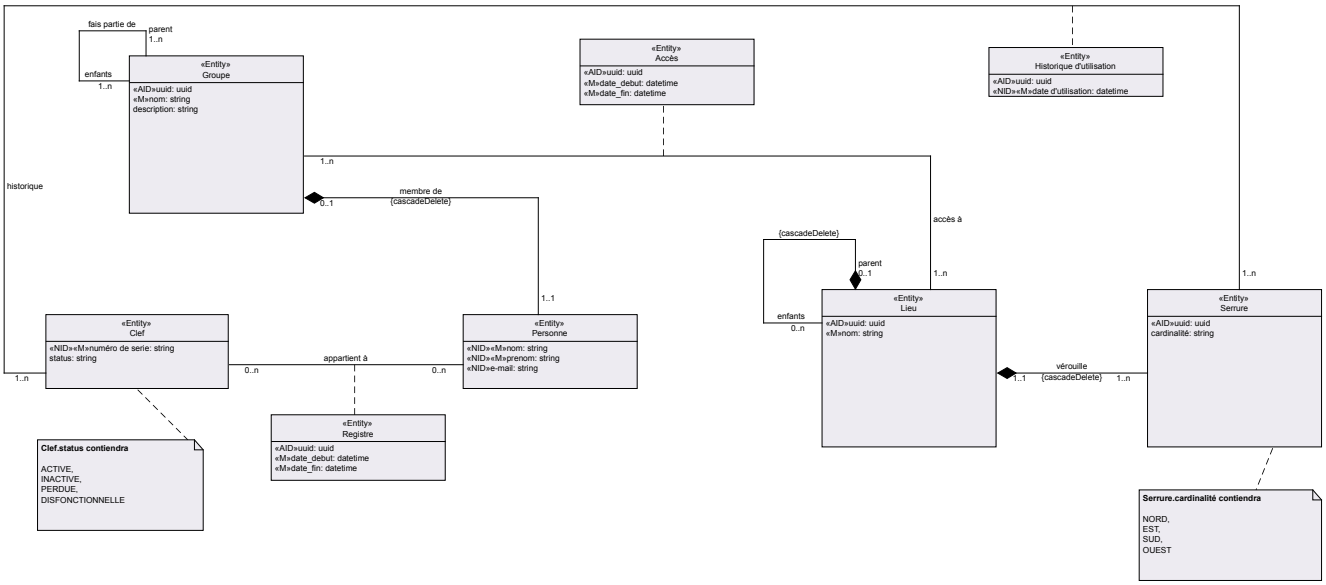
- Le MLD doit respecter les formes normales de 1 à 5
- Les scripts s'executent avec 0 erreur

Java:

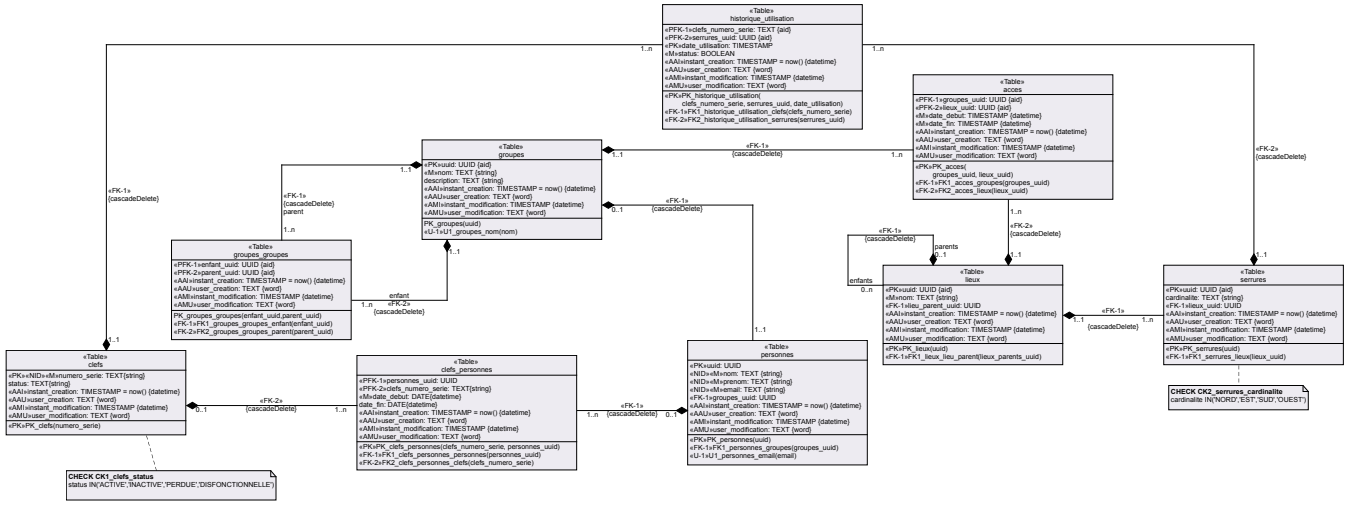
- La classe doit pouvoir build et compile sans erreur
- Les tests doivent couvrir au moins 95% du code
- Les tests doivent être 100% validés

