# Projet CSC4102 : Gestion des clefs dans un hôtel

Huang ShiHui et Mabileau Paul

Année 2019–2020 — 1er avril 2020

# Table des matières

# 1 Spécification

## 1.1 Diagrammes de cas d'utilisation

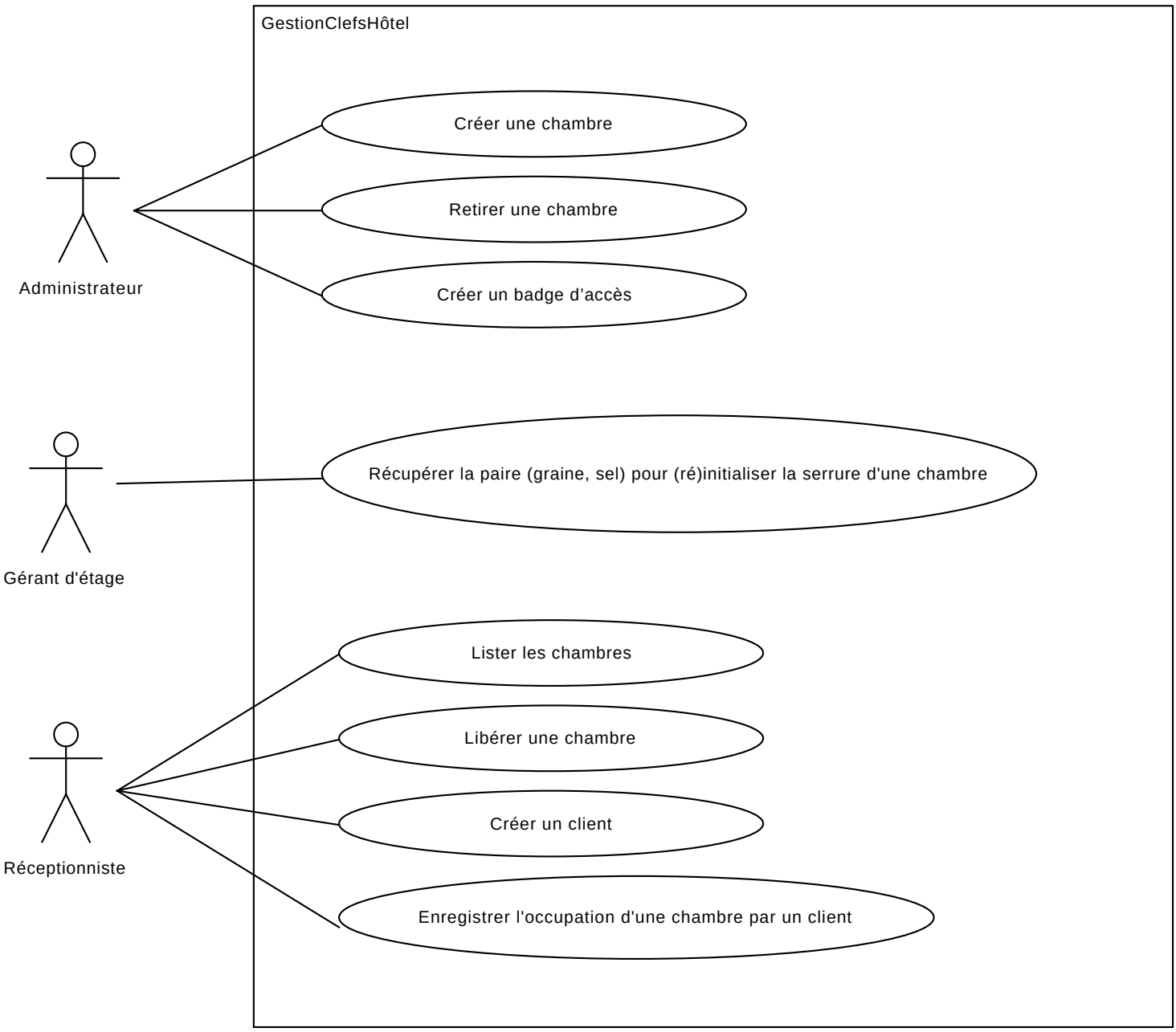


FIGURE 1 – Diagramme de cas d'utilisation du sprint 1

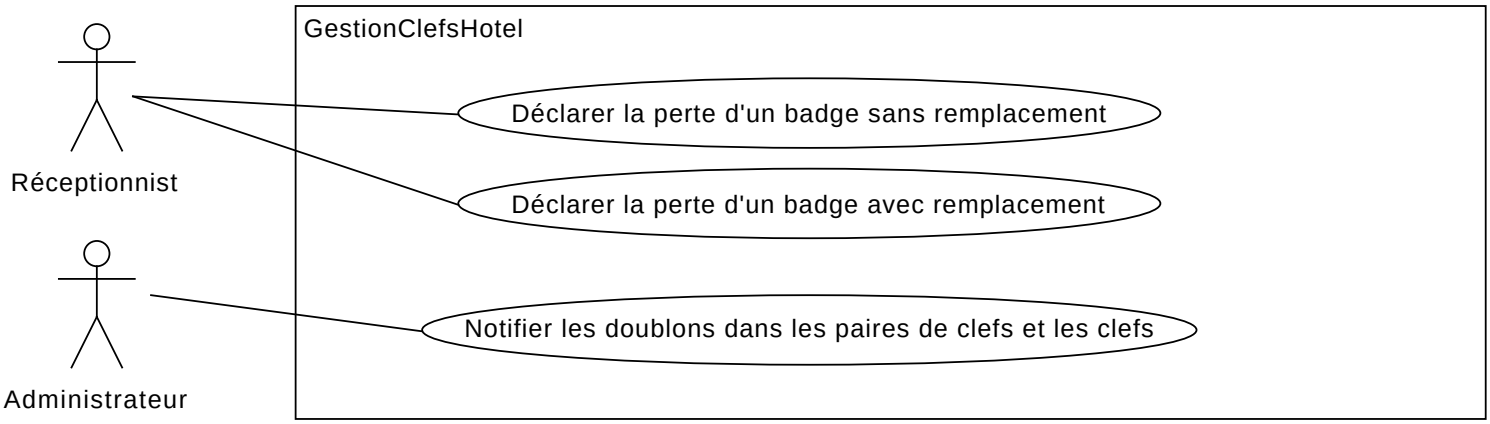


FIGURE 2 – Diagramme de cas d'utilisation du sprint 2

## 1.2 Priorités, préconditions et postconditions des cas d'utilisation

### 1.2.1 Sprint 1

HAUTE n° 1
- Créer une chambre
  - précondition :
    chambre non existant avec ce identifiant dans le système
    ∧ graine pour la génération des clefs bien formée (non null et non vide)
  - postcondition :
    chambre avec ce identifiant existante

HAUTE n° 2
- Créer un badge d'access
  - précondition :
    badge non existant avec ce identifiant dans le système
  - postcondition :
    badge avec cet identifiant existante

HAUTE n° 3
- Créer un client
  - précondition :
    client non existant dans le système
  - postcondition :
    client enregistré dans le système

HAUTE n° 4
- Enregistrer l'occupation d'une chambre par un client
  - précondition :
    ∧ client existante
    ∧ chambre existante
    ∧ le badge d'accèss existante
    ∧ client occupe aucune chambre
    ∧ chambre non occupée
    ∧ badge d'accèss disponible(badge n'associe aucun d'autre clients et chambres, paireClefs sont null)

— postcondition :
badge associe avec client et chambre
∧ chambre occupée

HAUTE n° 5 — Libérer une chambre
— précondition :
∧ client existante
∧ badge existante
∧ chambre existante
∧ client occupe une chambre
∧ client occupe la bonne chambre
∧ badge est en cours d'utilisation
∧ chambre occupée
— postcondition :
∧ vider le clef du badge
