Projet CSC4102 : Gestion de bureaux en situation de bilocalisation

El Otmani Riad et Lahlali Saad
Année 2020–2021 — 04 janvier 2021

TABLE DES MATIERES

1 Spécification

1.1 Diagrammes de cas d’utilisation

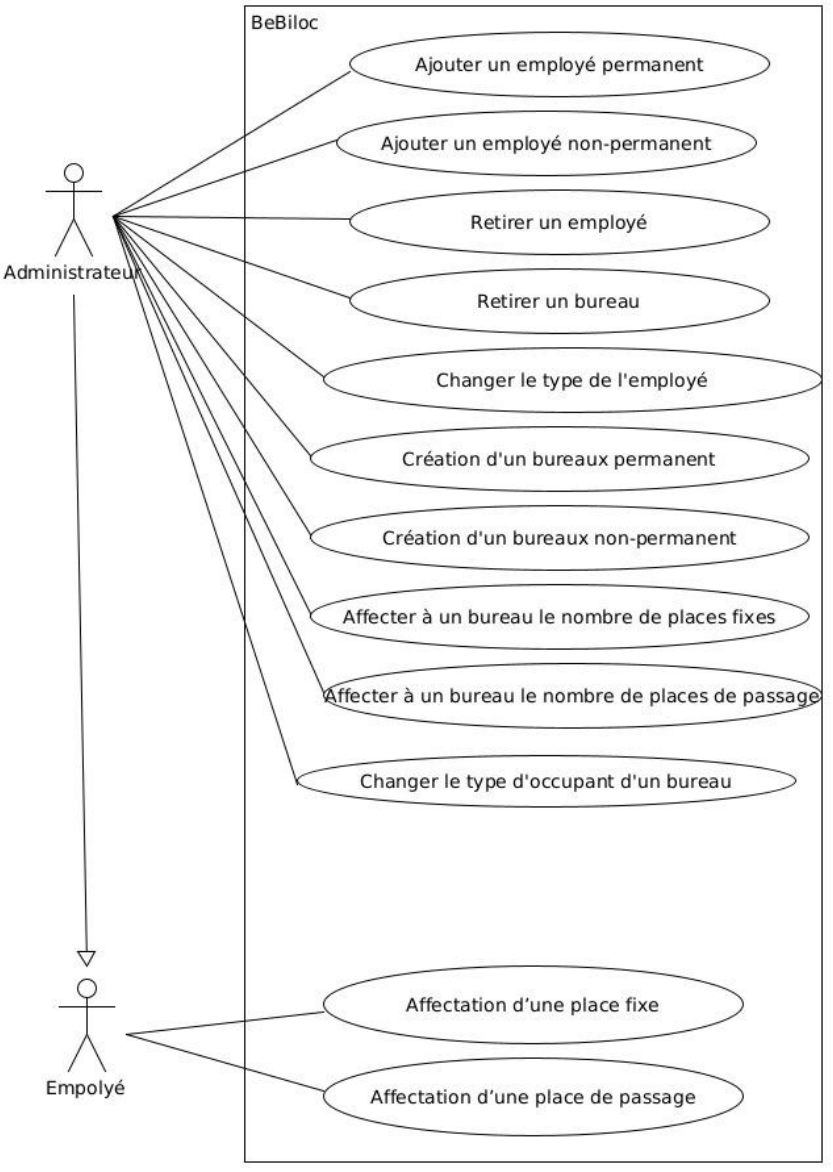


Figure 1: Diagramme de cas d’utilisation

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.

Nous avons préféré mettre deux cas d'utilisation pour ajouter un permanent ou un non-permanent, plutôt qu'un unique cas d'utilisation pour ajouter un employé. L'avantage des deux cas d'utilisations est d'éviter des données en entrée non utilisées dans certaines utilisation : un employé permanent ne possède pas de date de fin de contrat. Si l'on décide de mettre un seul cas d'utilisation, alors il faut vérifier (1) qu'un permanent n'a pas de date de fin de contrat, et (2) qu'un non-permanent possède une date de fin de contrat. C'est une préférence : cela n'impacte pas les autres cas d'utilisation.

Priorité HAUTE n°1 : Ajouter un employé permanent

- Précondition :

  ∧ identifiant de l'employé bien formé (non null et non vide)

  ∧ nom bien formé (non null et non vide)

  ∧ prénom bien formé (non null et non vide)

  ∧ date d'embauche non null

  ∧ fonction du permanent bien formée (non null et non vide)

  ∧ fonction du permanent ∈{ direction département, direction adjointe département, assistance gestion, enseignement recherche }

  ∧ employé avec cet identifiant inexistant
- Postcondition :

  employé permanent avec cet identifiant existant

Priorité HAUTE n°2 : Ajouter un employé non-permanent

- Précondition :

  ∧ identifiant de l'employé bien formé (non null et non vide)

  ∧ nom bien formé (non null et non vide)

  ∧ prénom bien formé (non null et non vide)

  ∧ date d'embauche non null

  ∧ date de fin de contrat non null

  ∧ date de fin de contrat ≥ date d'embauche

∧ fonction du non-permanent bien formée (non null et non vide)

∧ fonction du non-permanent ∈ { doctorat, post-doctorat, ingénierie recherche, stage }

∧ employé avec cet identifiant inexistant

– Postcondition :

∧ employé permanent avec cet identifiant existant

Priorité HAUTE n°3 : Création d'un bureau permanent

– Précondition :

∧ nombre de places fixes indiqués (non null et non vide)

∧ nombre de places de passages indiqués (non null et non vide)

∧ nombre de places total du bureau pour permanent ∈ { 1, 2 }

∧ bureau permanent contient une seul place fixe

∧ bureau avec cet identifiant inexistant

– Postcondition :

∧ bureau pour permanent avec cet identifiant existant

Priorité HAUTE n°4 : Création d'un bureau non-permanent

– Précondition :

∧ nombre de places fixes indiqués (non null et non vide)

∧ nombre de places de passages indiqués (non null et non vide)

∧ nombre de places total du bureau pour non permanent ∈ { 1,2,3,4,5,6}

∧ bureau avec cet identifiant inexistant

– Postcondition :

