

Projet CSC4102 : Gestion de bureaux en situation de bilocalisation

El Otmani Riad et Lahlali Saad
Année 2020–2021 — 01 avril 2021

TABLE DES MATIERES

1	Spécification.....	3
1.1	Diagrammes de cas d'utilisation.....	3
1.2	Priorités, préconditions et postconditions des cas d'utilisation.....	4
2	Préparation des tests de validation.....	8
2.1	Tables de décision des tests de validation.....	8
3	Conception.....	12
3.1	Liste des classes.....	12
3.2	Diagramme de classes.....	13
3.3	Diagrammes de séquence.....	14
	Figure 6 : affecter un employé à une place fixe.....	19
	– algorithme :.....	20
	– algorithme :.....	22
	– algorithme :.....	23
4	Fiche des classes.....	24
4.1	Classe BeBiloc.....	24
4.2	Classe Employé.....	24
4.3	Classe Bureau.....	25
4.4	Classe PlaceFixe.....	26
4.5	Classe PlaceDePassage.....	26
5	Diagrammes de machine à états et invariants.....	28
5.1	Classe Employé.....	28
5.2	Classe PlaceFixe.....	28
5.3	Classe Bureau.....	29
6	Préparation des tests unitaires.....	30
6.1	Classes Employé.....	30
6.2	Classes Bureau.....	31
6.3	Classes PlaceFixe.....	32

1 Spécification

1.1 Diagrammes de cas d'utilisation

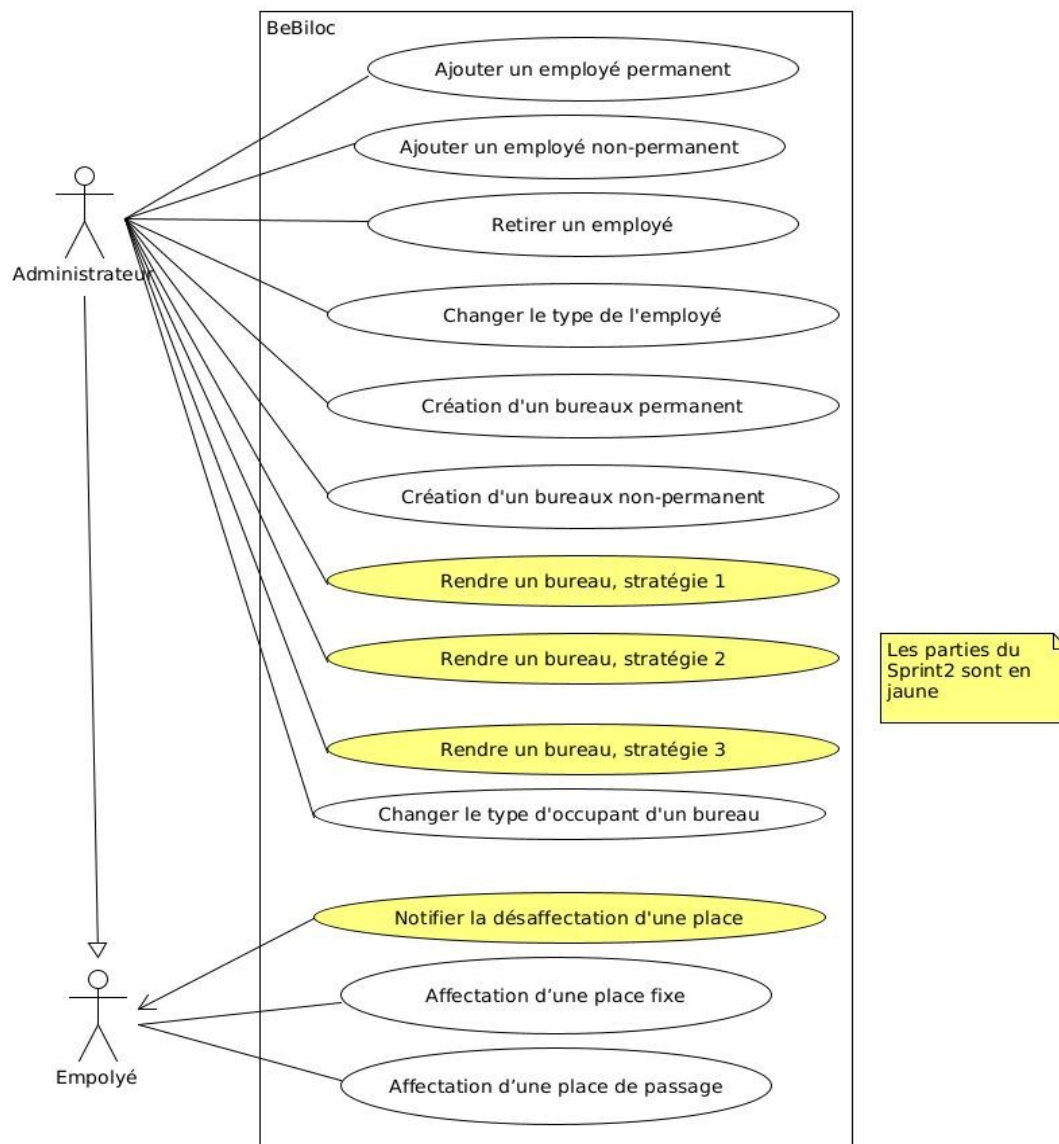


Figure 1: Diagramme de cas d'utilisation

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 utilisations : 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 :
 - \wedge identifiant de l'employé bien formé (non null et non vide)
 - \wedge nom bien formé (non null et non vide)
 - \wedge prénom bien formé (non null et non vide)
 - \wedge date d'embauche non null
 - \wedge fonction du permanent bien formée (non null et non vide)
 - \wedge fonction du permanent \in {direction département, direction adjointe département, assistance gestion, enseignement recherche}
 - \wedge employé avec cet identifiant inexistant
- Postcondition :
 - \wedge employé permanent avec cet identifiant existant

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

- Précondition :
 - \wedge identifiant de l'employé bien formé (non null et non vide)
 - \wedge nom bien formé (non null et non vide)
 - \wedge prénom bien formé (non null et non vide)
 - \wedge date d'embauche non null
 - \wedge date de fin de contrat non null
 - \wedge date de fin de contrat \geq date d'embauche

- \wedge fonction du non-permanent bien formée (non null et non vide)
- \wedge fonction du non-permanent $\in \{\text{doctorat, post-doctorat, ingénierie recherche, stage}\}$
- \wedge employé avec cet identifiant inexistant

– Postcondition :

- \wedge employé permanent avec cet identifiant existant

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

– Précondition :

- \wedge identifiant indiqué (non null et non vide)
- \wedge localisation (non null ou non vide)
- \wedge nombre de places total du bureau pour permanent $\in \{1, 2\}$
- \wedge bureau permanent contient une seule place fixe
- \wedge bureau avec cet identifiant inexistant

– Postcondition :

- \wedge bureau pour permanent avec cet identifiant existant

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

– Précondition :

- \wedge identifiant du bureau indiqué (non null et non vide)
- \wedge localisation (non null ou non vide)
- \wedge nombre de places total du bureau pour non permanent $\in \{1,2,3,4,5,6\}$
- \wedge bureau avec cet identifiant inexistant

– Postcondition :

- \wedge bureau pour non permanent avec cet identifiant existant

Priorité Moyenne n°1 : Retirer un employé

– Précondition :

- \wedge employé avec cet identifiant existant
- \wedge identifiant de l'employé bien formé (non null et non vide)

– Postcondition :

- \wedge identifiant inexistant

Priorité Moyenne n°2 : Affectation d'un employé à une place fixe

– Précondition :

