



X



# Bilan de gestion de projet

## **Auteurs :Equipe 2**

Mehdi El Idrissi El Fatmi

Khadija El Adnani

Alaa Jennine

Badr El Khoundafi

Yassine El Kouri

22/01/2025

# Contents

<b>1</b>	<b>Organisation Adoptée dans l'Équipe</b>	<b>2</b>
1.1	Charte d'Équipe . . . . .	2
1.2	Outils Utilisés . . . . .	3
<b>2</b>	<b>Historique et Déroulement du Projet</b>	<b>3</b>
2.1	Chronologie des étapes . . . . .	4
2.2	Déroulé Général du Projet . . . . .	4
2.3	Étape B : Vérifications Contextuelles . . . . .	5
2.4	Étape C : Génération de Code Assembleur . . . . .	5
2.5	Tests de Validation . . . . .	6
2.6	Documentation . . . . .	6
2.7	Impact des Vacances sur la Cadence de Travail . . . . .	7
<b>3</b>	<b>Analyse Critique</b>	<b>7</b>
<b>4</b>	<b>Leçons Apprises</b>	<b>8</b>
4.1	Gestion de Projet . . . . .	8
4.2	Aspects Techniques . . . . .	8
<b>5</b>	<b>Conclusion</b>	<b>8</b>

# Bilan de Gestion d'Équipe et de Projet

## Introduction

Le développement d'un projet informatique collaboratif demande une organisation efficace et une répartition judicieuse des rôles au sein de l'équipe. Dans le cadre de notre projet, qui portait sur la conception d'un compilateur pour le langage Deca, l'accent a été mis sur la gestion proactive des forces et des faiblesses individuelles. Une charte d'équipe a été établie dès le début pour identifier les compétences spécifiques de chaque membre et distribuer les responsabilités de manière optimale. Cette approche a permis de maximiser l'efficacité collective tout en favorisant un environnement de travail harmonieux.

## 1 Organisation Adoptée dans l'Équipe

### 1.1 Charte d'Équipe

Dès les premières réunions, une charte d'équipe a été élaborée pour structurer notre collaboration. Elle reposait sur les principes suivants :

**Identification des forces et faiblesses :** Chaque membre a partagé ses compétences techniques et ses préférences en termes de tâches.

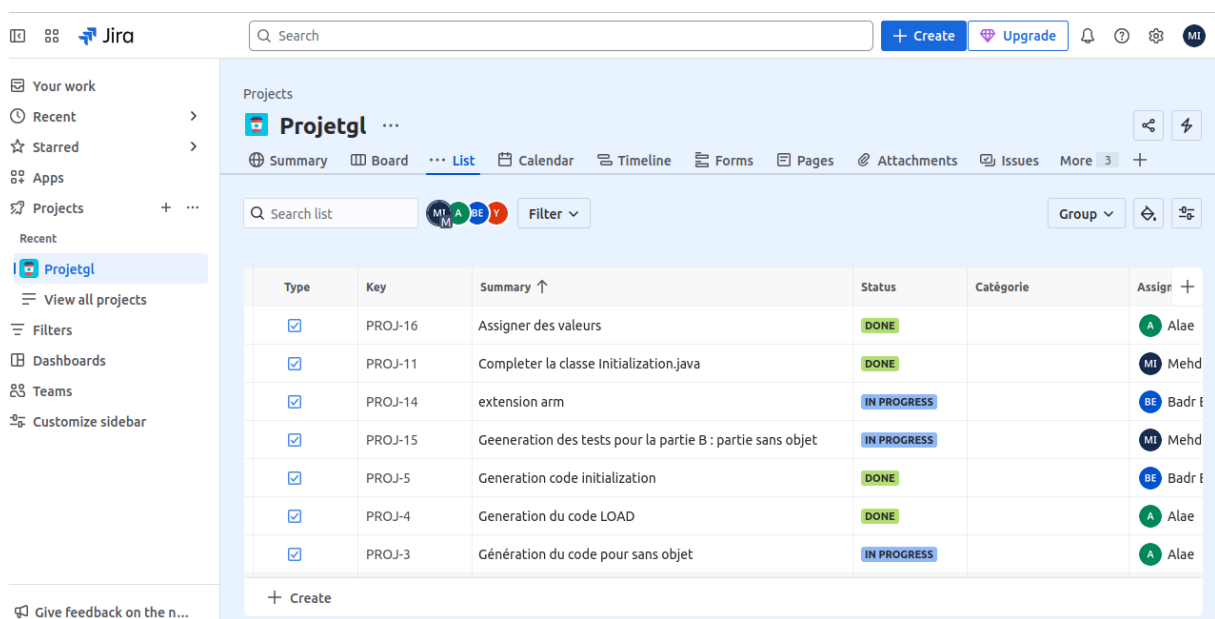
**Répartition des rôles :** Les rôles ont été assignés dès les premiers jours, grâce à la première phase d'identification des forces et faiblesses.