∧ bureau pour non permanent avec cet identifiant existant

Priorité Moyenne n°1 : Retirer un employé

– Précondition :

∧ employé avec cet identifiant existant

– Postcondition :

∧ identifiant inexistant

Priorité Moyenne n°2 : Retirer un bureau

– Précondition :

∧ bureau avec cet identifiant existant

– Postcondition :

∧ identifiant inexistant

Priorité Moyenne n°3 : Affectation d'une place fixe

– Précondition :

∧ employé avec cet identifiant existant

∧ bureau avec cet identifiant existant

∧ nombre de places fixes libre ≥ 1

∧ si non manager, le nombre de place fixe de l'employé vérifié (null)

∧ si manager, le nombre de place fixe sur le site actuel vérifié (null)

– Postcondition :

∧ la place fixe est attribué à l'employé

Priorité Moyenne n°4 : Affectation d'une place de passage

- Précondition :
  - ∧ employé avec cet identifiant existant
  - ∧ bureau avec cet identifiant existant
  - ∧ nombre de places de passage libre ≥ 1
  - ∧ nombre de place fixe de l'employé sur le site actuel vérifié (null)
  - ∧ période d'attribution indiquée (non null et non vide)
  - ∧ période d'occupation ≥ date d'affectation
- Postcondition :
  - ∧ la place fixe est attribué à l'employé

Priorité faible n°1 : Changer le type de l'employé

- Précondition :
  - ∧ identifiant de l'employé existant
  - ∧ type en entrée ∈ {permanent, non permanent}
- Postcondition :
  - ∧ le type de l'employé a changé

Priorité faible n°2 : Affecter à un bureau le nombre de places fixes

- Précondition :
  - ∧ identifiant du bureau existant
  - ∧ si le bureau est pour non permanent, nombre de places ∈ { 1, 2 ,3,4,5,6}
  - ∧ si le bureau est pour permanent, nombre de places ∈ { 1}
- Postcondition :
  - ∧ le nombre de places fixes du bureau a changé

Priorité faible n°3 : Affecter à un bureau le nombre de places de passage

- Précondition :
  - ∧ identifiant du bureau existant
  - ∧ si le bureau est pour non permanent, nombre de places ∈ { 1, 2 ,3,4,5,6}
  - ∧ si le bureau est pour permanent, nombre de places ∈ { 1,2}
- Postcondition :
  - ∧ le nombrede places de passage du bureau a changé

Priorité faible n°4 : Changer le type d'occupant d'un bureau

- Précondition :
  - ∧ identifiant du bureau existant
  - ∧ si le bureau est pour non permanent, nombre de places ∈ { 1, 2 ,3,4,5,6}
  - ∧ si le bureau est pour permanent, nombre de places ∈ { 1,2}
  - ∧ type d'occupant en entrée ∈ {permanent, non permanent}
- Postcondition :
  - ∧ le type d'occupant a changé

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 ceux de priorité HAUTE.

Tableau 1: Cas d'utilisation « ajouter un employé permanent ». Le test 6 possède 5 jeux de test pour les 4 fonctions de non-permanents et pour une fonction inconnue. Le test 8 possède 4 jeux de test pour les 4 fonctions de permanents.

Numéro de test	1	2	3	4	5	6	7	8
Identifiant de l'employé bien formé (non nullnon vide)	F	T	T	T	T	T	T	T
Nom bien formé (non nullnon vide)		F	T	T	T	T	T	T
Prénom bien formé (non nullnon vide)			F	T	T	T	T	T
Date d'embauche ≠ null				F	T	T	T	T
Fonction du permanent bien formé (non nullnon vide)					F	T	T	T
fonction du permanent ∈{ direction département, direction adjointe département, assistance gestion, enseignement recherche }						F	T	T
Employé avec cet identifiant non existant							F	T
Création acceptée	F	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	2	5	1	4

Tableau 2: Cas d'utilisation « ajouter un employé non-permanent ». Le test 8 possède 5 jeux de test pour les 4 fonctions de permanents et pour une fonction inconnue. Le test 10 possède 4 jeux de test pour les 4 fonctions de non-permanents

Numéro de test	1	2	3	4	5	6	7	8	9	10
Identifiant de l'employé bien formé (non nullnon vide)	F	T	T	T	T	T	T	T	T	T
Nom bien formé (non nullnon vide)		F	T	T	T	T	T	T	T	T
Prénom bien formé (non nullnon vide)			F	T	T	T	T	T	T	T
Date d'embauche ≠ null				F	T	T	T	T	T	T
Date de fin de contrat ≠ null					F	T	T	T	T	T
Date de fin de contrat ≥ date d'embauche						F	T	T	T	T
Fonction du non-permanent bien formé (non nullnon vide)							F	T	T	T
Fonction du non-permanent ∈{ doctorat, post-doctorat, ingénierie recherche, stage }								F	T	T
Employé avec cet identifiant non existant									F	T
Création acceptée	F	F	F	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	1	1	2	5	1	4

Tableau 3: Cas d’utilisation « création d’un bureau permanent ».

Numéro de test	1	2	3	4	5
Identifiant du bureau non null et non vide	F	T	T	T	T
Localisation du bureau non null et non vide		F	T	T	T
Nombre de places fixes ∈ { 1 }			F	T	T
Nombre de places passages ∈ { 0, 1 }				F	T
Bureau avec cet identifiant inexistant					F
Création acceptée	F	F	F	F	F
Nombre de jeux de test	2	2	2	3	2

Tableau 4: Cas d’utilisation « création d 'un bureau pour non-permanent ».