∧ disassocie les relation entre le badge et la chambre, le badge et le client
∧ chambre non occupée

HAUTE n° 6 — (Re)Initialiser la serrure d'une chambre
— précondition :
∧ identifiant de la serrure bien formé (non null et non vide)
∧ graine et sel pour la génération des clefs bien formée (non null et non vide)
— postcondition :
serrure initialisé

Moyenne — Lister les chambres

basse — Retirer une chambre

### 1.2.2 Sprint 2

HAUTE n° 7 — Déclarer la perte d'un badge sans remplacement
— précondition :
badge avec ce identifiant existant
— postcondition :
(Si le badge était en cours d'utilisation) vider le clef du badge
∧ (Si le badge était en cours d'utilisation) disassocie les relation entre le badge et la chambre, le badge et le client
∧ (Si le badge était en cours d'utilisation) chambre non occupée
∧ badge avec ce identifiant inexistant

HAUTE n° 8 — Déclarer la perte d'un badge avec remplacement
— précondition :
badgePerdu avec ce identifiant existant
∧ badgeRemplce avec ce identifiant existant
— postcondition :
(Si le badgePerdu était en cours d'utilisation) vider le clef du badge
∧ (Si le badgePerdue était en cours d'utilisation) disassocie les relation entre le badgePerdu et la chambre, le badgePerdu et le client

∧ (Si le badgePerdu était en cours d'utilisation) chambre non occupée
∧ badgePerdu avec ce identifiant inexistant
∧ enregistrerOccupationChambre avec la même client, même chambre et badgeRemplce

Moyenne — Notifier les doublons dans les paires de clefs et les clefs

# 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.

### 2.1.1 Sprint 1

| Numéro de test | 1 | 2 | 3 |
|---|---|---|---|
| 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 | T |
| Nombre de jeux de test | 2 | 1 | 1 |