- \wedge identifiant de l'employé bien formé (non null et non vide)
- \wedge identifiant du bureau indiqué (non null et non vide)
- \wedge employé avec cet identifiant existant
- \wedge bureau avec cet identifiant existant
- \wedge nombre de places fixes libre ≥ 1
- \wedge l'employé et le bureau ont le même type (permanent ou non permanent)
- \wedge ((« doctorat » OR « post-doctorat » OR « ingénierie de recherche » OR « stage » OR « assistance gestion » OR « enseignement recherche ») \wedge pas de place fixe sur aucun des deux

sites) OR (pas de place fixe sur le site actuel \wedge (« direction du département » OR « direction adjointe du département »))

– Postcondition :

\wedge la place fixe est attribué à l'employé

Priorité Moyenne n°3 : Affectation d'un employé à une place de passage

– Précondition :

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

\wedge employé avec cet identifiant existant

\wedge identifiant du bureau indiqué (non null et non vide)

\wedge bureau avec cet identifiant existant

\wedge l'employé et le bureau ont le même type (permanent ou non permanent)

\wedge date début d'affectation non null

\wedge date début affectation après la date d'aujourd'hui

\wedge date fin affectation non null

\wedge date fin affectation après la date de début d'affectation

\wedge date de début d'affectation est après la date d'embauche de l'employé

\wedge date de fin d'affectation est avant la date fin contrat de l'employé (nonpermanent)

\wedge nombre de places fixe de l'employé sur le site actuel vérifié (null)

\wedge nombre de places de passages sur le site et pendant la période (null)

\wedge nombre de places de passage libre ≥ 1

– Postcondition :

\wedge la place fixe est attribué à l'employé

Priorité Moyenne n°4 : Rendre un bureau, Stratégie 1

– Précondition :

\wedge identifiant de du bureau bien formé (non null et non vide)

\wedge bureau avec cet identifiant existant

– Postcondition :

\wedge notification à tous les employés qui avaient une place fixe sur ce bureau

\wedge retire à chaque employé la place fixe occupé

\wedge notification à tous les employés qui avaient une place de passage dont la date de fin d'affectation est \geq à la date actuelle sur ce bureau

\wedge retire à chaque employé l'occupation de passage occupée

\wedge ce bureau n'existe plus

Priorité Moyenne n°5 : Rendre un bureau, Stratégie 2

– Précondition :

\wedge identifiant de du bureau bien formé (non null et non vide)

\wedge bureau avec cet identifiant existant

\wedge aucune place fixe n'est affecté

– Postcondition :

\wedge notification à tous les employés qui avaient une place de passage dont la date de fin d'affectation est \geq à la date actuelle sur ce bureau

\wedge retire à chaque employé l'occupation de passage occupée

\wedge ce bureau n'existe plus

Priorité Moyenne n°6 : Rendre un bureau, Stratégie 3

– Précondition :

\wedge identifiant de du bureau bien formé (non null et non vide)

\wedge bureau avec cet identifiant existant

\wedge aucune place fixe n'est affecté

\wedge aucune place de passage dont la date de fin d'affectation est \geq à la date actuelle

– Postcondition :

\wedge ce bureau n'existe plus

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

– Précondition :

\wedge identifiant de l'employé existant

\wedge type en entrée $\in \{\text{permanent}, \text{non permanent}\}$

– Postcondition :

\wedge le type de l'employé a changé

Priorité faible n°2 : Changer le type d'occupant d'un bureau

– Précondition :

\wedge identifiant du bureau existant

\wedge si le bureau est pour non permanent, nombre de places $\in \{1, 2, 3, 4, 5, 6\}$

\wedge si le bureau est pour permanent, nombre de places $\in \{1, 2\}$

\wedge type d'occupant en entrée $\in \{\text{permanent}, \text{non permanent}\}$

– Postcondition :

\wedge 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 null non vide)	F	T	T	T	T	T	T	T
Nom bien formé (non null non vide)		F	T	T	T	T	T	T
Prénom bien formé (non null non vide)			F	T	T	T	T	T
Date d'embauche \neq null				F	T	T	T	T
Fonction du permanent bien formé (non null non vide)					F	T	T	T
Fonction du permanent \in {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 null non vide)	F	T	T	T	T	T	T	T	T	T
Nom bien formé (non null non vide)		F	T	T	T	T	T	T	T	T
Prénom bien formé (non null non vide)			F	T	T	T	T	T	T	T
Date d'embauche \neq null				F	T	T	T	T	T	T
Date de fin de contrat \neq null					F	T	T	T	T	T
Date de fin de contrat \geq date d'embauche						F	T	T	T	T
Fonction du non-permanent bien formé (non null non vide)							F	T	T	T
Fonction du non-permanent \in {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 indiqué (non null et non vide)	F	T	T	T	T
Localisation (non null ou non vide)		F	T	T	T
Nombre de places total du bureau pour permanent $\in \{1, 2\}$			F	T	T
Bureau permanent contient une seule place fixe				F	T
Bureau avec cet identifiant inexistant					F
Création acceptée	F	F	F	F	F
Nombre de jeux de test	2	2	2	1	2

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

Numéro de test	1	2	3	4	5
Identifiant du bureau indiqué (non null et non vide)	F	T	T	T	T
Localisation (non null ou non vide)		F	T	T	T
Nombre de places total du bureau pour non permanent $\in \{1,2,3,4,5,6\}$			F	T	T
Bureau avec cet identifiant inexistant				F	T
Création acceptée	F	F	F	F	T
Nombre de jeux de test	2	2	3	1	2

Tableau 5: Cas d'utilisation « affecter un employé à une place fixe ».

Numéro de test	1	2	3	4	5	6	7	8
Identifiant de l'employé bien formé (non null et non vide)	F	T	T	T	T	T	T	T
Identifiant du bureau indiqué (non null et non vide)		F	T	T	T	T	T	T
Employé avec cet identifiant existant			F	T	T	T	T	T
Bureau avec cet identifiant existant				F	T	T	T	T
Nombre de places fixes libre ≥ 1					F	T	T	T
L'employé et le bureau ont le même type (permanent ou non permanent)						F	T	T
((« doctorat » OR « post-doctorat » OR « ingénierie de recherche » OR « stage » OR « assistance gestion » OR « enseignement recherche ») \wedge pas de place fixe sur aucun des deux sites) OR (pas de place fixe sur le site actuel \wedge (« direction du département » OR « direction adjointe du département »))							F	T
Création acceptée	F	F	F	F	F	F	F	T
Nombre de jeux de test	2	2	1	1	1	1	8	8

Tableau 6: Cas d'utilisation « affecter un employé à une place de passage ».

Numéro de test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Identifiant de l'employé bien formé (non null et non vide)	F	T	T	T	T	T	T	T	T	T	T	T	T	T	T
Employé avec cet identifiant existant		F	T	T	T	T	T	T	T	T	T	T	T	T	T
Identifiant du bureau indiqué (non null et non vide)			F	T	T	T	T	T	T	T	T	T	T	T	T
Bureau avec cet identifiant existant				F	T	T	T	T	T	T	T	T	T	T	T
L'employé et le bureau ont le même type (permanent ou non permanent)					F	T	T	T	T	T	T	T	T	T	T
date début d'affectation non null						F	T	T	T	T	T	T	T	T	T
date début affectation après la date d'aujourd'hui							F	T	T	T		T	T	T	T
date fin affectation non null								F	T	T	T	T	T	T	T
date fin affectation après la date de début d'affectation									F	T	T	T	T	T	T
date de debut d'affectation est après la date d'embauche de l'employé (nonpermanent)										F	T	T	T	T	T
date de fin d'affectation est avant la date fin contrat de l'employé (nonpermanent)											F	T	T	T	T
Nombre de places fixe de l'employé sur le site actuel vérifié (null)												F	T	T	T
Nombre de places de passages sur le site et pendant la période (null)													F	T	T
Nombre de places fixes libre ≥ 1														F	T
Création acceptée	F	F	F	F	F	F	F	F	F	F	F	F	F	F	T
Nombre de jeux de test	2	1	2	1	2	1	1	1	1	1	1	1	1	1	1

Tableau 7 : Cas d'utilisation « : Rendre un bureau, Stratégie 1».