Numéro de test	1	2	3	4	5
Nombre de places indiquées (non null et non vide)	F	T	T	T	T
Localisation du bureau non null et non vide		F	T	T	T
Nombre de places total du bureau pour non permanent <= 6			F	T	T
Bureau avec cet identifiant inexistant				F	T
Création acceptée	F	F	F	F	T
Nombre de jeux de test	2	2	5	1	2

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 :

- BeBiloc (la façade),
- Employé — identifiant, nom, prénom, dateEmbauche, dateFinContrat, fonction,
- Fonction (énumération) — nom, typeFonction, avec 8 énumérateurs,
- TypeFonction (énumération) — 2 énumérateurs
- Bureau — identifiant, localisation, nombre de places fixes totale, nombre de places de passage totale, nombre de places fixes utilisées, nombre de places de passage utilisées, isPermanent, liste des identifiants des employés dans sles places fixes, liste des identifiants des employés dans sles places de passage

Une variante serait une généralisation spécialisation au lieu d’une énumération autour de la classe Employé : Employé en classe parente, et DIRECTION_DÉPARTEMENT, etc. en classes enfants. Nous avons préféré une classe avec une énumération car un employé peut changer de fonction (ce cas d’utilisation est plus fréquent qu’on ne le croit) et beaucoup de traitements dépendent du type de l’employé (l’utilisation d’une énumération est quelque peu plus aisée).

3.2 Diagramme de classes

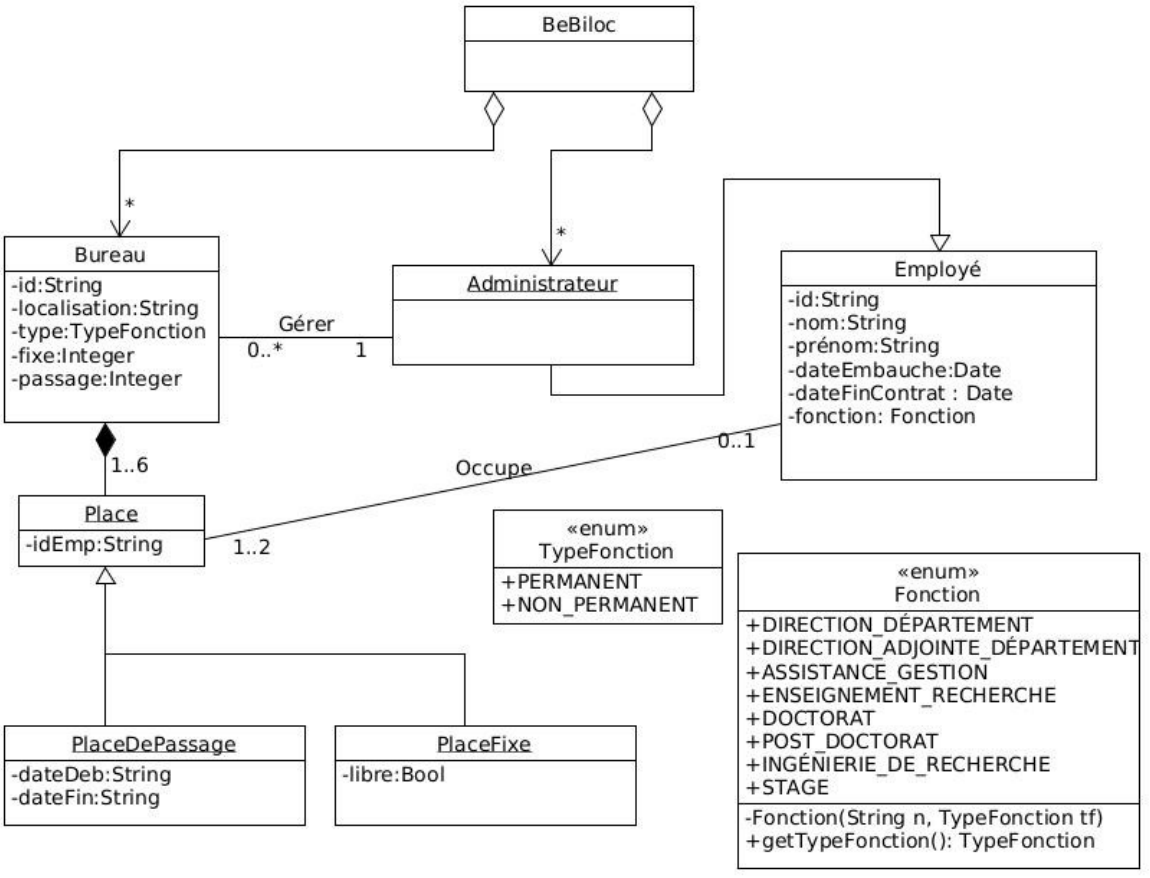


Figure 2: Diagramme de classes

3.3 Diagrammes de séquence

Voici la description textuelle du cas d'utilisation « ajouter un employé non-permanent » :

- arguments en entrée : identifiant de l'employé, nom et prénom de l'employé, date d'embauche, date de fin de contrat, fonction ;
- rappel de la précondition : identifiant de l'employé bien formé (non nullnon vide) ∧ nom bien formé (non null et non vide) ∧ prénom bien formé (non nullnon vide) ∧ date d'embauche non null∧ date de fin de contrat non null∧ date de fin de contrat ≥ date d'embauche ∧ fonction du non-permanent non null∧ fonction du permanent ∈{doctorat, post-doctorat, ingénierie recherche, stage} ∧ employé avec cet identifiant inexistant
- algorithme :
  1. vérifier les arguments
  2. chercher un employé avec cet identifiant
  3. vérifier que l'employé est inexistant
  4. instancier l'employé
  5. ajouter l'employé dans la collection des employés