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

| Numéro de test | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Chambre existante | F | T | T | | | | T |
| Badge existant | | F | T | | | | T |
| Client existant | | | F | | | | T |
| Chambre non occupée | | | | F | T | T | T |
| Client occupe aucune chambre | | | | | F | T | T |
| Badge disponible | | | | | | F | T |
| Enregistrement accepté | F | F | F | F | F | F | T |
| Nombre de jeux de test | 1 | 1 | 1 | 1 | 1 | 1 | 3 |

TABLE 2 – Cas d'utilisation « enregistrer l'occupation d'une chambre par un client »

| Numéro de test | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Chambre existante | F | T | T | | | | T |
| Badge existant | | F | T | | | | T |
| Client existant | | | F | | | | T |
| Client occupe une chambre | | | | F | T | T | T |
| Client occupe la bonne chambre | | | | | F | T | T |
| Chambre occupeé | | | | | | F | T |
| Libération acceptée | F | F | F | F | F | F | T |
| Nombre de jeux de test | 1 | 1 | 1 | 1 | 1 | 1 | 3 |

TABLE 3 – Cas d'utilisation « libérer une chambre »

### 2.1.2 Sprint 2

| Numéro de test | 1 | 2 |
|---|---|---|
| Badge avec ce identifiant existant | F | T |
| Badge déclaré perdu acceptée | F | T |
| Nombre de jeux de test | 1 | 1 |

TABLE 4 – Cas d'utilisation « déclarer la perte d'un badge d'accès sans remplacement »

# 3 Conception

## 3.1 Liste des classes

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, sel
- Client — identifiant, nom, prénom (ces deux derniers sont ajoutés ici mais ne sont pas essentiels au fonctionnement du système GestionClefsHotel)
- Badge — identifiant
- Clef — value
- PaireClefs — clef1, clef2
- ClefVide — nullvalue
- PaireClefsVide — clef1, clef2
- Consomateur — id,nom,prenom
- Util (classe utilitaire déjà programmée) — attribut de classe TAILLE_CLEF, méthodes de classe genererUneNouvelleClef et clefToString)

