Projet CSC4102 : Gestion des clefs dans un hôtel

Huang ShiHui et Mabileau Paul

Année 2019–2020 — 27 janvier 2020

Table des matières

1 Spécification

1.1 Diagrammes de cas d'utilisation

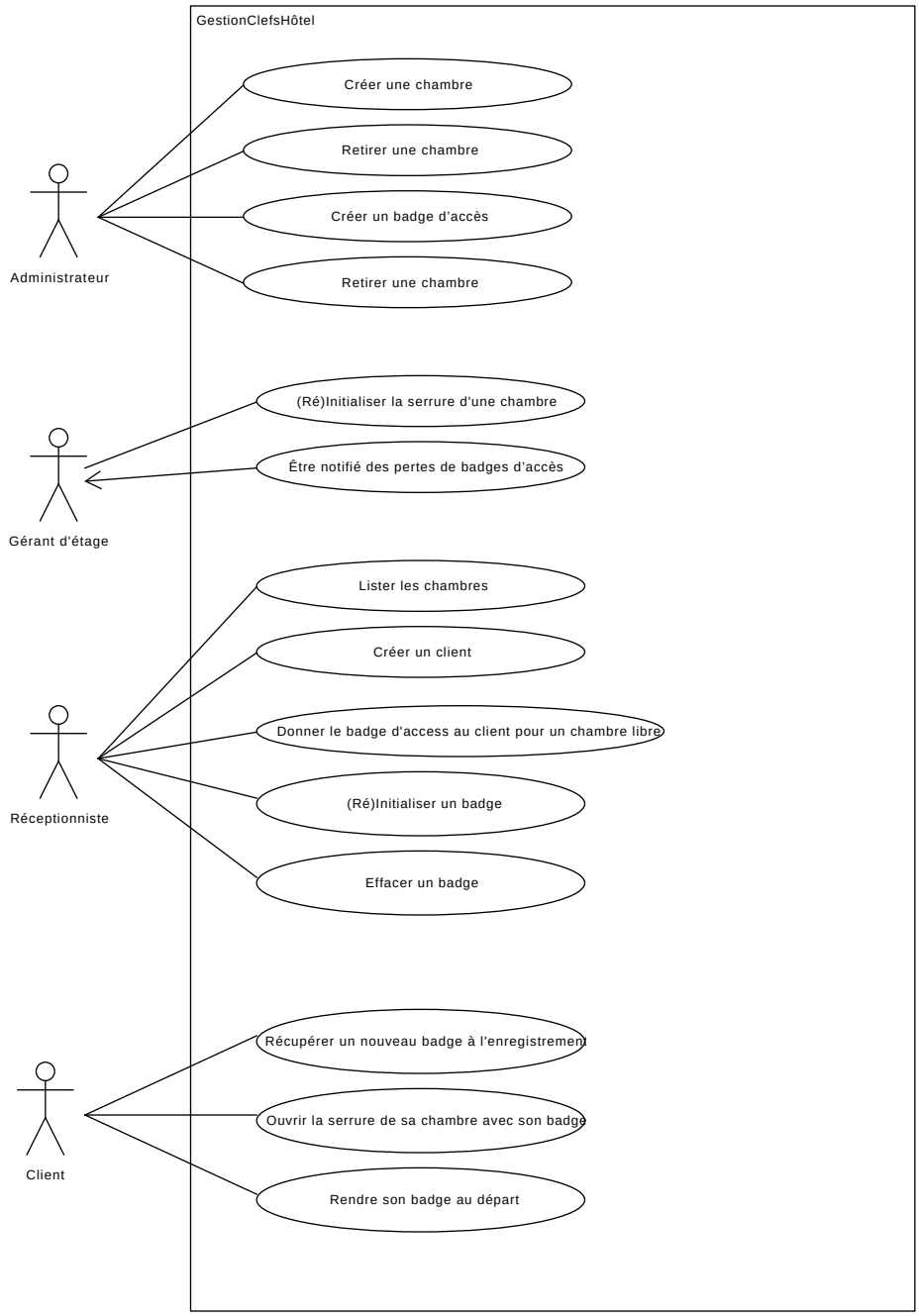

FIGURE 1 – Diagramme de cas d'utilisation

1.2 Priorités, préconditions et postconditions des cas d'utilisation

Les priorités des cas d'utilisation pour le sprint 1 sont choisies avec les règles de bon sens suivantes :

- pour retirer une entité du système, elle doit y être. La priorité de l'ajout est donc supérieure ou égale à la priorité du retrait ;
- pour lister les entités d'un type donné, elles doivent y être. La priorité de l'ajout est donc supérieure ou égale à la priorité du listage ;
- il est *a priori* possible, c.-à-d. sans raison contraire, de démontrer la mise en œuvre d'un sous-ensemble des fonctionnalités du système, et plus particulièrement la prise en compte des principales règles de gestion, sans les retraits ou les listages.
- la possibilité de lister aide au déverminage de l'application pendant les activités d'exécution des tests de validation.