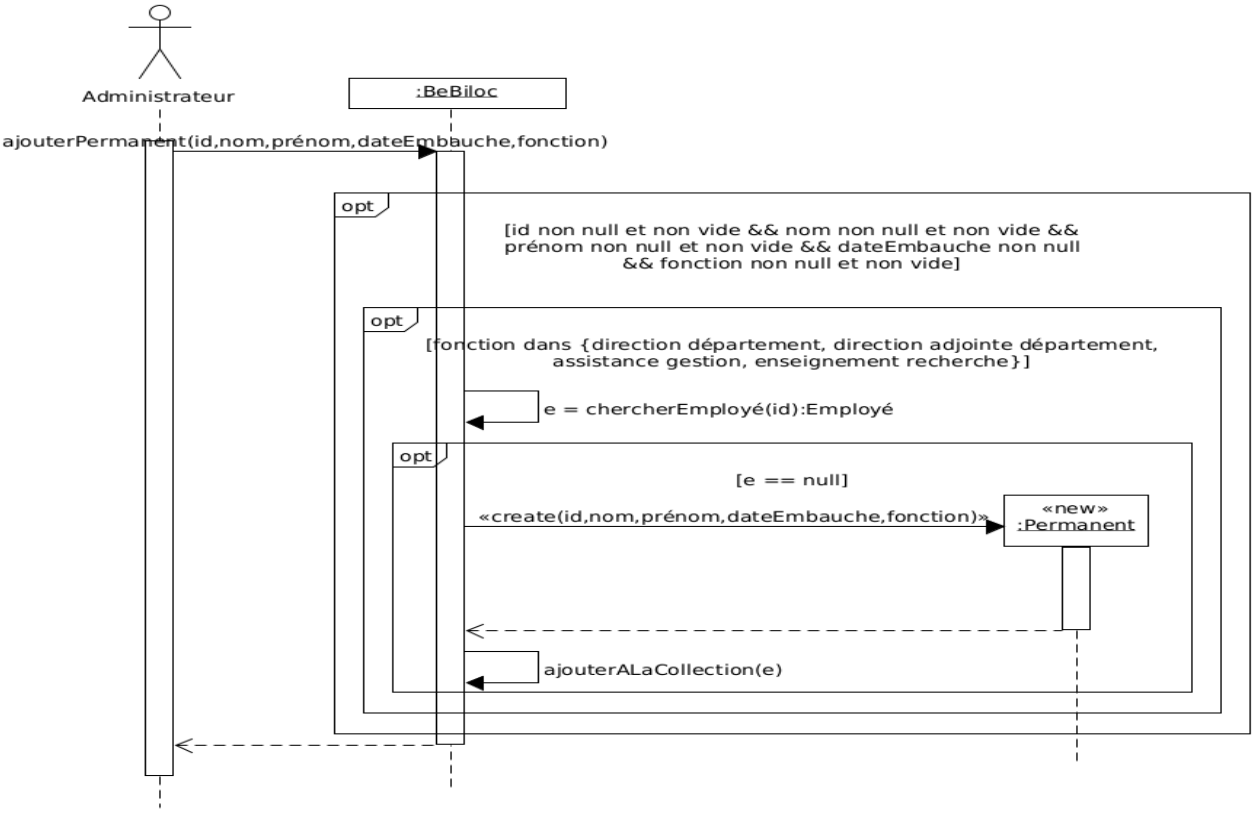


Figure 3: Diagramme de séquence du cas d'utilisation « ajouter un employé permanent »

Voici la description textuelle du cas d'utilisation « ajouter un employé non-permanent » :

- arguments en entrée : identifiant de l'employé, nom et prénom de l'employé, date d'embauche, date de fin de contrat, fonction ;
- rappel de la précondition : identifiant de l'employé bien formé (non nullnon vide) ∧ nom bien formé (non null et non vide) ∧ prénom bien formé (non nullnon vide) ∧ date d'embauche non null∧ date de fin de contrat non null∧ date de fin de contrat ≥ date d'embauche ∧ fonction du non-permanent non null∧ fonction du permanent ∈{doctorat, post-doctorat, ingénierie recherche, stage} ∧ employé avec cet identifiant inexistant
- algorithme :
    1. vérifier les arguments
    2. chercher un employé avec cet identifiant
    3. vérifier que l'employé est inexistant
    4. instancier l'employé
    5. ajouter l'employé dans la collection des employés

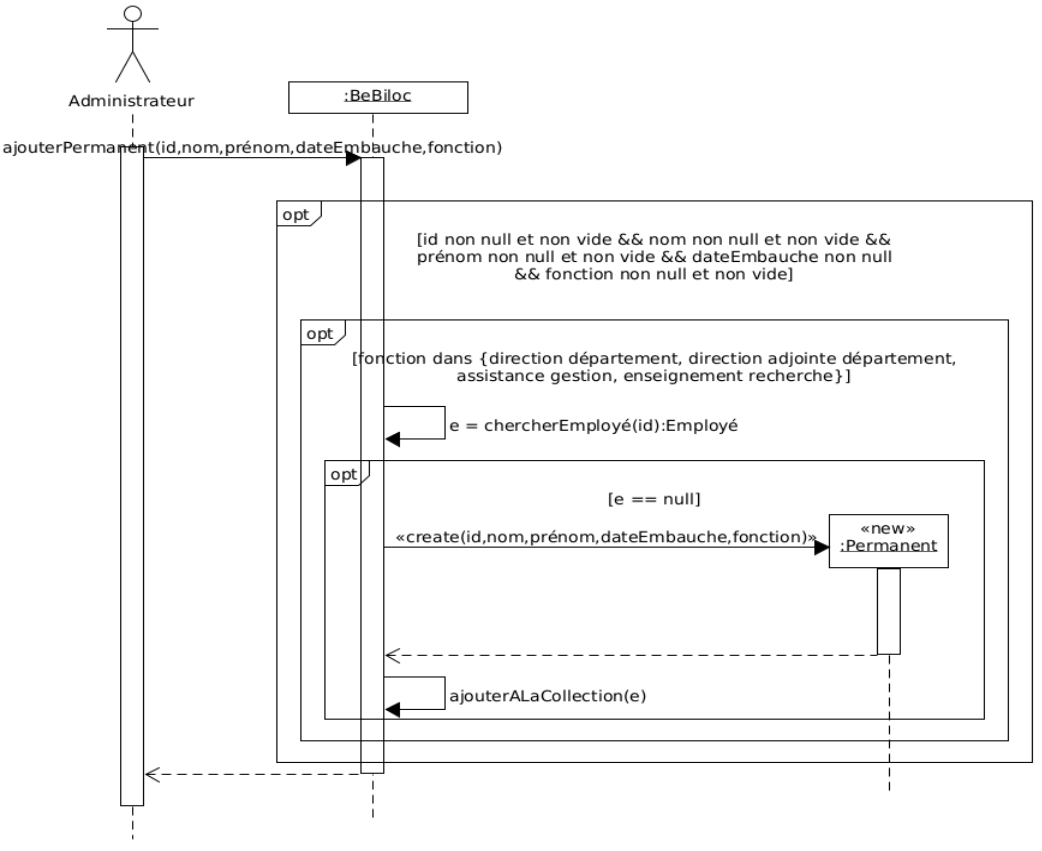


Figure 4: Diagramme de séquence du cas d'utilisation « ajouter un bureau pour permanents »