### 3.2 Diagramme de classes

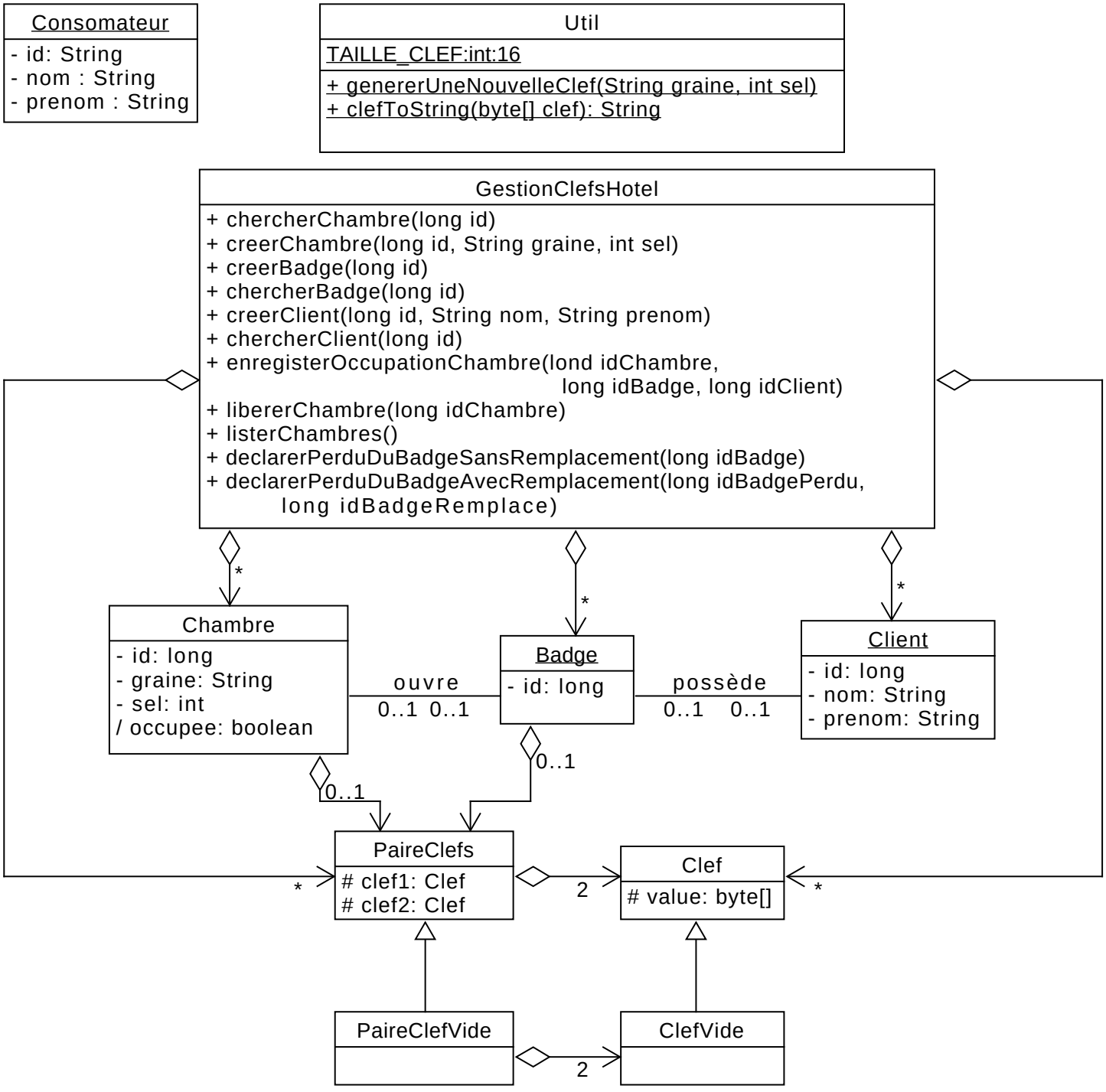


FIGURE 3 – Diagramme de classes

### 3.3 Diagrammes de séquence

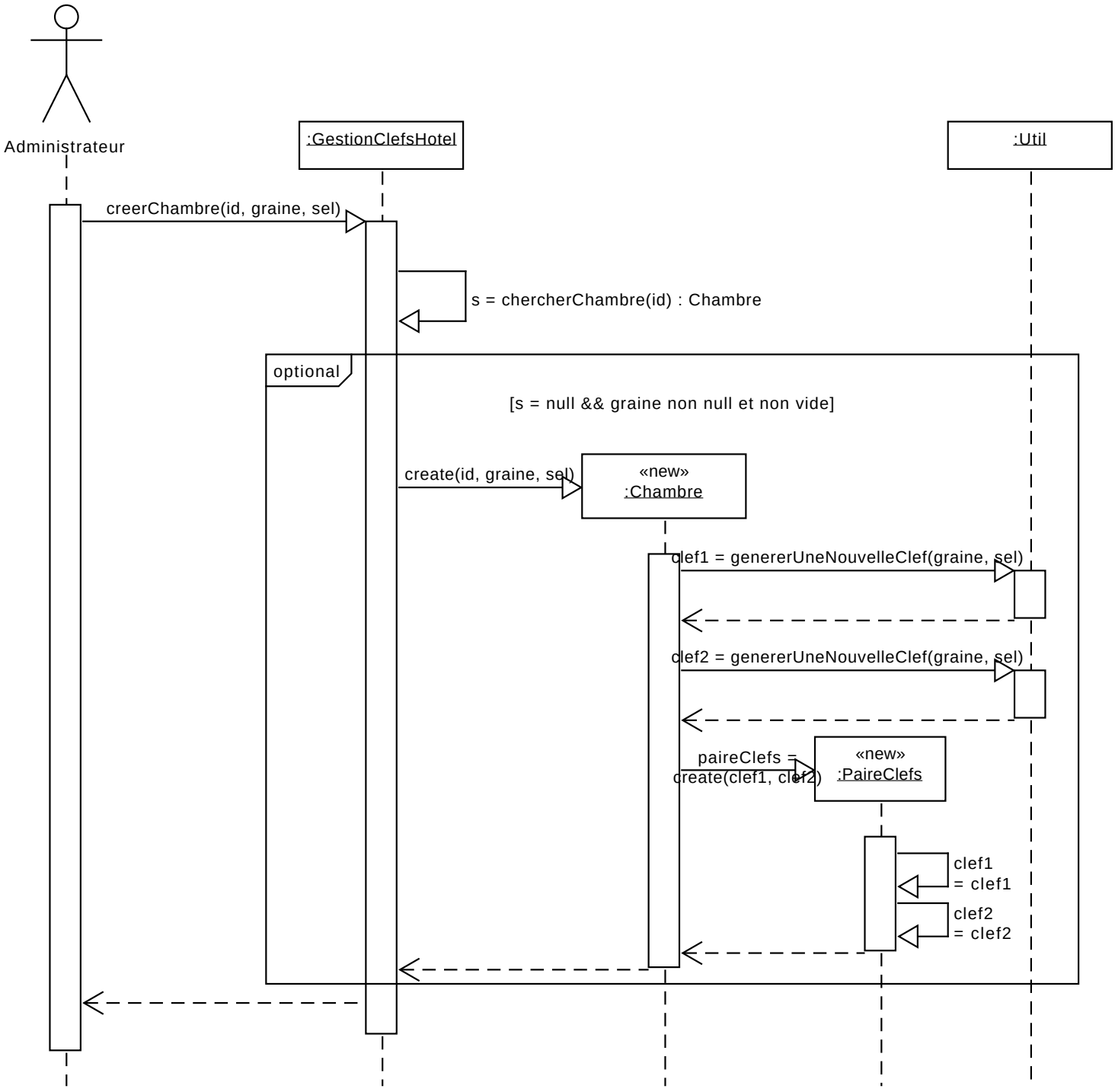


FIGURE 4 – Diagramme de séquence DSUC1 : «Créer une chambre»