Numéro de test	1	2	3
Identifiant de du bureau bien formé (non null et non vide)	F	T	T
Bureau avec cet identifiant existant		F	T
Désindexation acceptée avec notification des employés	F	F	T
Nombre de jeux de test	2	1	1

Tableau 8 : Cas d'utilisation « : Rendre un bureau, Stratégie 2».

Numéro de test	1	2	3	4
Identifiant de du bureau bien formé (non null et non vide)	F	T	T	T
Bureau avec cet identifiant existant		F	T	T
Aucune place fixe n'est affectée			F	T
Désindexation acceptée avec notification des employés	F	F	F	T
Nombre de jeux de test	2	1	1	1

Tableau 9 : Cas d'utilisation « : Rendre un bureau, Stratégie 3».

Numéro de test	1	2	3	4	5
----------------	---	---	---	---	---

Identifiant de du bureau bien formé (non null et non vide)	F	T	T	T	T
Bureau avec cet identifiant existant		F	T	T	T
Aucune place fixe n'est affectée			F	T	T
Aucune place de passage dont la date de fin d'affectation est \geq à la date actuelle				F	T
Désindexation acceptée	F	F	F	F	T
Nombre de jeux de test	2	1	1	1	1

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 les places fixes, liste des identifiants des employés dans les places de passage
- PlaceFixe — identifiant, estLibre
- PlaceDePassage — identifiant, liste des occupations de passage
- OccupationDePassage — intervalle de dates, identifiant de l'employé et identifiant de la place
- Localisation (énumération) — 2 énumérateurs
- Notification — contenu
- ConsommateurAnnulationAffectation — idEmployé, souscription
- StrategieRendreBureau —
- RendreUnBureauS1 —
- RendreUnBureauS2 —
- RendreUnBureauS3 —

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

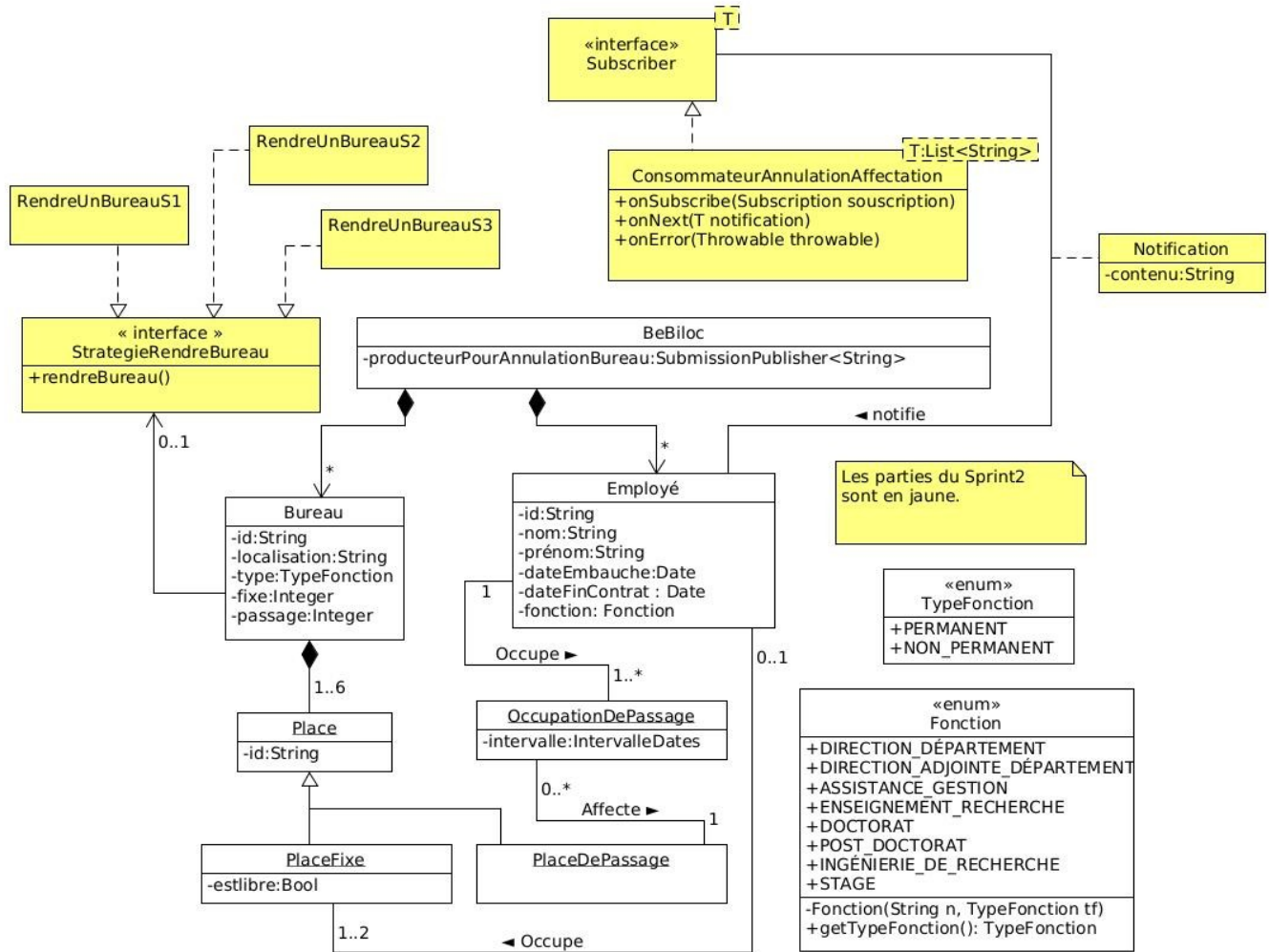


Figure 2: Diagramme de classes

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 null non vide) \wedge nom bien formé (non null et non vide) \wedge prénom bien formé (non null non vide) \wedge date d'embauche non null \wedge date de fin de contrat non null \wedge date de fin de contrat \geq date d'embauche \wedge fonction du non-permanent non null \wedge fonction du permanent $\in \{\text{doctorat, post-doctorat, ingénierie recherche, stage}\} \wedge$ 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