Voici la description textuelle du cas d'utilisation « ajouter un bureau pour permanents » :

- arguments en entrée : identifiant du bureau, localisation, nbFixe + nbPassage du bureau ;
- rappel de la précondition : identifiant du bureau bien formé (non nullnon vide) ∧ site de localisation non null ∧ nbFixe + nbPassage ∈[1..2] ∧ bureau avec cet identifiant inexistant ;
- algorithme :
    1. vérifier les arguments
    2. chercher un bureau avec cet identifiant
    3. vérifier que le bureau n'existe pas
    4. instancier le bureau
        - création d'une place fixe
        - si nbFixe + nbPassage =2 alors création d'une place de passage
    5. ajouter le bureau dans la collection des bureaux

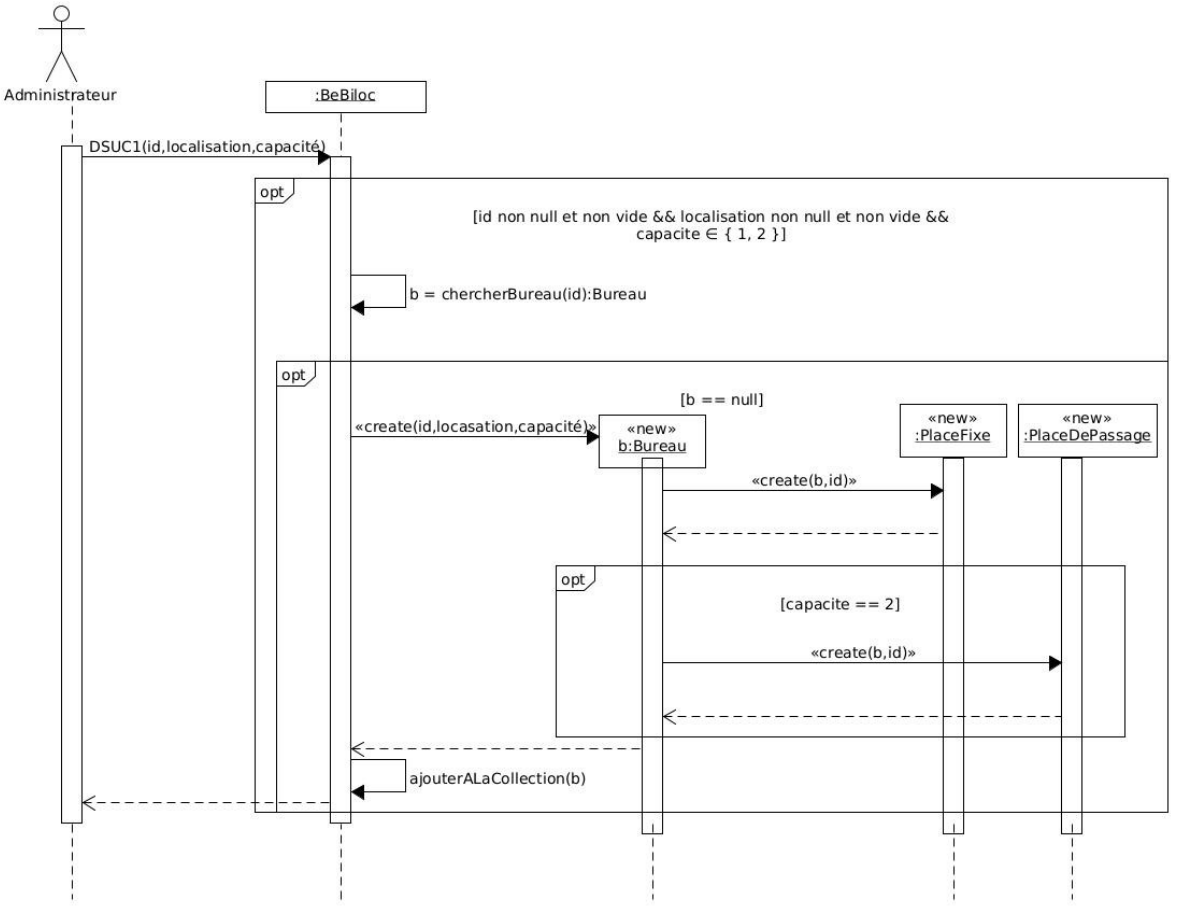


Figure 5 : ajouter un bureau pour permanents

Voici la description textuelle du cas d'utilisation « ajouter un bureau pour non-permanents » :

- arguments en entrée : identifiant du bureau, localisation, nbFixes; nbPassage
- rappel de la précondition : identifiant du bureau bien formé (non null & non vide) ∧ site de localisation non null ∧ (nbFixes+nbPassage)∈[1..6] ∧ bureau avec cet identifiant inexistant ;
- algorithme :
    1. vérifier les arguments
    2. chercher un bureau avec cet identifiant
    3. vérifier que le bureau n'existe pas
    4. instancier le bureau
        - création de nbFixes places fixes
        - création de nbPassage places de passage
    5. ajouter le bureau dans la collection des bureaux

