Projet CSC4102 : « Suivi d’activité de projet »

Juliette Debono et Nathan Feret

Année 2022–2023 — 1er février 2023

1 Spécification

1.1 Diagrammes de cas d’utilisation

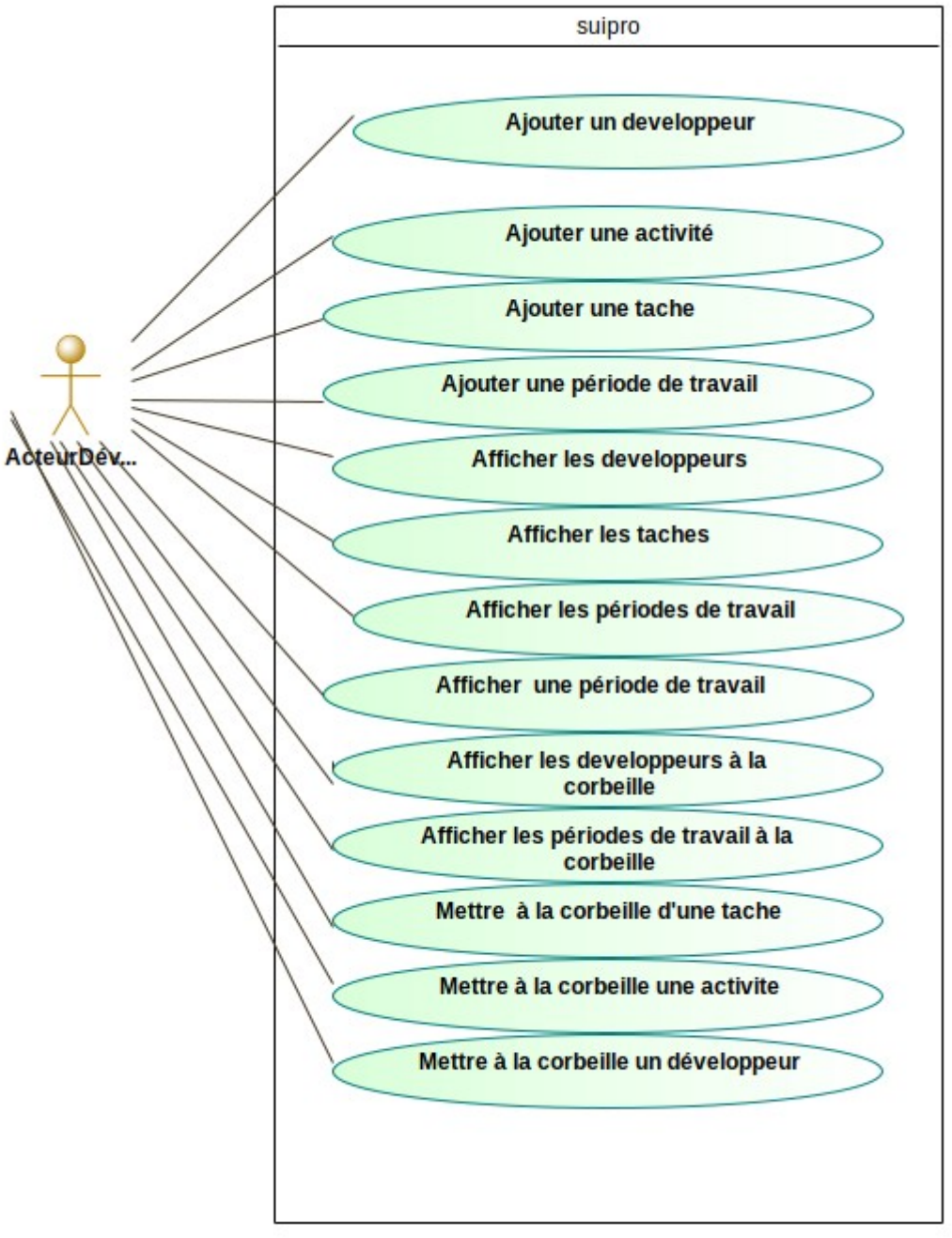

FIG. 1: Diagramme de cas d’utilisation — partie développeur.

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

Voici les préconditions et postconditions des cas d'utilisation du premier sprint :

Haute

Ajouter un développeur

- Précondition :
  - ∧ alias du développeur bien formé (non null et non vide)
  - ∧ nom bien formé (non null et non vide)
  - ∧ prénom bien formé (non null et non vide)
  - ∧ développeur avec cet alias inexistant
- Postcondition :

  développeur avec cet alias existant

Ajouter une activité
Ajouter une tache
-Précondition
^Intitulé bien formé
^tâche avec cet intitulé inexistante
- Postcondition
^tâche avec cet intitulé existant.