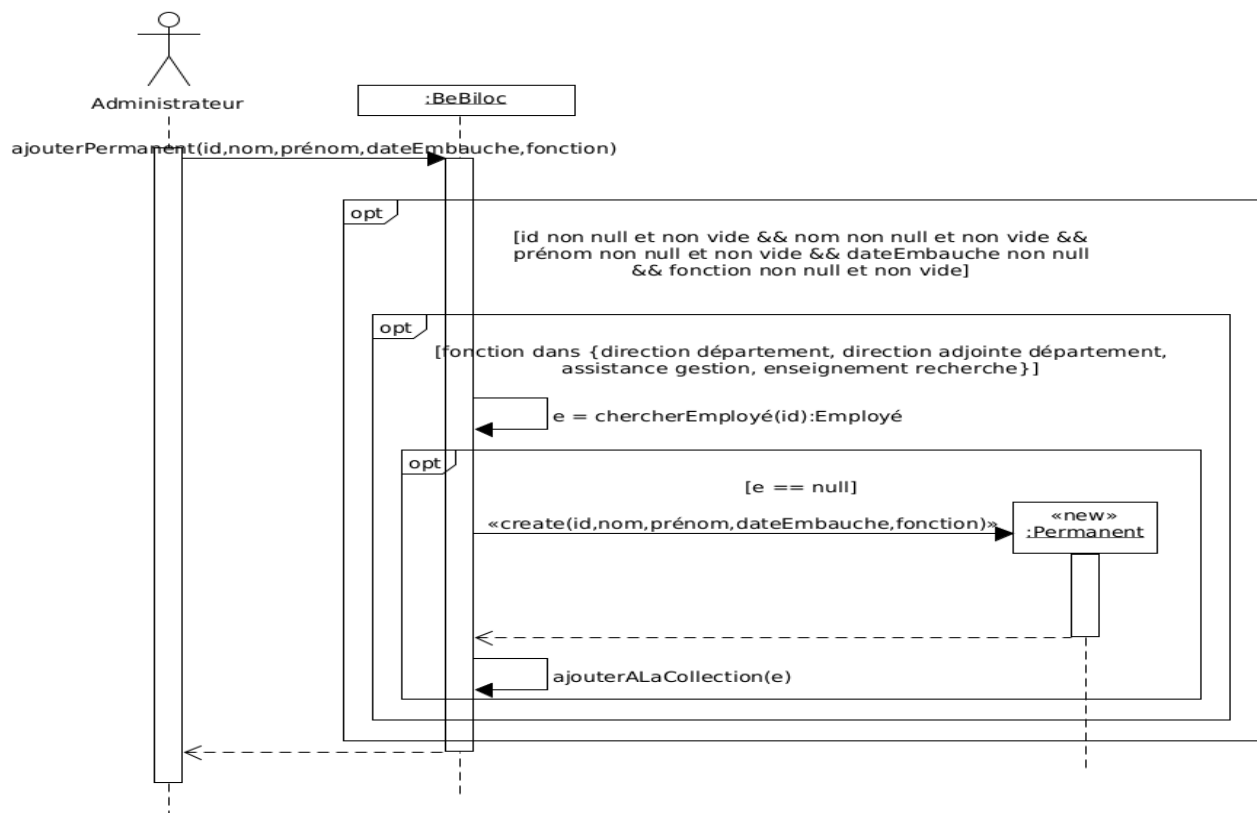


Figure 3: Diagramme de séquence du cas d'utilisation « ajouter un employé permanent »

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 null non vide) \wedge nom bien formé (non null et non vide) \wedge prénom bien formé (non null non vide) \wedge date d'embauche non null \wedge date de fin de contrat non null \wedge date de fin de contrat \geq date d'embauche \wedge fonction du non-permanent non null \wedge fonction du permanent $\in \{\text{doctorat, post-doctorat, ingénierie recherche, stage}\} \wedge$ 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

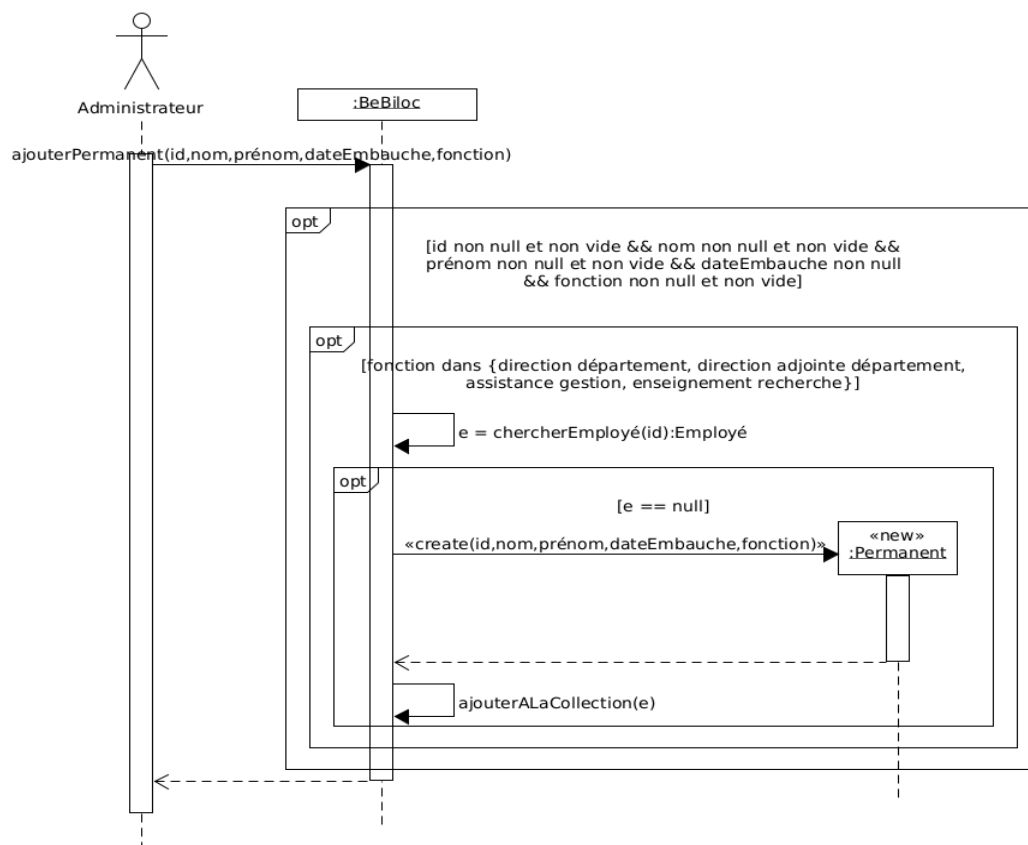
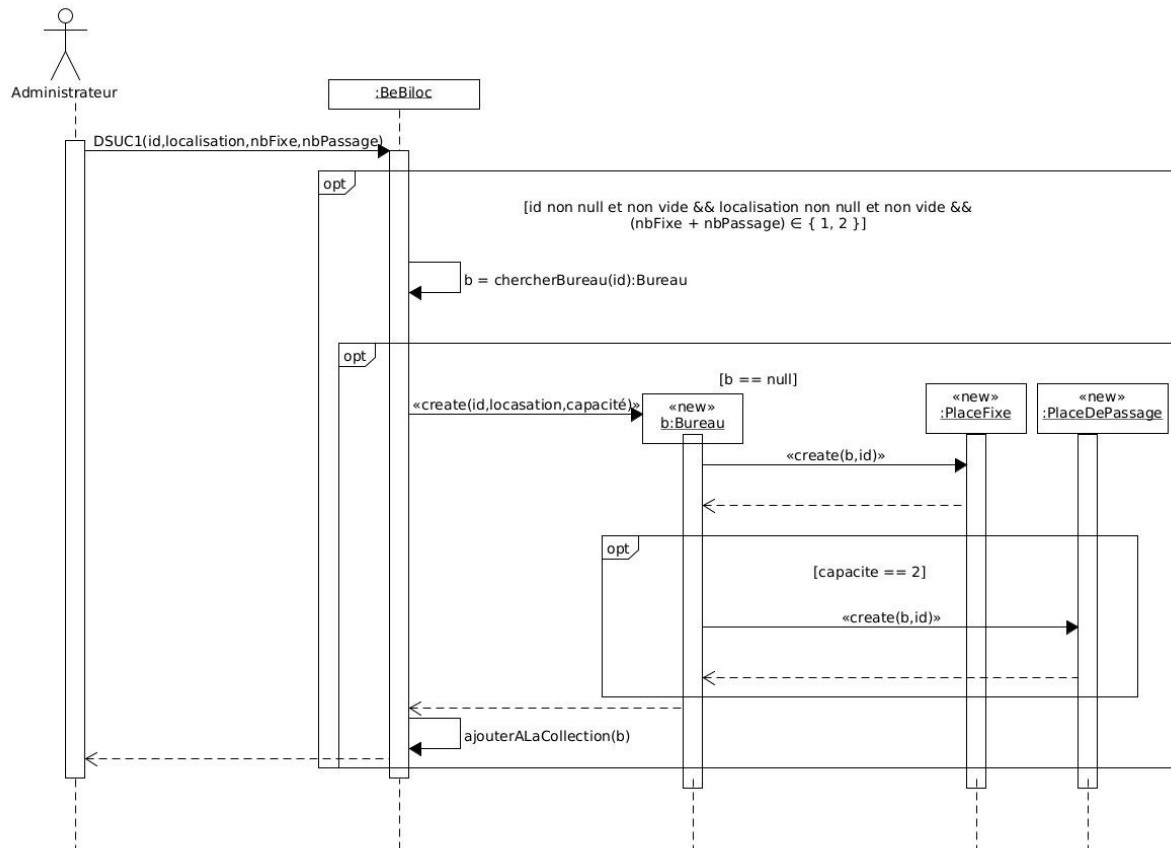