4. Base de donnée PSQL

4.1. MCD



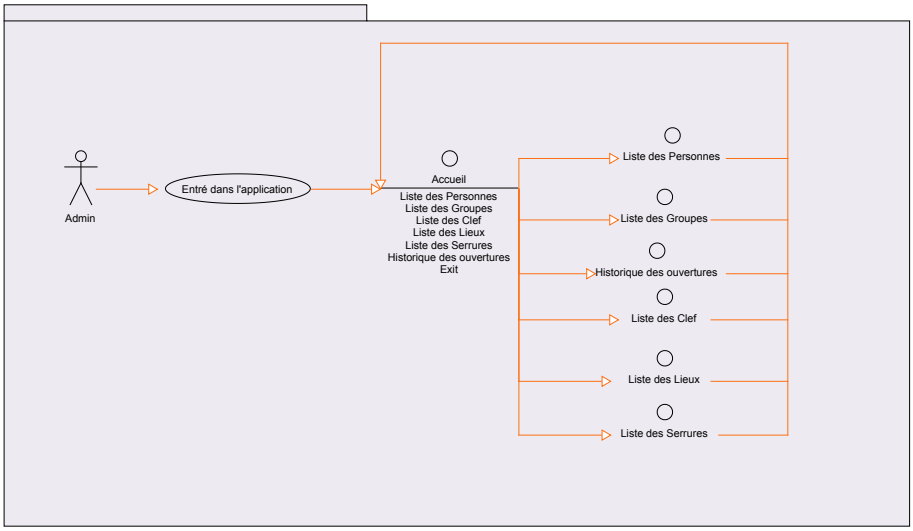
4.2. MLD

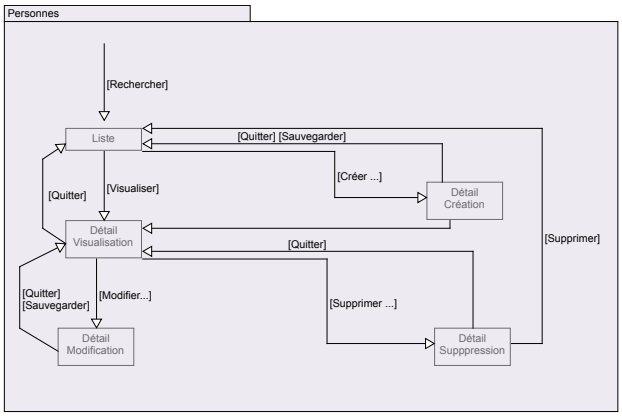


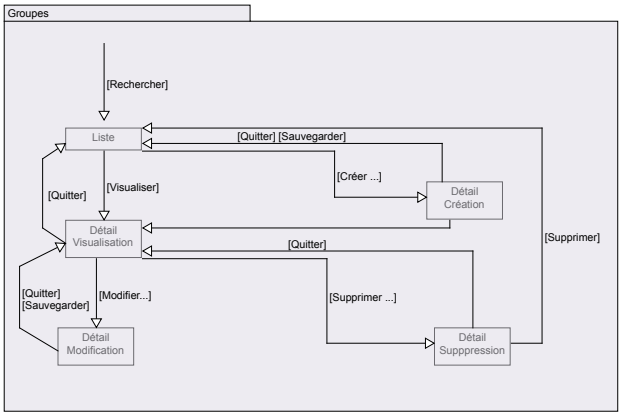
4.3. Prototypes

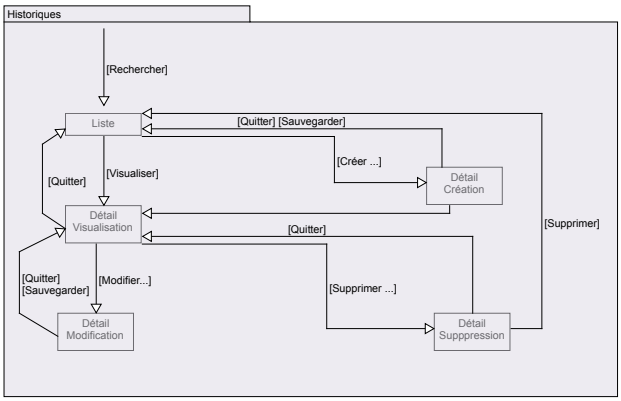
Prototypes

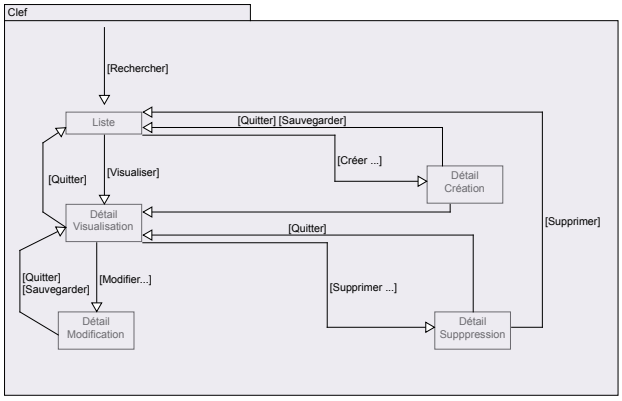
4.4. Diagramme de navigation

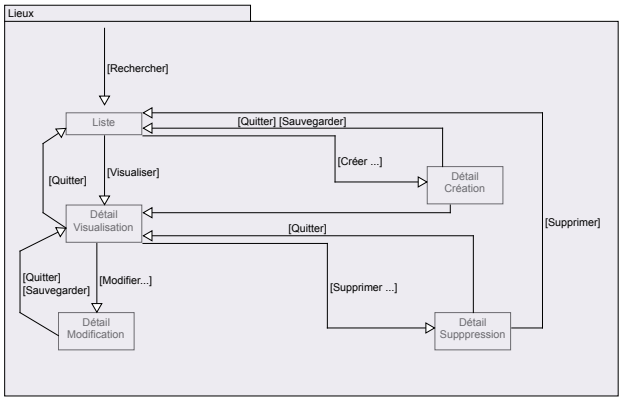

Admin
Entré dans l'application
Accueil
Liste des Personnes
Liste des Groupes
Liste des Clef
Liste des Lieux
Liste des Serrures
Historique des ouvertures
Exit
Liste des Personnes
Liste des Groupes
Historique des ouvertures
Liste des Clef
Liste des Lieux
Liste des Serrures

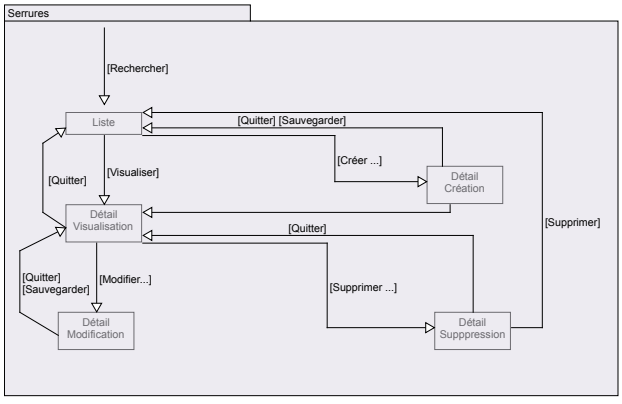

Personnes
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression


Groupes
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression


Historiques
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression


Clef
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression