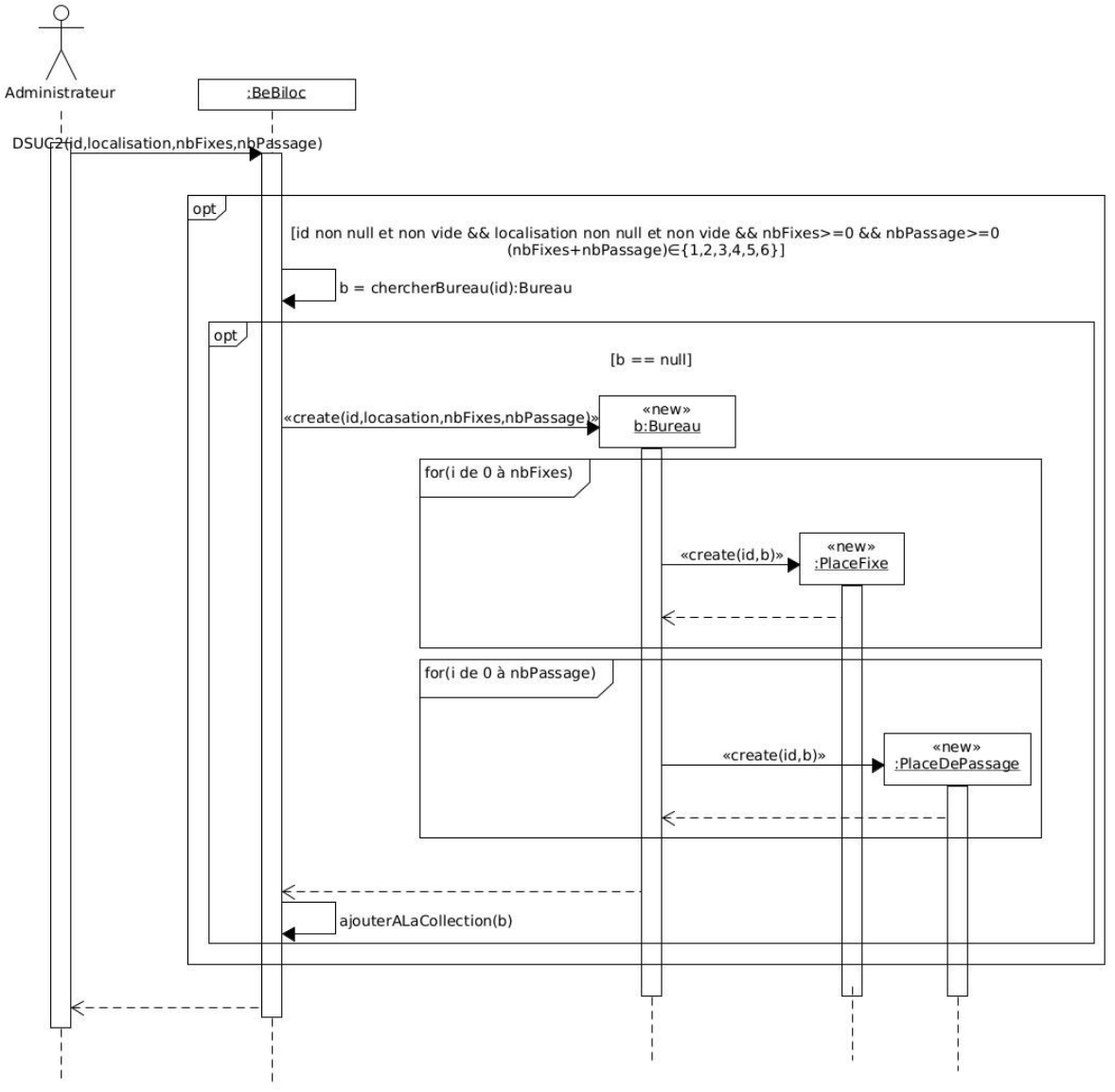
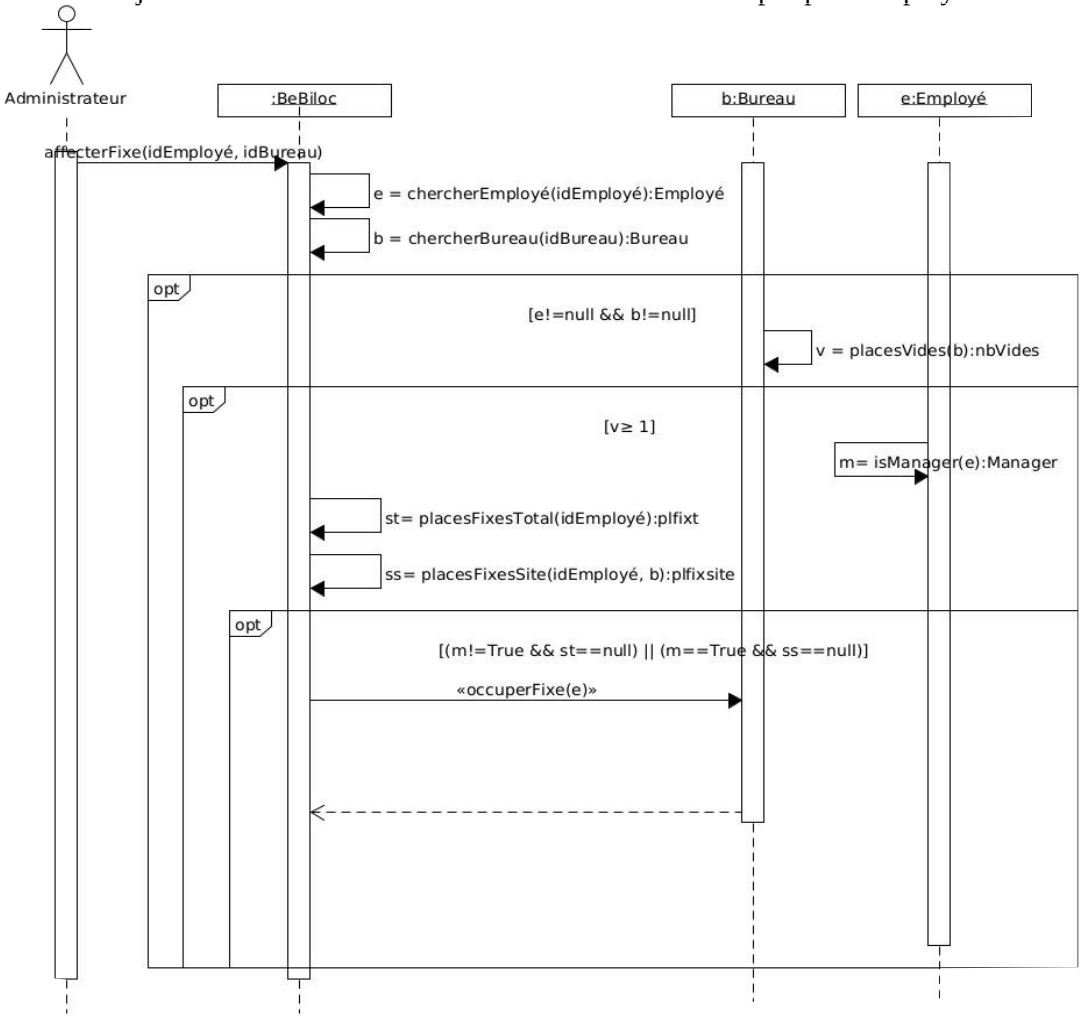