Par conséquent, les cas d'utilisation d'ajout sont *a priori* de priorité « haute », ceux de listage de priorité « moyenne », et ceux de retrait de priorité « basse ».

Dans la suite, nous donnons les préconditions et postconditions pour les cas d'utilisation de priorité « Haute ». Pour les autres, nous indiquons uniquement leur niveau de priorité.

La précondition suivante est à compléter.

HAUTE n° 1
- Créer une chambre
  - précondition : identifiant/code de la chambre bien formé (non `null` et non vide) $\wedge$ chambre avec ce code inexistante $\wedge$ graine pour la génération des clefs bien formée (non `null` et non vide)
  - postcondition : chambre avec cet identifiant existante

basse
- Retirer une chambre

Moyenne
- Lister les chambres

2 Préparation des tests de validation

2.1 Tables de décision des tests de validation

La fiche programme du module CSC4102 ne permettant pas de développer des tests de validation couvrant l'ensemble des cas d'utilisation de l'application, les cas d'utilisation choisis sont de priorité HAUTE.

La section est à compléter avec les tables de décision d'autres cas d'utilisation.

Numéro de test	1	2	3	4
Identifiant/code de la chambre bien formé (non null et non vide)	F	T	T	T
Graine pour la génération des clefs bien formée ($\neq$ null $\wedge$ $\neq$ vide)		F	T	T
Chambre inexistante avec ce code			F	T
Création acceptée	F	F	F	T
Nombre de jeux de test	2	2	1	1

TABLE 1 – Cas d'utilisation « créer une chambre »

3 Conception

3.1 Liste des classes

La liste des classes suivante est à compléter.

À la suite d'un parcours des diagrammes de cas d'utilisation et d'une relecture de l'étude de cas, voici la liste de classes avec quelques attributs :

— GestionClefsHotel (la façade),
— Chambre — identifiant, graine,
— Util (classe utilitaire déjà programmée) — 'attribut de classe TAILLE_CLEF, méthodes de classe genererUneNouvelleClef et clefToString),
— ...

3.2 Diagramme de classes

Le diagramme de classes suivant est à compléter.

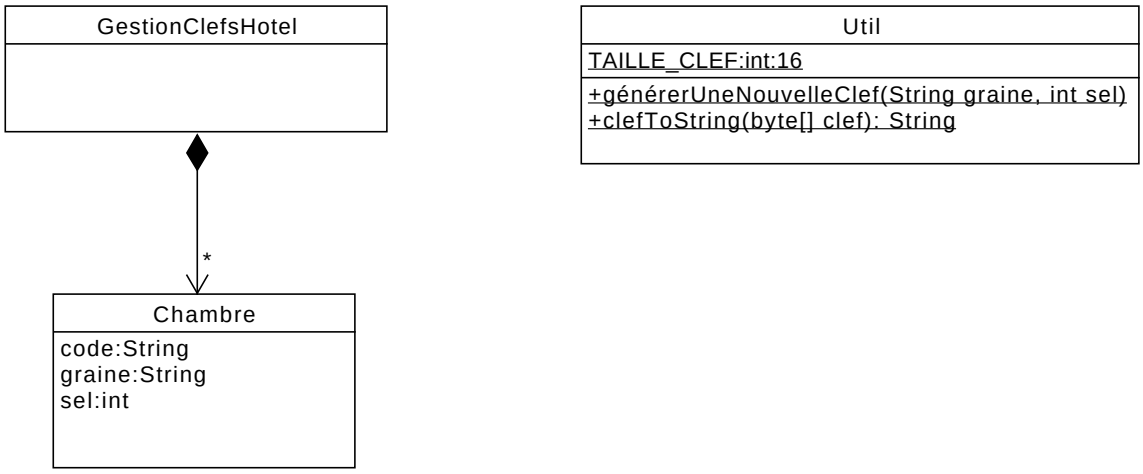

FIGURE 2 – Diagramme de classes

3.3 Diagrammes de séquence

La section est à compléter avec les diagrammes de séquence de vos cas d'utilisation les plus importants.

4 Fiche des classes

La section est à compléter avec les fiches de vos classes les plus importantes. La première fiche, celle de la façade, est aussi à compléter.

4.1 Classe GestionClefsHotel

GestionClefsHotel
<– attributs « association » –> – chambres : collection de @Chambre
<– constructeur –> + GestionClefsHotel() + invariant() :booléen **<– operations « cas d'utilisation » –>** + créerChambre(String code, String graine)

5 Diagrammes de machine à états et invariants

La section est à compléter avec les diagrammes de machine à états et les invariants de vos classes les plus importantes.

6 Préparation des tests unitaires

La section est à compléter avec les tables de décision de certaines méthodes des classes les plus importantes.