Figure 4: Diagramme de séquence du cas d'utilisation « ajouter un bureau pour permanents »



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 null et non vide) \wedge site de localisation (non null et non vide) \wedge nbFixe + nbPassage $\in [1..2]$ \wedge 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) \wedge site de localisation non null \wedge (nbFixes+nbPassage) \in [1..6] \wedge 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

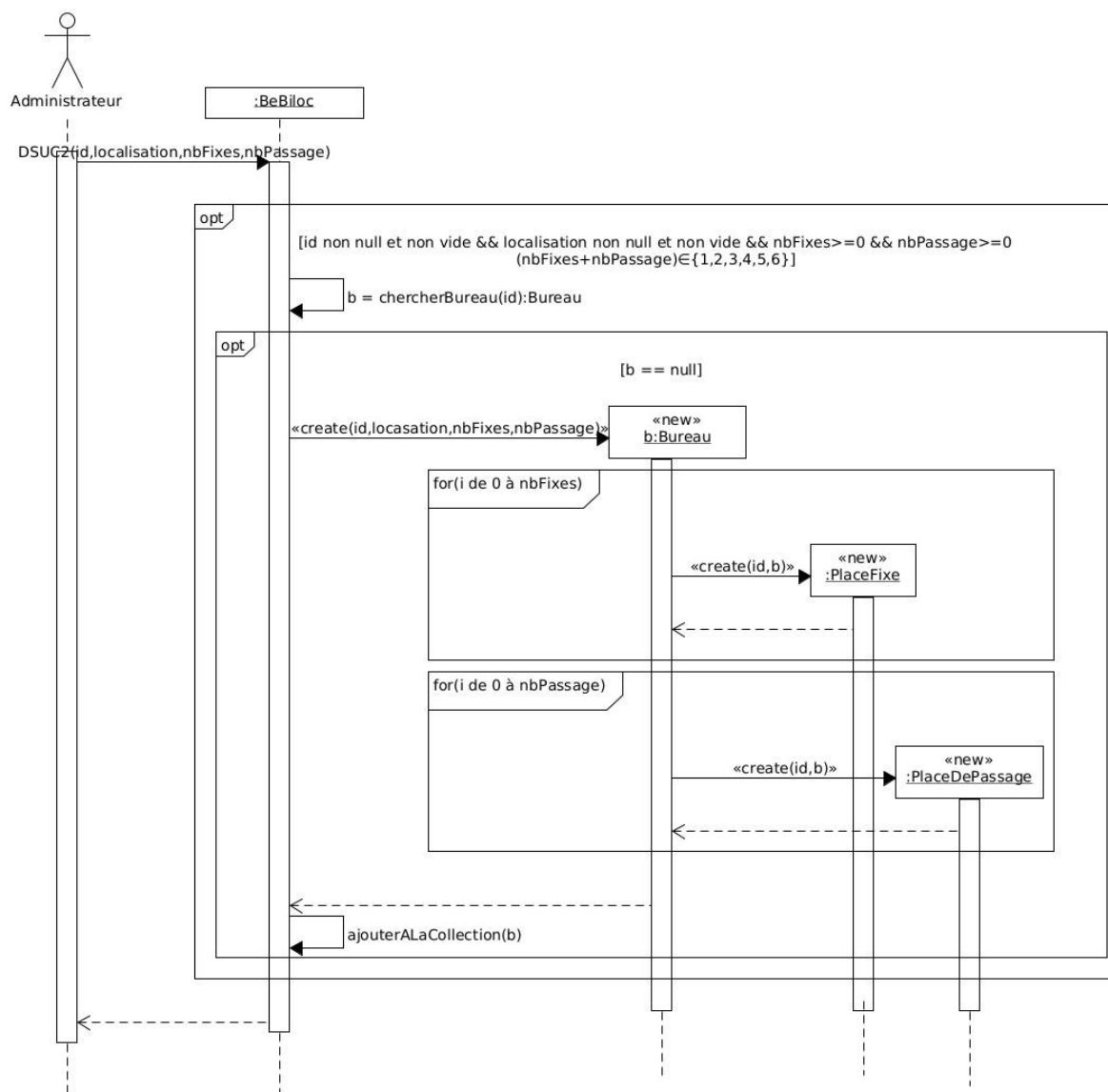


Figure 6 : ajouter un bureau pour non-permanents

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

- arguments en entrée : `identifiantEmployé` , `identifiantBureau`,
- rappel de la précondition :
 - \wedge `identifiant` de l'employé bien formé (non null et non vide)
 - \wedge `identifiant` du bureau indiqué (non null et non vide)
 - \wedge employé avec cet `identifiant` existant
 - \wedge bureau avec cet `identifiant` existant
 - \wedge nombre de places fixes libre ≥ 1
 - \wedge l'employé et le bureau ont le même type (permanent ou non permanent)
 - \wedge ((« doctorat » OR « post-doctorat » OR « ingénierie de recherche » OR « stage » OR « assistance gestion ») \wedge pas de place fixe sur aucun des deux sites) OR (pas de place fixe sur le site actuel \wedge (« direction du département » OR « direction adjointe du département »));
- algorithme :
 1. vérifier que l'identifiant de l'employé et du bureau sont non null et non vide
 2. cherche l'employé et le bureau dans les collections de `bebiloc`
 3. on vérifie que le bureau et l'employé existent
 4. on récupère la fonction de l'employé et du bureau (permanent/nonpermanent)
 5. on vérifie qu'ils ont bien la même fonction
 6. on récupère le nombre de places fixes total du bureau et le nombre de places fixes occupées
 7. on vérifie qu'il reste bien des places fixes libres
 8. on récupère la localisation du bureau
 9. on regarde si l'employé a atteint le nombre maximal de place en fonction de sa fonction
 10. on récupère une place fixe libre
 11. on l'attribue à l'employé
 12. on met à jour les attributs de la l