Ajouter une période de travail
Mettre à la corbeille une tache
- Précondition
^ La tâche existe
^ Intitulé bien formé

- Postcondition
^La tache est déplacée dans la corbeille
^Toutes les périodes de travail associées à la tache sont déplacées dans la corbeille

Mettre à la corbeille un développeur
-Précondition
∧ Le développeur existe
∧ nom bien formé (non null et non vide)
∧ prénom bien formé (non null et non vide)

-Postcondition
^Le développeur est déplacé dans la corbeille
^Toutes les périodes de travail associées au développeur sont déplacées dans la corbeille

Moyenne

Lister les développeurs

Afficher une période de travail

Afficher un développeur de la corbeille
Afficher une tache
Afficher une période de travail

Basse
Mettre à la corbeille une activité
Mettre à la corbeille une période de travail

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.

Numéro de test	1	2	3	4	5
Alias du développeur bien formé (non null et non vide)	F	T	T	T	T
Nom bien formé (non null et non vide)		F	T	T	T
Prénom bien formé (non null et non vide)			F	T	T
Développeur avec cet alias inexistant				F	T
Création acceptée	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	1

TAB. 1: Cas d'utilisation « Ajouter un développeur »

Numéro de test	1	2	3
Intitulé de la tâche bien formé (non null et non vide)	F	T	T
Tâche avec cet intitulé inexistante		F	T
Création acceptée	F	F	T
Nombre de jeux de test	2	2	1

TAB. 2: Cas d'utilisation « Ajouter une tâche»

Numéro de test	1	2	3
Intitulé de la tâche bien formé (non null et non vide)	F	T	T
Tâche avec cet intitulé existante		F	T
Mise à la corbeille acceptée	F	F	T
Nombre de jeux de test	2	2	1

TAB. 3: Cas d'utilisation « Mettre à la corbeille une tâche »

Numéro de test	1	2	3	4	5
Alias du développeur bien formé (non null et non vide)	F	T	T	T	T
Nom bien formé (non null et non vide)		F	T	T	T
Prénom bien formé (non null et non vide)			F	T	T
Développeur avec cet alias existant				F	T
Toutes les périodes de travail associées au développeur sont déplacées dans la corbeille	F	F	F	F	T
Mise à la corbeille acceptée	F	F	F	F	T
Nombre de jeux de test	2	2	2	1	1

TAB. 4: Cas d'utilisation «Mettre à la corbeille un développeur »