Figure 6 : ajouter un bureau pour non-permanents

Voici la description textuelle du cas d'utilisation « affecter un employé à une place fixe » :

- arguments en entrée : identifiantEmployé , identifiantBureau,
- rappel de la précondition : employé avec cet identifiant existant ∧ bureau avec cet identifiant existant ∧ nombre de places fixes libre ≥ 1 ∧ si non manager, le nombre de place fixe de l'employé sur les deux campus (null) ∧ si manager, le nombre de place fixe sur le site du bureau (null) ;
- algorithme :
  1. chercher un employé avec cet identifiant
  2. chercher un bureau avec cet identifiant
  3. verifier que l'identifiant de l'employé et du bureau existent
  3. chercher le nombre de places vides dans le bureau
  4. vérifie que le nombre de places vides est supérieur à 1
  5. identifier le status de l'employé
  6. chercher le nombre de places fixes totales occupées par l'employé
  7. chercher le nombre de places fixes occupées par l'employé dans le site du bureau
  8. vérifier que soit l'employé est non manager et le nombre de place fixe de l'employé sur les deux campus est null soit l'employé est un manager et le nombre de place fixe sur le site du bureau est null
  9. augmenter de 1 le nombre de places fixes du bureau
  10. Ajouter l'identifiant Bureau à la liste des bureaux occupés par l'employé



4 Fiche des classes

4.1 Classe BeBiloc

BeBiloc
<– attributs « association » –> − bureau : collection de Bureau − employes : collection de Employe
<– constructeur –> + BeBiloc() + invariant() : booléen
<– operations « cas d'utilisation » –> + ajouterPermanent(String id, String nom, String prenom, LocalDate dateEmbauche, String fonction) + ajouterNonPermanent(String id, String nom, String prenom, LocalDate dateEmbauche, LocalDate dateFinContrat, String fonction) + listerEmployes() : collection de String + ajouterBureauPermanents(String id, Localisation localisation, int nbFixe, int nbPassage) + ajouterBureauNonPermanents(String id, Localisation localisation, int nbFixe, int nbPassage) +affecterPlaceFixe(String idEmploye, String idBureau) + affecterPlaceDePassage(String idEmploye, String idBureau, String dateDeb, String dateFin)

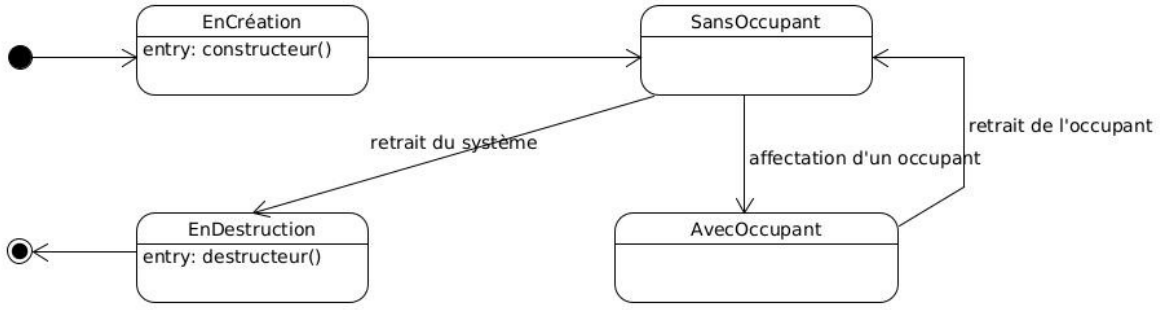
4.2 Classe Employé

Employé
− identifiant : String − nom : String − prénom : String − dateEmbauche : Date − dateFinContrat : Date - fonction : Fonction
<– attributs « association » –> − fonction : TypeFonction − placesOccupées : collection de places occupées
<– constructeur –> + Employé(String id, String nom, String prénom, LocalDate dateEmbauche, Fonction fonction) + Employé(String id, String nom, String prénom, LocalDate dateEmbauche, LocalDate dateFinContratFonction fonction) + invariant() : booléen +Destructeur()

4.3 Classe PlaceFixe

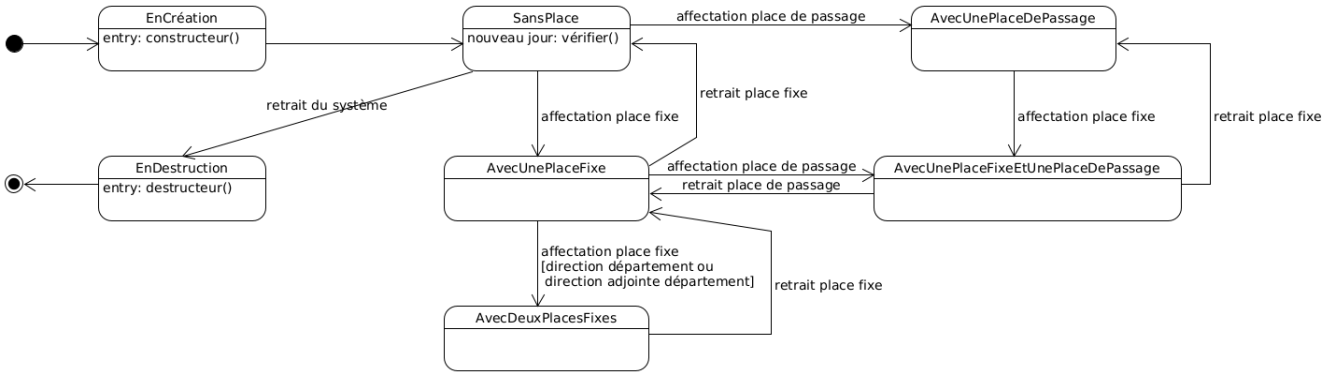
PlaceFixe
-idEmp:String -libre:Bool

<– constructeur –>
+ PlaceFixe(bureau Bureau, String id)
+ Affectation(String idEmp)
+ Desaffectation()
+ Destructeur()
+ hashCode() //override
+ equals() //override



5 Diagrammes de machine à états et invariants

5.1 Classe Employé



Pour garder l'invariant quelque peu local à la classe, c'est-à-dire sans trop d'appels sur d'autres objets métier, les conditions suivantes ne sont pas intégrées à l'invariant :

- le nombre de places de passage est 1,
- le nombre de places sur un même site est 1.