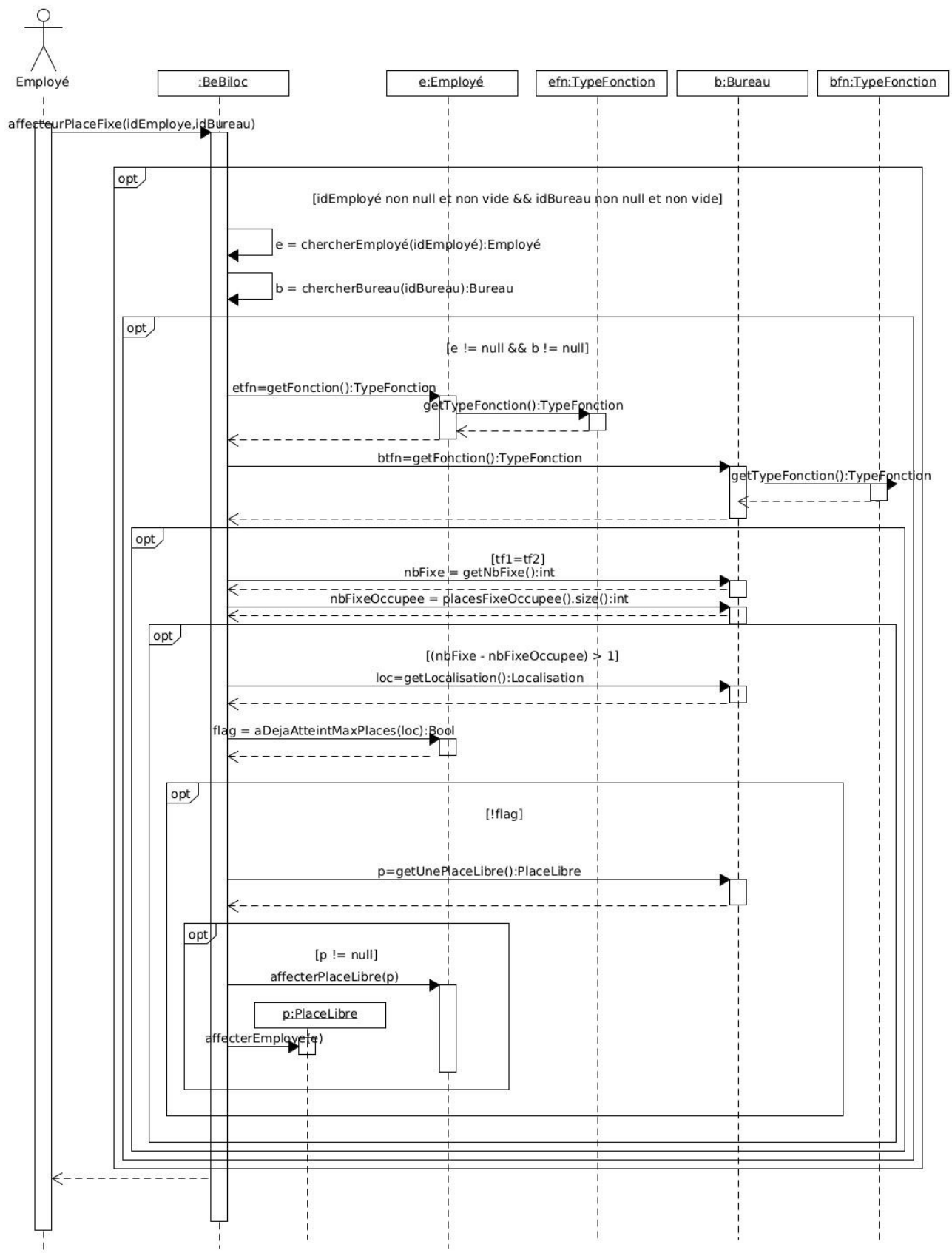


Figure 6 : affecter un employé à une place fixe

Voici la description textuelle du cas d'utilisation « rendre un bureau » :

- arguments en entrée : idBureau,
- rappel de la précondition :
 - \wedge bureau avec cet identifiant existant
- algorithme :
 1. vérifie qu l'identifiant entrée est non null et non vide
 2. chercher le bureau avec cet identifiant
 3. verifier que le bureau existe bien
 4. récupérer la liste des places fixes occupées dans le bureau
 5. récupérer la liste des places de passage occupées en cours ou dans le future
 7. Pour chaque place fixe occupée dans le bureau :
 - i/ on récupère l'identifiant de l'employé occupant cette place :
 - ii/ on cherche cet employé dans la collection
 - iii/ on notifie l'employé que cette place fixe sera libéré
 8. Pour chaque place de passage occupées en cours ou dans le future :
 - i/ on récupère la liste des occupations de passage de la place
 - ii/ Pour chaque occupation :
 - a- on récupère l'employé affecté à cette occupation
 - b- on notifie cet employé que son occupation va être perdu
 8. rendre le bureau

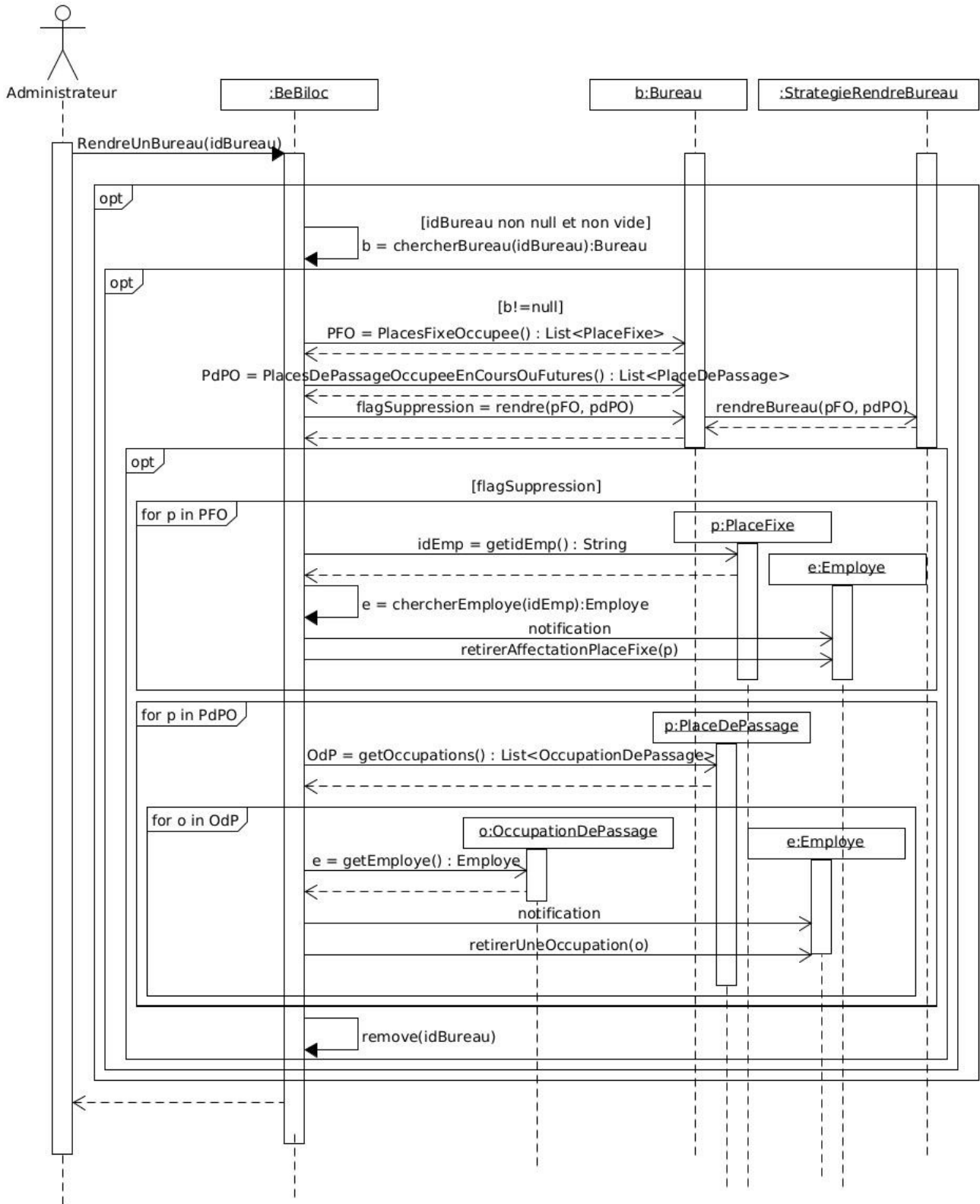


Figure 7 : rendre un bureau, stratégie 1

Voici la description textuelle du cas d'utilisation « rendre un bureau, stratégie 2 » :