3 Conception

3.1 Diagramme de classes

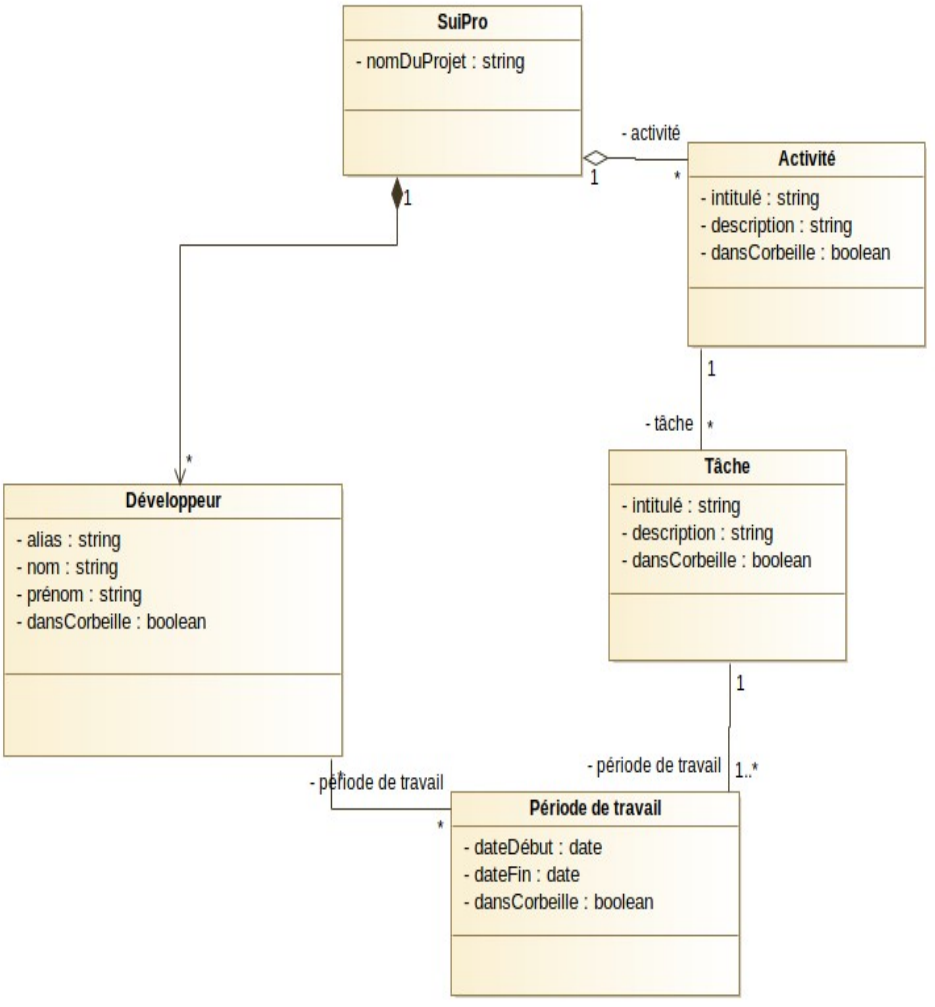

FIG. 2: Diagramme de classes.

3.2 Diagrammes de séquence

- arguments en entrée : l'alias du développeur, et le nom et le prénom du développeur
- rappel de la précondition : alias bien formé (non nullet non vide) ∧ nom bien formé (non nullet non vide) ∧ prénom bien formé (non nullet non vide) ∧ développeur avec cet identifiant inexistant
- algorithme :
    1. vérifier les arguments
    2. chercher un développeur avec cet alias
    3. vérifier que le développeur est inexistant
    4. instancier le développeur
    5. ajouter le développeur dans la collection des développeurs

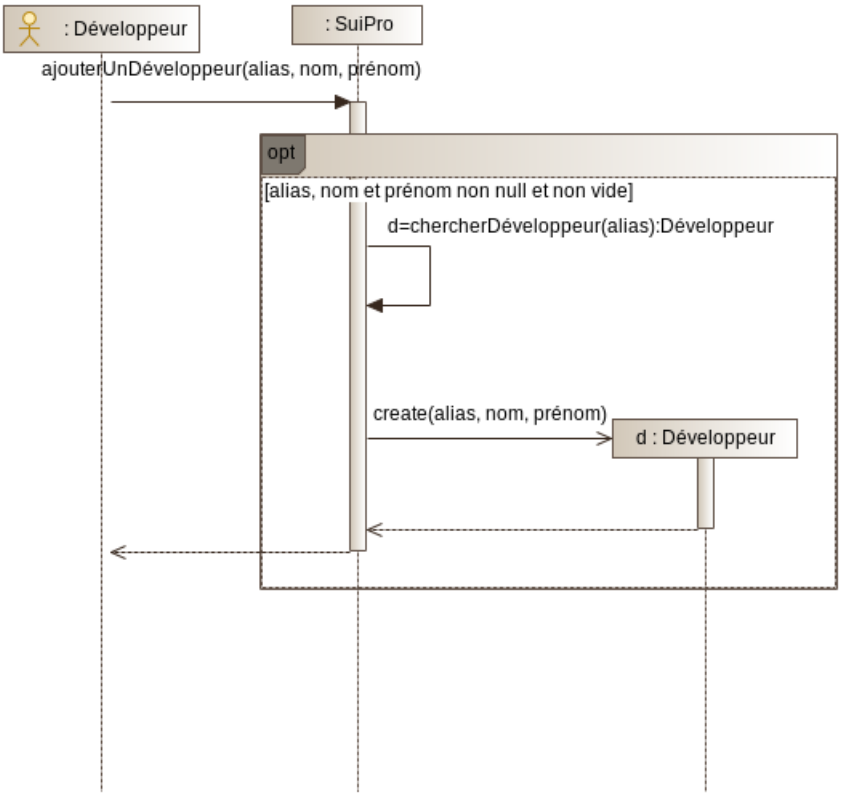

FIG. 3: Diagramme de séquence du cas d'utilisation « ajouter un développeur »

Création de tâche :

– arguments en entrée : nom de la tache, nom de l'activité
– rappel de la précondition : nom de la tâche bien formé (non nullet non vide) ^ activité existante
– algorithme :

1. vérifier les arguments
2. chercher l'activité avec son nom
3. vérifier que l'activité existe
4. chercher une tâche avec ce nom, associée à l'activité
5. vérifier que la tâche est inexistante pour cette activité
6. instancier la tâche
7. ajouter la tâche dans la collection des tâches de cette activité