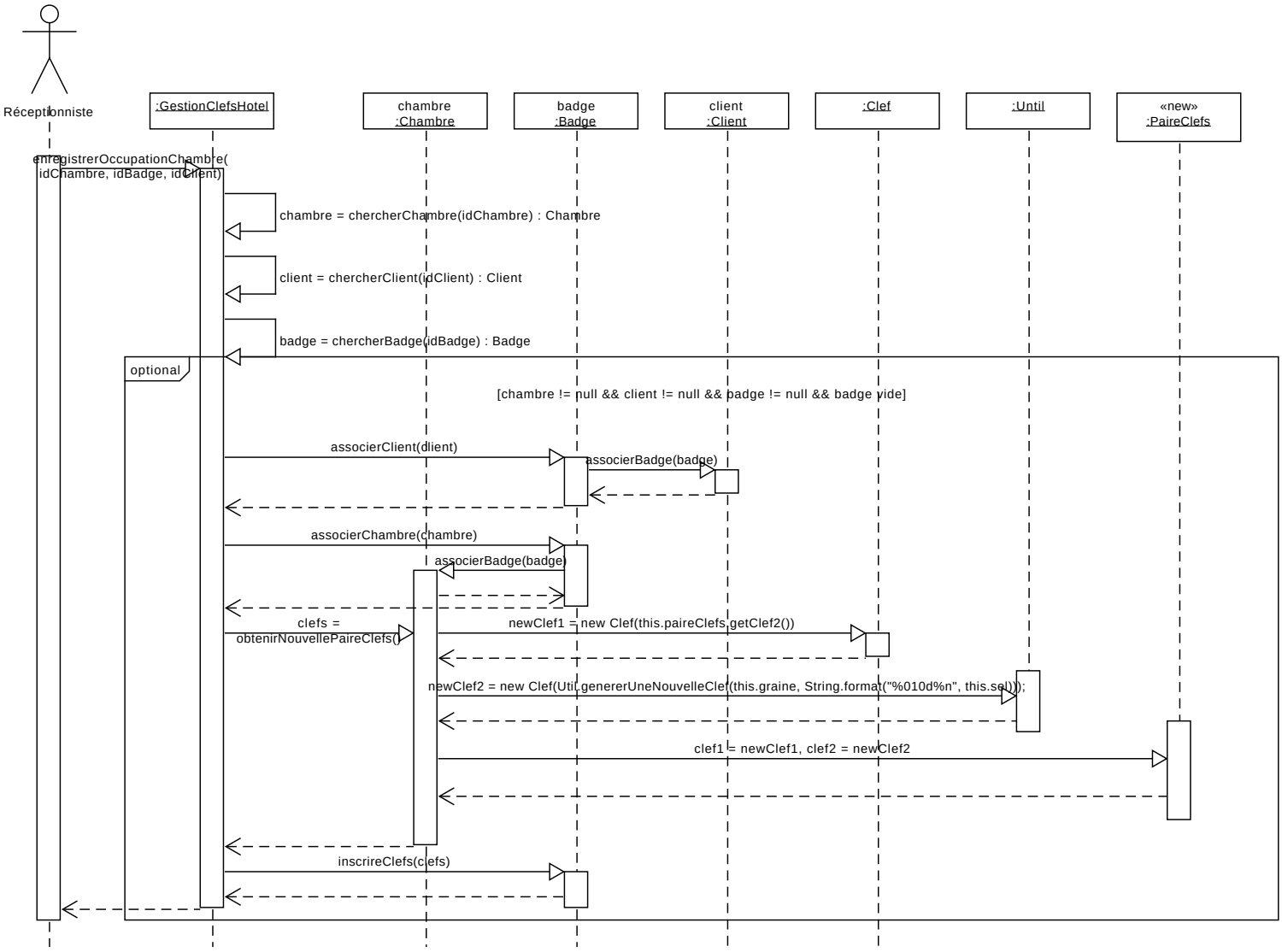


FIGURE 5 – Diagramme de séquence DSUC2 : «Enregistrer l'occupation d'une chambre par un client»