- arguments en entrée : idBureau,
- rappel de la précondition :
 - ∧ bureau avec cet identifiant existant
- algorithme :
 1. vérifie qu l'identifiant entrée est non null et non vide
 2. chercher le bureau avec cet identifiant
 3. verifier que le bureau existe bien
 4. récupérer la liste des places fixes occupées dans le bureau
 5. on vérifie que la liste des places fixes occupées dans le bureau est nulle
 6. récupérer la liste des places de passage occupées en cours ou dans le future
 7. Pour chaque place de passage occupées en cours ou dans le future :
 - i/ on récupère la liste des occupations de passage de la place
 - ii/ Pour chaque occupation :
 - a- on récupère l'employé affecté à cette occupation
 - b- on notifie cet employé que son occupation va être perdu
 8. rendre le bureau

Voici la description textuelle du cas d'utilisation « rendre un bureau, stratégie 3 » :

- arguments en entrée : idBureau,
- rappel de la précondition :
 - \wedge bureau avec cet identifiant existant
- algorithme :
 1. vérifie qu le'identifiant entrée est non null et non vide
 2. chercher le bureau avec cet identifiant
 3. verifier que le bureau existe bien
 4. récupérer la liste des places fixes occupées dans le bureau
 5. on vérifie que la liste des places fixes occupées dans le bureau est nulle; si c'est le cas notification refut du système
 6. récupérer la liste des places de passage occupées en cours ou dans le future
 7. on vérifie que la liste des places de passage occupées dans le bureau est nulle
 8. rendre le bureau

4 Fiche des classes

4.1 Classe BeBiloc

BeBiloc
<p><- attributs « association » -></p> <ul style="list-style-type: none">- listeBureaux : collection de Bureau- employes : collection de Employe
<p><- constructeur -></p> <ul style="list-style-type: none">+ BeBiloc()+ invariant() : booléen
<p><- operations « cas d'utilisation » -></p> <ul style="list-style-type: none">+ ajouterPermanent(String id, String nom, String prenom, LocalDate dateEmbauche, String fonction)+ ajouterNonPermanent(String id, String nom, String prenom, LocalDate dateEmbauche, LocalDate dateFinContrat, String fonction)+ 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)+ rendreBureau(String idBureau)+ rendreBureauS1(String idBureau)+ rendreBureauS2(String idBureau)+ rendreBureauS3(String idBureau)+ ajouterConsommateurPourNotificationAnnulationPlace(String idEmploye)+ getListeBureaux()+ listerEmployes()+ listerEmployesPermanents()+ listerEmployesNonPermanents()

4.2 Classe Employé

Employé
<ul style="list-style-type: none">- id : String- nom : String- prénom : String- dateEmbauche : Date- dateFinContrat : Date- fonction : Fonction
<p><- attributs « association » -></p>

<ul style="list-style-type: none"> - fonction : TypeFonction - placesFixesReservees : collection de places fixes occupées - occupationsDePassage : collection de places de Passage occupées - producteurPourAnnulationBureau : producteur pour annulation bureau
<- constructeur -> + Employé(String id, String nom, String prénom, LocalDate dateEmbauche, Fonction fonction) + Employé(String id, String nom, String prénom, LocalDate dateEmbauche, LocalDate dateFinContratFonction fonction) + aDejaBureauSurLocalisation(Localisation localisation) + aDejaAtteintMaxPlaces(Localisation localisation) + aDejaPlaceDePassageSurLocalisation(Localisation localisation, LocalDate dateDebutAffectation, LocalDate dateFinAffectation) + affecterPlaceLibre(PlaceFixe placeFixe) + getProducteurPourAnnulationBureau() + setProducteurPourAnnulationBureau() + ConsommateurAnnulationAffectation() + setConsommateur() + invariant() : booléen + getId() + getNom() + getPrenom() + getDateEmbauche() + getDateFinContrat() + getFonction() + getTypeFonction() + getPlacesFixesReservees() + getOccupationsDePassage() + retirerAffectationPlaceFixe(PlaceFixe placeFixe) + retirerUneOccupation(OccupationDePassage occupation) + affecterPlaceDePassage(PlaceDePassage placeDePassage, LocalDate dateDebut, LocalDate dateFin) + hashCode() // override + equals() // override + toString() //override

4.3 Classe Bureau

Bureau
<ul style="list-style-type: none"> - idBureau : String - localisation : String - fixe : int - passage : int
<- attributs « association » -> - places : collection des places du bureau - type : TypeFonction - strategieRendreBureau : stratégie pour rendre ce bureau
<- constructeur ->

```

+ Bureau(String idBureau, Localisation localisation, TypeFonction type, int fixe, int
passage)
+ Bureau(String idBureau, Localisation localisation, int fixe, int passage)
+ creerPlaces()
+ getUnePlaceFixeLibre()
+ invariant()
+ getId()
+ getLocalisation()
+ getTypeFonction()
+ getNbFixe()
+ getNbPassage()
+ getPlaces()
+ placesFixeOccupee()
+ placesDePassageOccupeeEnCoursOuFutures()
+ placesDePassageLibreDate( LocalDate dateDebutAffectation, LocalDate
dateFinAffectation)
+ setStrategieRendreBureau(StrategieRendreBureau s)
+ rendre(List <PlaceFixe> pFO, List<PlaceDePassage> pdPO)
+ hashCode() // override
+ equals() // override
+ toString() //override

```

4.4 Classe PlaceFixe

PlaceFixe
-id:String -estLibre:Bool <- attributs « association » -> - idEmp: identifiant de l'employé occupant la place
<- constructeur -> + PlaceFixe(bureau Bureau, String id) + invariant() + getidEmp() + estLibre() + affecterEmploye(Employe employe) + retirerAffectation() + hashCode() // override + equals() // override + toString() //override

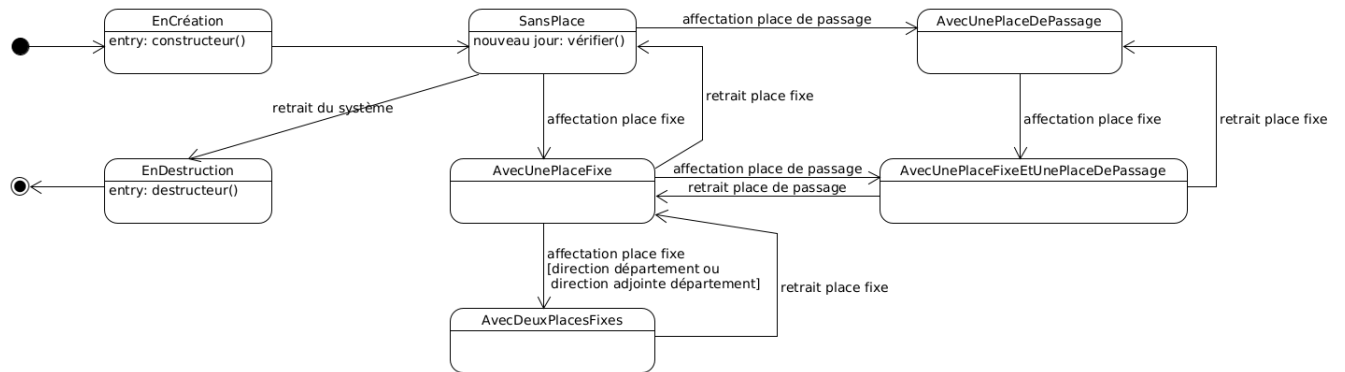
4.5 Classe PlaceDePassage

PlaceDePassage
-id:String

<- attributs « association » -> - occupations : occupations de cette place de passage
<- constructeur -> + PlaceDePassage(String idPlace, Bureau bureau) + invariant() + getOccupations() + estOccupee() + ajouterOccupationDePassage(OccupationDePassage nouvelleOccupation) + retirerOccupations() + dejaOccupeeACetteDate(LocalDate dateDebutAffectation, LocalDate dateFinAffectation) + hashCode() // override + equals() // override + toString() //override