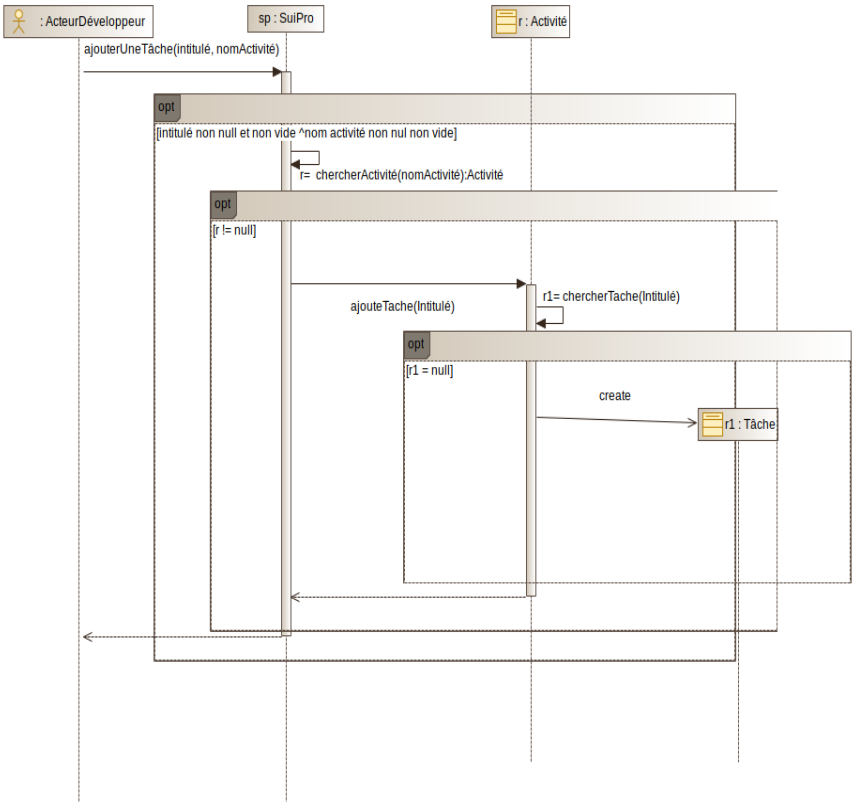

FIG. 3: Diagramme de séquence du cas d'utilisation « ajouter une tache»

Mettre a la corbeille une tâche.

– arguments en entrée : nom de la tache, nom de l'activité

– rappel de la précondition : nom de la tâche bien formé (non nullet non vide) ^ activité existante ^ tache existante dans l'activité

– algorithme :

1. vérifier les arguments
2. chercher l'activité avec son nom
3. vérifier que l'activité existe
4. chercher une tâche avec ce nom, associée à l'activité
5. vérifier que la tâche est existe pour cette activité
6. modifier l'attribut corbeille de tache à True

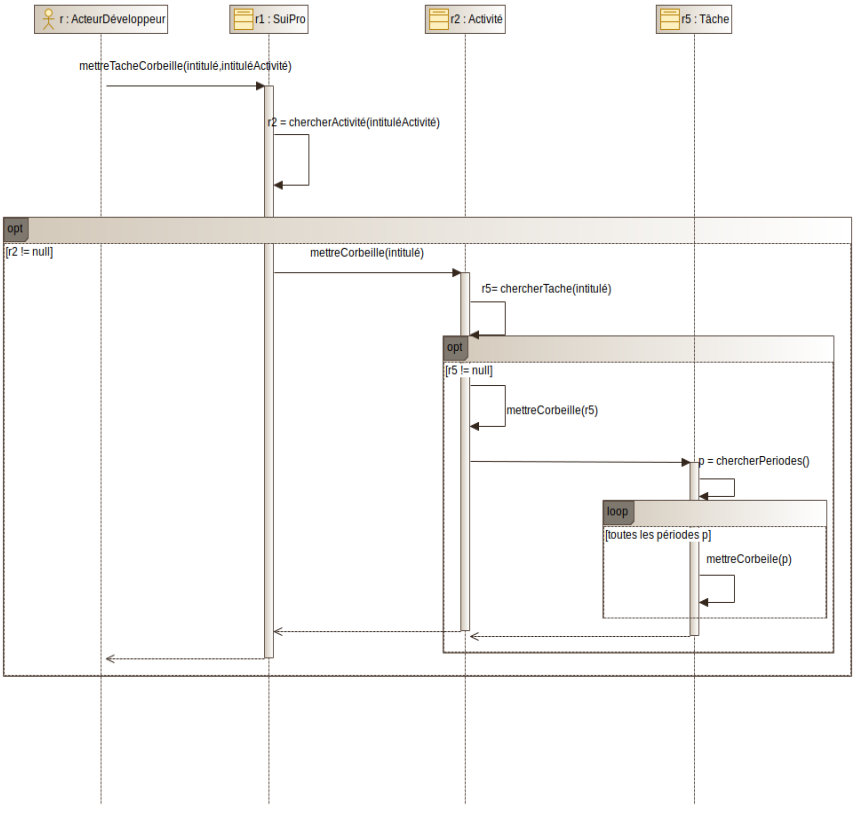

FIG. 3: Diagramme de séquence du cas d'utilisation «mettre une tache à la corbeille»