L'invariant de la classe Employé est le suivant :

∧ *identifiant≠null* ∧ *identifiant≠vide*

∧ *nom≠null* ∧ *nom≠vide*

∧ *prenom≠null* ∧ *prenom≠vide*

∧ *dateEmbauche≠null* ∧ *fonction≠null*

∧ *fonction=permanent => dateFinContrat=null*

∧ *fonction≠permanentdate => FinContrat≠null*

5.2 Classe PlaceFixe

L'invariant de la classe PlaceFixe est le suivant :

(idPlace ≠null ∧ *idPlace ≠vide)*

5.3 Classe Bureau

L’invariant de la classe Bureau est le suivant :

∧ *identifiant≠null* ∧ *identifiant≠vide*

∧ *localisation=Evry* || *localisation=Palaiseau*

∧ *fixe≠null* ∧ *fixe≠vide*

∧ *passage≠null* ∧ *passage≠vide*

∧ *type=PERMANENT* || *type=NON_PERMANENT*

6 Préparation des tests unitaires

6.1 Classes Employé

Tableau 3: Méthode constructeurEmployé de la classe Employé pour un employé permanent. Nous avons mis 4 tests pour les jeux de test 6 et 7 : 1 par fonction non autorisée et 1 par fonction autorisée.

Numéro de test	1	2	3	4	5	6	7
identifiant ≠*null*∧≠*vide*	F	T	T	T	T	T	T
nom ≠*null*∧≠*vide*		F	T	T	T	T	T
prénom ≠*null*∧≠*vide*			F	T	T	T	T
dateEmbauche ≠*null*				F	T	T	T
fonction ≠*null*					F	T	T
fonction ∈{direction département, direction adjointe département, assistance gestion, enseignement recherche}						F	T
identifiant'=*identifiant*							T
nom'=*nom*							T
prenom'=*prenom*							T
dateEmbauche'=*dateEmbauche*							T
dateFinContrat'=*null*							T
fonction'=*fonction*							T
invariant							T
Levée d'une exception	OUI	OUI	OUI	OUI	OUI	OUI	NON
Objet créé	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	1	4	4

Tableau 4: Méthode constructeurEmployé de la classe Employé pour un employé non-permanent. Nous avons mis 2 tests pour le jeu de test 6 : 1 pour l'égalité des dates et 1 lorsqu'elles ne sont pas égales. Nous avons mis 4 tests pour les jeux de test 7 et 8 : 1 par fonction non autorisée et 1 par fonction autorisée.

Numéro de test	1	2	3	4	5	6	7	8
identifiant ≠*null*∧≠*vide*	F	T	T	T	T	T	T	T
nom ≠*null*∧≠*vide*		F	T	T	T	T	T	T
prénom ≠*null*∧≠*vide*			F	T	T	T	T	T
dateEmbauche ≠*null*				F	T	T	T	T
dateFinContrat ≠*null*					F	T	T	T
dateFinContrat ≥ dateEmbauche					F	T	T	T
fonction ≠*null*						F	T	T
fonction ∈{doctorat, post-doctorat, ingénierie recherche, stage}							F	T
identifiant'=*identifiant*								T
nom'=*nom*								T
prenom'=*prenom*								T
dateEmbauche'=*dateEmbauche*								T
dateFinContrat'=*dateFinContrat*								T
fonction'=*fonction*								T
invariant								T
Levée d'une exception	OUI	OUI	OUI	OUI	OUI	OUI	OUI	NON
Objet créé	F	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	1	2	4	4

Tableau 5 : Méthode constructeurBureau de la classe Bureau pour un bureau permanent

Numero de test	1	2	3	4
identifiant ≠*null*∧identifiant≠*vide*	F	T	T	T
localisation ∈*{Evry, Palaiseau}*		F	T	T
NbFixe + nbPassage ∈ [1..2]			F	T
id’=id				T
localisation’=localisation				T
nbFixe + nbPassage’=nbFixe + nbPassage				T
type’=type				T
invariant				T
Levée d’une exception	OUI	OUI	OUI	NON
Objet crée	F	F	F	T
Nombre de jeux de test	2	1	2	2

Tableau 6 : Méthode constructeurBureau de la classe Bureau pour un bureau non permanent

Numero de test	1	2	3	4	5	6	7
identifiant ≠*null*∧identifiant≠*vide*	F	T	T	T	T	T	T
localisation ∈*{Evry, Palaiseau}*		F	T	T	T	T	T
nbfixe>=0			F	T	T	T	T
nbpassage>=0				F	T	T	T
type≠*null*∧type≠*vide*					F	T	T
(nbfixe+nbpassage)<=6						F	T
id’=id							T
localisation’=localisation							T
nbfixe’=nbfixe							T
nbpassage’=nbpassage							T
type’=type							T
invariant							T
Levée d’une exception	OUI	OUI	OUI	OUI	OUI	OUI	NON
Objet crée	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	1	1	1	1	2

Tableau 6 : AssocierPlaceFixeLibreAEmp

Numero de test	1	2	3	4	5	6	7
idEmploye ∈ collection(Employes)	F	T	T	T	T	T	T
idBureau ∈ collection(Bureaux)		F	T	T	T	T	T
idEmploye != null			F	T	T	T	T
idBureau != null				F	T	T	T
Employe.getTypeFonction != « manager » ∧ placeFixeEmp == null					F	T	T
Employe.getTypeFonction == « manager » ∧ placeFixeEmpLocalisation == null						F	T
idEmp’=idEmp							T
idBureau’=idBureau							T
invariant							T
Levée d’une exception	OUI	OUI	OUI	OUI	OUI	OUI	NON
Objet crée	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	1	1	1	1	2