Lieux
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression


Serrures
[Rechercher]
Liste
[Quitter] [Sauvegarder]
[Créer ...]
[Quitter]
[Visualiser]
Détail Création
Détail Visualisation
[Supprimer]
[Quitter]
[Quitter] [Sauvegarder]
[Modifier...]
[Supprimer ...]
Détail Modification
Détail Supppression

4.5. Scripts

4.5.1. Tutoriel de création de la base de données

- Créer une base de donnée local "clefDB"
- Créer un utilisateur "clef" avec le MDP "clefPASS"
- Ajouter l'utilisateur clef à la base de donnée clefDB
- Executer le script "script.sh"

```
./script.sh > requete.log
```

4.5.1.1. Tutoriel de création de la base de données de test

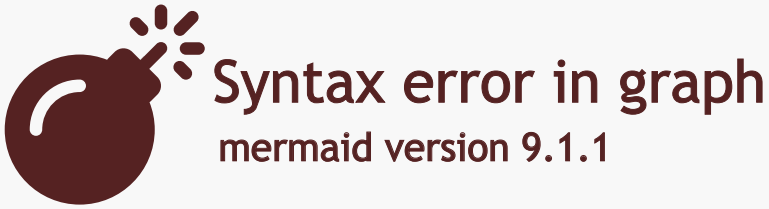
- Créer une base de donnée local "clefDB_Dev"
- Utiliser l'utilisateur "clef" avec le MDP "clefPASS"
- Utiliser l'utilisateur clef à la base de donnée clefDB
- Executer le script "scriptDev.sh"

```
./scriptDev.sh > requeteDev.log
```