4 Diagrammes de machine à états et invariants

4.1 Classe Développeur

La section est à mettre à jour. Ce commentaire est à retirer ensuite.

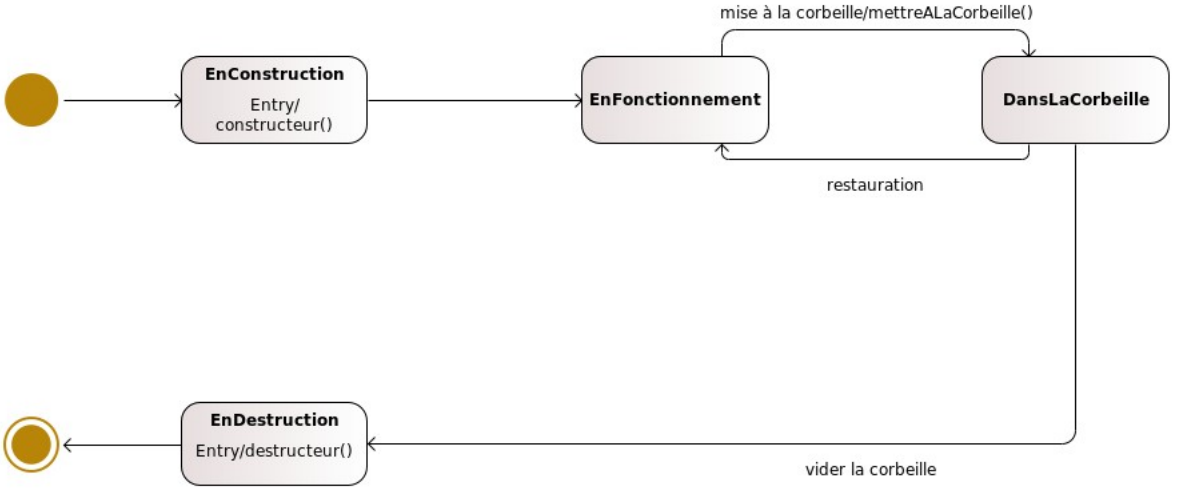

FIG. 4: Diagramme de machine à états de la classe Développeur

L'invariant de la classe Développeur est le suivant :

∧ *alias≠null* ∧ *alias≠vide*

∧ *nom≠null* ∧ *nom≠vide*

∧ *prenom≠null* ∧ *prenom≠vide*

4.2 Classes Tâche

La section est à compléter. Ce commentaire est à retirer ensuite.

FIG. 5: Diagramme de machine à états de la classe Tâche.

L'invariant de la classe Tâche est le suivant :

∧ ...

5 Fiche des classes

5.1 Classe Développeur

La section est à mettre à jour. Ce commentaire est à retirer ensuite.

Développeur
<– **attributs « association »** –> − periodesDeTravail : Liste<PeriodeDeTravail>
<– **attributs « modifiables »** –>− alias : String − nom : String − prénom : String <– **opérations** –> + constructeur(String identifiant, String description) + invariant() : boolean + getIdentifiant() : String + getDescription() : String + afficher() : List<String>

5.2 Classe Tâche

La section est à compléter. Ce commentaire est à retirer ensuite.

6 Préparation des tests unitaires

6.1 Classe Développeur

La section est à mettre à jour. Ce commentaire est à retirer ensuite.

Numéro de test	1	2	3	4
alias ≠*null* ∧ ≠*vide*	F	T	T	T
nom ≠*null* ∧ ≠*vide*		F	T	T
prenom ≠*null* ∧ ≠*vide*			F	T
alias'=*alias*				T
nom'=*nom*				T
prenom'=*prenom*				T
invariant				T
Levée d'une exception	OUI	OUI	OUI	NON
Objet créé	F	F	F	T
Nombre de jeux de test	2	2	2	1

TAB. 2: Méthode constructeur de la classe Développeur

6.2 Classe Tâche

La section est à compléter. Ce commentaire est à retirer ensuite.