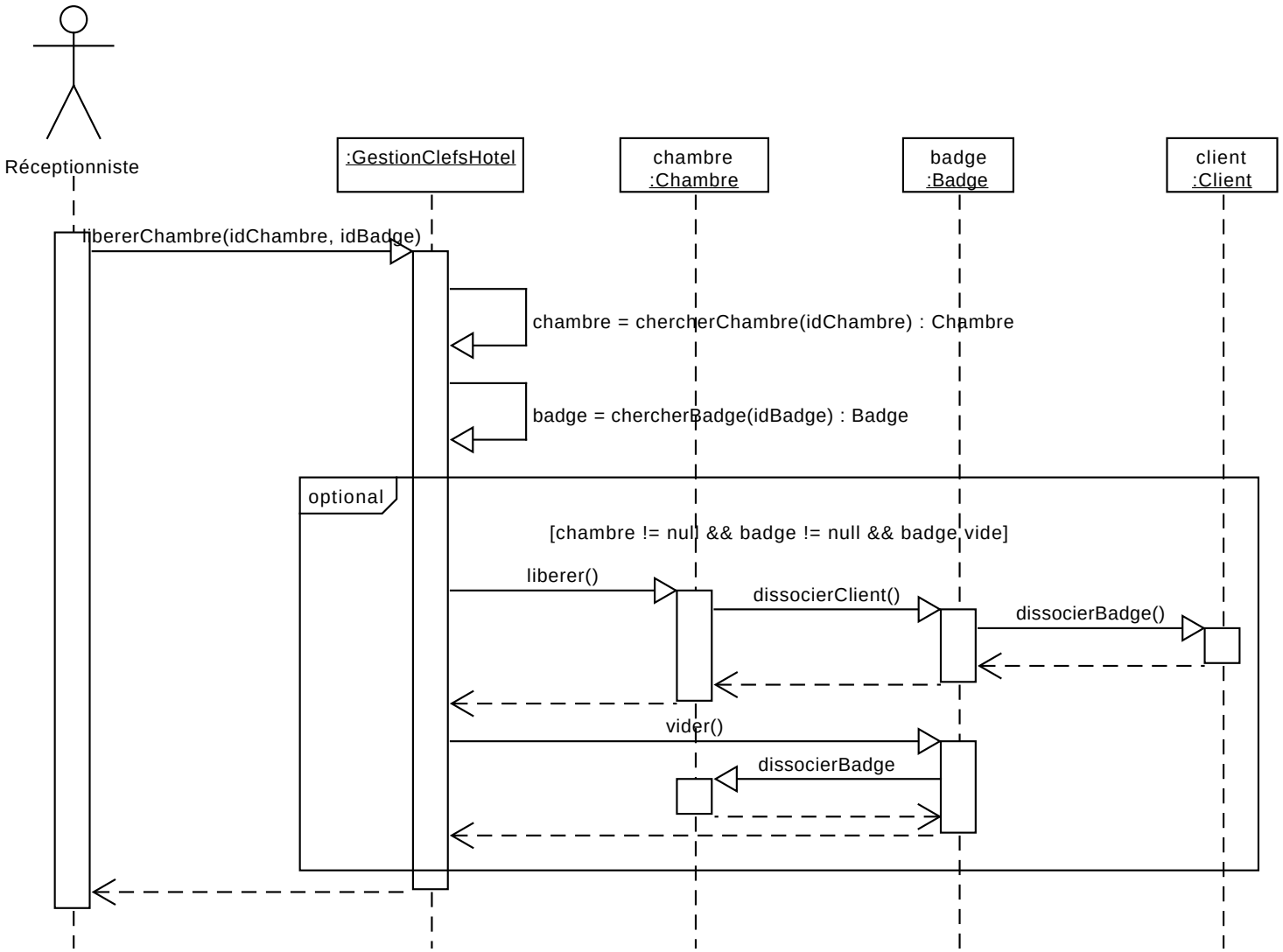


FIGURE 6 – Diagramme de séquence DSUC3 : «Libérer une chambre»