5 Diagrammes de machine à états et invariants

5.1 Classe Employé



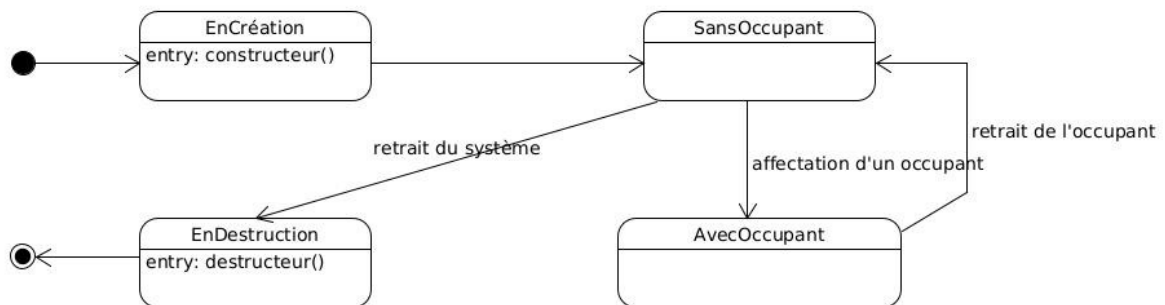
Pour garder l'invariant quelque peu local à la classe, c'est-à-dire sans trop d'appels sur d'autres objets métier, les conditions suivantes ne sont pas intégrées à l'invariant :

- 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 :

$\wedge \text{identifiant} \neq \text{null} \wedge \text{identifiant} \neq \text{vide}$
 $\wedge \text{nom} \neq \text{null} \wedge \text{nom} \neq \text{vide}$
 $\wedge \text{prenom} \neq \text{null} \wedge \text{prenom} \neq \text{vide}$
 $\wedge \text{dateEmbauche} \neq \text{null} \wedge \text{fonction} \neq \text{null}$
 $\wedge \text{fonction} = \text{permanent} \Rightarrow \text{dateFinContrat} = \text{null}$
 $\wedge \text{fonction} \neq \text{permanent} \wedge \text{date} \neq \text{null} \Rightarrow \text{FinContrat} \neq \text{null}$

5.2 Classe PlaceFixe



L'invariant de la classe PlaceFixe est le suivant :

$\wedge \text{bureau} \neq \text{null}$
 $\wedge (\text{idPlace} \neq \text{null} \wedge \text{idPlace} \neq \text{vide})$

5.3 Classe Bureau

L'invariant de la classe Bureau est le suivant :

$\wedge \text{identifiant} \neq \text{null} \wedge \text{identifiant} \neq \text{vide}$

$\wedge \text{localisation} = \text{Evry} \parallel \text{localisation} = \text{Palaiseau}$

$\wedge \text{fixe} \neq \text{null} \wedge \text{fixe} \neq \text{vide}$

$\wedge \text{passage} \neq \text{null} \wedge \text{passage} \neq \text{vide}$

$\wedge \text{type} = \text{PERMANENT} \parallel \text{type} = \text{NON_PERMANENT}$

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
<i>identifiant $\neq null \wedge \neq vide$</i>	F	T	T	T	T	T	T
<i>nom $\neq null \wedge \neq vide$</i>		F	T	T	T	T	T
<i>prénom $\neq null \wedge \neq vide$</i>			F	T	T	T	T
<i>dateEmbauche $\neq null$</i>				F	T	T	T
<i>fonction $\neq null$</i>					F	T	T
<i>fonction $\in \{\text{direction département, direction adjointe département, assistance gestion, enseignement recherche}\}$</i>						F	T
<i>identifiant'=identifiant</i>							T
<i>nom'=nom</i>							T
<i>prenom'=prenom</i>							T
<i>dateEmbauche'=dateEmbauche</i>							T
<i>dateFinContrat'=null</i>							T
<i>fonction'=fonction</i>							T
<i>invariant</i>							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
<i>identifiant $\neq null \wedge \neq vide$</i>	F	T	T	T	T	T	T	T
<i>nom $\neq null \wedge \neq vide$</i>		F	T	T	T	T	T	T
<i>prénom $\neq null \wedge \neq vide$</i>			F	T	T	T	T	T
<i>dateEmbauche $\neq null$</i>				F	T	T	T	T
<i>dateFinContrat $\neq null$</i>					F	T	T	T
<i>dateFinContrat \geq dateEmbauche</i>					F	T	T	T
<i>fonction $\neq null$</i>						F	T	T
<i>fonction $\in \{\text{doctorat, post-doctorat, ingénierie recherche, stage}\}$</i>							F	T
<i>identifiant'=identifiant</i>								T
<i>nom'=nom</i>								T
<i>prenom'=prenom</i>								T
<i>dateEmbauche'=dateEmbauche</i>								T
<i>dateFinContrat'=dateFinContrat</i>								T
<i>fonction'=fonction</i>								T
<i>invariant</i>								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

6.2 Classes Bureau

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

Numero de test	1	2	3	4
$\text{identifiant} \neq \text{null} \wedge \text{identifiant} \neq \text{vide}$	F	T	T	T
$\text{localisation} \in \{\text{Evry}, \text{Palaiseau}\}$		F	T	T
$\text{NbFixe} + \text{nbPassage} \in [1..2]$			F	T
$\text{id}' = \text{id}$				T
$\text{localisation}' = \text{localisation}$				T
$\text{nbFixe} + \text{nbPassage}' = \text{nbFixe} + \text{nbPassage}$				T
$\text{type}' = \text{type}$				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
$\text{identifiant} \neq \text{null} \wedge \text{identifiant} \neq \text{vide}$	F	T	T	T	T	T	T
$\text{localisation} \in \{\text{Evry}, \text{Palaiseau}\}$		F	T	T	T	T	T
$\text{nbfixe} \geq 0$			F	T	T	T	T
$\text{nbpassage} \geq 0$				F	T	T	T
$\text{type} \neq \text{null} \wedge \text{type} \neq \text{vide}$					F	T	T
$(\text{nbfixe} + \text{nbpassage}) \leq 6$						F	T
$\text{id}' = \text{id}$							T
$\text{localisation}' = \text{localisation}$							T
$\text{nbfixe}' = \text{nbfixe}$							T
$\text{nbpassage}' = \text{nbpassage}$							T
$\text{type}' = \text{type}$							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

6.3 Classes PlaceFixe

Tableau 7 : Méthode constructeurPlaceFixe de la classe PlaceFixe

Numero de test	1	2	3
id != null et != ''	F	T	T
Bureau existe		F	T
id'=id			T
idBureau'=idBureau			T
invariant			T
Levée d'une exception	OUI	OUI	OUI
Objet crée	F	F	T
Nombre de jeux de test	2	1	1

Tableau 8 : Méthode affecterEmploye de la classe PlaceFixe

Numero de test	1	2
Employe existe	F	T
id'=id		T
EstLibre = false		T
invariant		T
Levée d'une exception	OUI	OUI
Objet crée	F	T
Nombre de jeux de test	1	1