5. Programme JAVA

5.1. Modèle de classe

5.1.1. Modèle



5.1.2. Controller

```
classDiagram
direction RL

class Controller{
- clefs : ArrayList~Clef~
- applicationState : State
- ds : DataSource
- connectionDB : Connection
- statementDB : Statement
- clefDB : Clef
```

```
- groupeDB : Groupe
- LieuDB : lieu
+ Controller(state : State)
+ setApplicationState(state : State)
+ controllerTearDown
+ addGroupe()
+ listGroupe() HashMap~UUID, Groupe~
+ getGroupe(findGroupe : Groupe) Groupe
+ updateGroupe(newGroupe : Groupe)
+ deleteGroupe(groupe : Groupe)
+ addGroupeGroupe(groupeParent : Groupe, groupeEnfant : Groupe)
+ listGroupeGroupe() HashMap~UUID, UUID~
+ getGroupeParent(groupe : Groupe) Groupe
+ getGroupeEnfant(groupe : Groupe) Groupe
+ deleteGroupeGroupeParent(groupeToDelete)
+ deleteGroupeGroupeEnfant(groupeToDelete)
+ addAccestoGroupeExistant()
+ listAcces() ArrayList~Acces~
+ getAcces(groupe : Groupe) Acces
+ addClef(clef : Clef)
+ listClef() ArrayList~Clef~
+ getClef(numeroSerie : String) Clef
+ updateClef(numeroSerie : String, clef : Clef)
+ deleteClef(clef : Clef)
+ addHistoriqueUtilisation(serrure : Serrures)
+ listHistoriqueUtilisation() ArrayList~HistoriqueUtilisation~
+ getHistoriqueUtilisation()
+ updateHistoriqueUtilisation()
+ DeleteHistoriqueUtilisation()
+ addLieu(lieu : Lieu)
+ AddLieuParent(lieu : Lieu)
+ listLieu() HashMap~UUID, Lieu~
+ getLieu(finLieu : Lieu) Lieu
+ updateLieu(uuidLieu : UUID, newLieu : Lieu)
+ deleteLieu(lieu : Lieu)
+ addPersonne(personne : Personne)
+ listPersonne() HashMap~UUID, Personne~
+ getPersonne(findPersonne : Personne) Personne
+ updatePersonne(newPersonne : Personne)
+ deletePersonne(deletePersonne : Personne)
+ addRegistre(clef : Clef)
+ listRegistre()
+ getRegistre()
+ updateRegistre()
+ deleteRegistre()
+ addSerrure(serrure : Serrure)
+ listSerrure() ArrayList~Serrure~
+ getSerrure()
+ updateSerrureLieu(lieu : Lieu)
+ deleteSerrure()
}
class InstanceDB{
- insert : DemoData
- controlelrdb : Controller
```

```
+ InstanceDB()
+ init()
+ getInsert(): DemoData
- insertGroupe()
- insertGroupeGroupe()
- insertAcces()
- insertClef()
- insertHistorique()
- insertLieu()
- insertLieuParent()
- insertPersonne()
- InsertRegistre()
- insertSerrure()
+ clear()
}
class DemoData
InstanceDB <-- DemoData
InstanceDB <-- Controller
```

6. Utilisation du projet

6.1. Java

- Pour utiliser le projet sur un serveur différent modifier Le fichier de propriété

```
#Données de connexion à la BD
#Nom ou adresse du serveur postgre
serverName = 127.0.0.1
#port d'accès du serveur de postgre
portNumber = 5432
#Nom de la base de donnée
dataBaseName = clefDB
#Utilisateur ayant accès à la BD
user = clef
#Mot de passe de l'utilisateur
password = clefPASS
```

Idem pour la BD de dev

6.2. Postgres

- Pour utiliser les scripts il faut modifier le script de connexion

```
#!/bin/bash
#Nom ou adresse du serveur postgres
HOST=127.0.0.1
#port d'accès du serveur de postgres
PORT=5432
```

```
#Utilisateur ayant accès à la BD
USER=clef
#Mot de passe de l'utilisateur
PASSWORD=clefPASS
#Nom de la base de donnée
DATABASE=clefDB
PSQL_SCRIPT=script.psql

PGPASSWORD=$PASSWORD psql -h $HOST -p $PORT -U $USER $DATABASE -f
$PSQL_SCRIPT
```

Idem pour la BD de test

7. Liens

SCRUM POKER

UML

JDBC

MERMAID markdown Diagramme de classe

DemoData - Fichier de démo