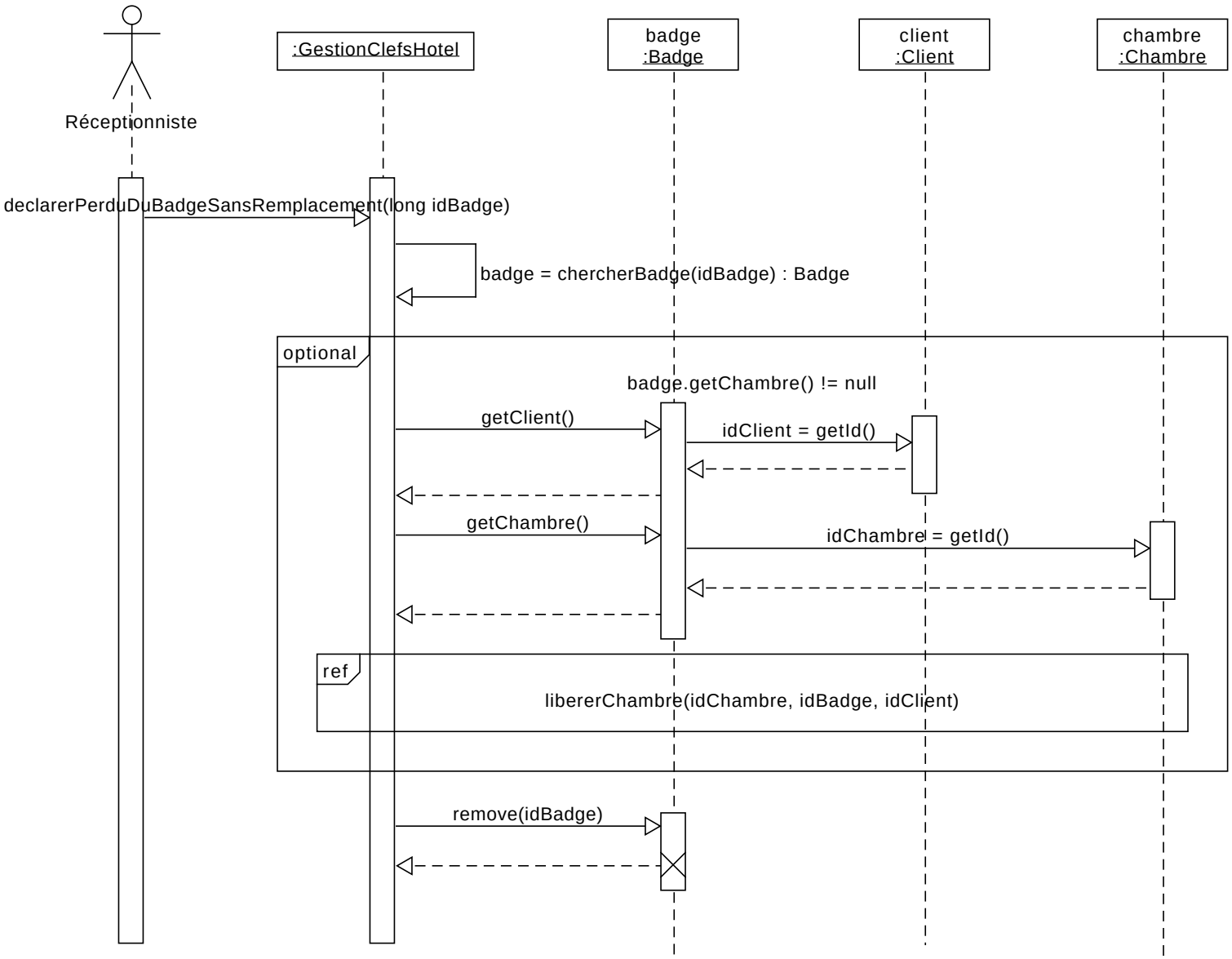


FIGURE 7 – Diagramme de séquence DSUC4 : «Déclarer le perte du badge sans remplacement»

# 4 Fiche des classes

## 4.1 Classe GestionClefsHotel

| GestionClefsHotel |
|---|
| <– **attributs « association »** –><br>– chambres : HashMap<Long, Chambre><br>– badges : HashMap<Long, Badge><br>– clients : HashMap<Long, Client><br>– pairesClefs : List<PaireClefs><br>– clefs : List<Clef><br>– publisher : SubmissionPublisher<String> |
| <– **constructeur** –><br>+ GestionClefsHotel()<br>+ boolean invariant()<br><– **operations « cas d'utilisation »** –><br>+ Chambre creerChambre(long id, String graine, int sel)<br>+ Badge creerBadge(long id)<br>+ Client creerClient(long id, String nom, String prenom)<br>+ void enregistrerOccupationChambre(long idChambre, long idBadge, long idClient)<br>+ void libererChambre(long idChambre, long idBadge, long idClient)<br>+ void declarerPerduDuBadgeSansRemplacement(long idBadge)<br>+ void declarerPerduDuBadgeAvecRemplacement(long idBadgePerdu,long idBadgeRemplace)<br>+ [Chambre] listerChambres()<br>+ void libererChambre(long id)<br><– **opérations de recherche** –><br>+ Badge chercherBadge(long id)<br>+ Chambre chercherChambre(long id)<br>+ Client chercherClient(long id) |

## 4.2 Classe Chambre

| Chambre |
|---|
| <– **attributs** –><br>– id : long<br>– graine : String<br>– sel : int<br>– occupee : boolean<br><– **attributs « association »** –><br>– paireClefs : PaireClefs<br>– badge : Badge |
| <– **constructeur** –><br>+ Chambre(long id, String graine, int sel)<br>+ boolean invariant() |

| |
|---|
| <– **operations** –><br>+long getId()<br>+Badge getBadge()<br>+void associerBadge(final Badge badge)<br>+void associerBadge(final Badge badge, final boolean bidirectionnel)<br>+void dissocierBadge()<br>+void dissocierBadge(final boolean bidirectionnel)<br>+boolean estOccupee()<br>+PaireClefs getClefs()<br>+PaireClefs obtenirNouvellePaireClefs()<br><– **operations« cas d'utilisation »** –><br>+ void inscrireClefs(PaireClefs paireClefs)<br>+ void liberer()<br>+ void enregistrerChambre() |

### 4.3 Classe Badge

| Badge |
|---|
| <– **attributs** –><br>– id : long<br><– **attributs « association »** –><br>– paireClefs : PaireClefs<br>– chambre : Chambre<br>– client : Client |
| <– **constructeur** –><br>+ Badge(long id)<br>+ boolean invariant()<br><– **operations « cas d'utilisation »** –><br>+ void inscrireClefs(PaireClefs paireClefs)<br>+ void vider()<br>+ void associerClient(Client client)<br>+ void associerChambre(Chambre chambre)<br>+ void dissocierClient() |

### 4.4 Classe Client

| Client |
|---|
| <– **attributs** –><br>– id : long<br>– nom : String<br>– prenom : String<br><– **attributs « association »** –><br>– badge : Badge |

| |
|---|
| <– **constructeur** –><br>+ Client(long id, String nom, String prenom)<br>+ boolean invariant() |

### 4.5 Classe PaireClefs

| PaireClefs |
|---|
| <– **attributs** –><br># clef1 : Clef<br># clef2 : Clef |
| <– **constructeurs** –><br>+ PaireClefs(Clef clef1, Clef clef2)<br>+ boolean invariant() |

## 5 Diagrammes de machine à états et invariants

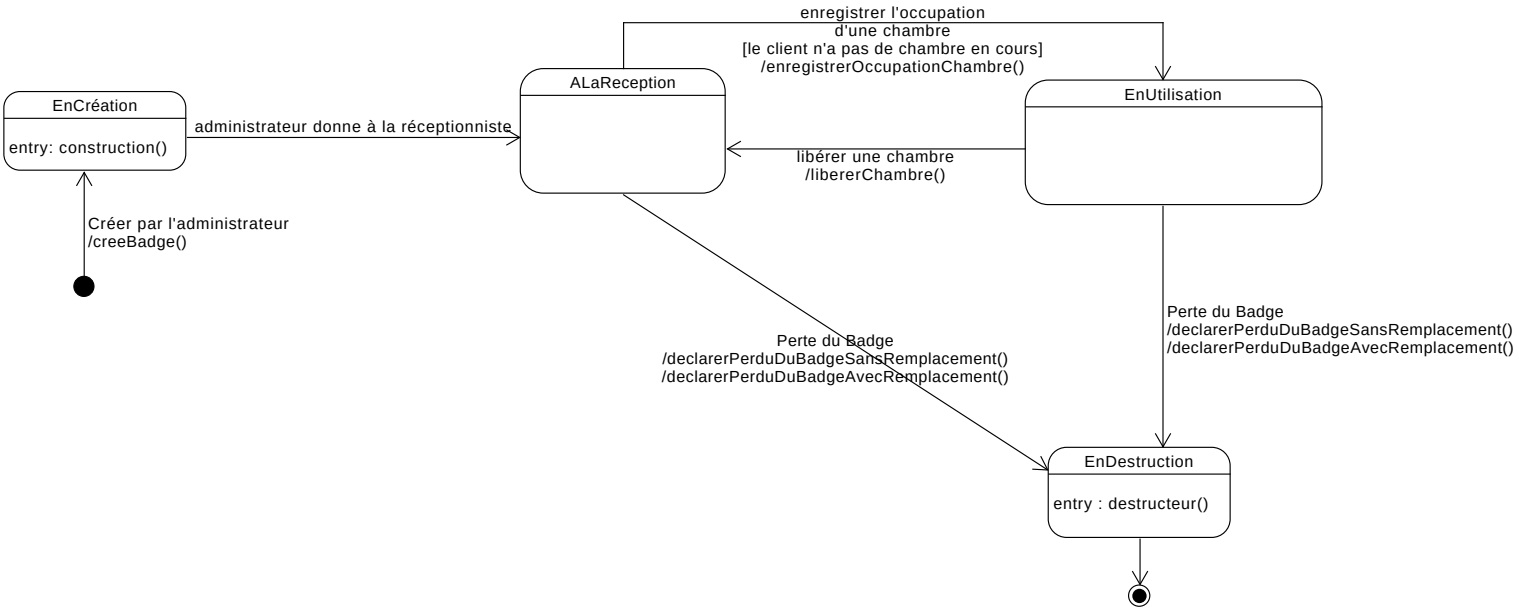


FIGURE 8 – DiagrammesDeMachineAEtats

Invariants :

( un client associe le badge
∧ un chambre associe le badge

∧ les paireClefs de badge ≠ null
∧ le badge du chambre est pareil que ce badge
∧ le badge du client est pareil que ce badge)
∨
( ∧ aucun client associe le badge
∧ aucun chambre associe le badge
∧ les paireClefs de badge = null
)

# 6 Préparation des tests unitaires

| | | |
|---|---|---|
| | Numéro de test | 1 |
| Postcondition | badge $\neq$ null | T |
| | paireClefs = null | T |
| | client = null | T |
| | chambre = null | T |
| Exception | Levée d'une exception | NON |
| Effet | Création du badge accepté | T |
| | Nombre de jeux de test | 1 |

Table 10 – Table de décision du "constructeur" de la classe "Badge"

| | | |
|---|---|---|
| | Numéro de test | 1 |
| Postcondition | client = client | T |
| | badge.clent = client.badge | T |
| Exception | Levée d'une exception | NON |
| Effet | Association Client Bidirectionnelle accepté pour le badge | T |
| | Nombre de jeux de test | 1 |

Table 11 – Table de décision du "AssociationClientBidirectionnelle" de la classe "Badge"