- Yassine a été désigné responsable de la partie A.
- Mehdi et Khadija ont collaboré sur la partie B.
- Alaa et Badr se sont concentrés sur la partie C ainsi que sur l'extension (ARM).

**Communication au sein de l'équipe :**

- **Phase initiale :** Au début du projet, avant le premier suivi, notre communication s'appuyait principalement sur WhatsApp et Discord. Ces outils nous permettaient d'échanger rapidement des informations, de poser des questions, et de partager nos avancées. Nous organisions presque quotidiennement des points de synchronisation pour discuter de l'état d'avancement, identifier les blocages, et répartir les tâches à venir.
- **Évolution après le premier suivi :** Suite au premier suivi, nous avons identifié la nécessité d'une meilleure organisation pour suivre efficacement les tâches en cours, celles à venir, et celles déjà terminées. Nous avons décidé d'adopter Jira, une plateforme de gestion de projet, afin de structurer nos efforts.

- Avec Jira, nous avons pu créer des tableaux spécifiques pour chaque étape du projet, définir des tâches claires, assigner des responsabilités à chaque membre, et fixer des délais pour chaque activité.
- Ce changement a permis une vision plus claire de l'état global du projet, une meilleure anticipation des échéances, et une réduction des oublis ou des doublons dans les tâches.
- **Impact de cette transition** : L'introduction de Jira a transformé notre manière de travailler. La centralisation des informations et des tâches a non seulement amélioré notre organisation, mais également renforcé la collaboration entre les membres de l'équipe. Chacun savait précisément ce qu'il avait à faire, ce qui a contribué à fluidifier les échanges et à renforcer la coordination globale.



Type	Key	Summary ↑	Status	Catégorie	Assign
<input checked="" type="checkbox"/>	PROJ-16	Assigner des valeurs	DONE		Alae
<input checked="" type="checkbox"/>	PROJ-11	Compléter la classe Initialization.java	DONE		Mehd
<input checked="" type="checkbox"/>	PROJ-14	extension arm	IN PROGRESS		Badr t
<input checked="" type="checkbox"/>	PROJ-15	Generation des tests pour la partie B : partie sans objet	IN PROGRESS		Mehd
<input checked="" type="checkbox"/>	PROJ-5	Generation code initialization	DONE		Badr t
<input checked="" type="checkbox"/>	PROJ-4	Generation du code LOAD	DONE		Alae
<input checked="" type="checkbox"/>	PROJ-3	Génération du code pour sans objet	IN PROGRESS		Alae

Figure 1: Capture d'écran de l'utilisation de Jira pour le suivi des tâches

## 1.2 Outils Utilisés

- **Outils de gestion de projet** : GitLab pour le suivi des versions et Jira pour la planification des tâches.
- **Méthode de gestion** : Approche Scrum avec des sprints hebdomadaires.
- **Environnement de développement** : Maven pour la compilation, IntelliJ IDEA et VsCode comme IDE.

## 2 Historique et Déroulement du Projet

Un **planning prévisionnel** a été établi avant le démarrage, avec des jalons clés alignés sur les échéances intermédiaires et finales. Chaque membre était responsable d'une étape spécifique .

## 2.1 Chronologie des étapes

La réalisation du projet s'est déroulée en plusieurs étapes, chacune marquée par des défis spécifiques et une organisation adaptée à nos objectifs. Voici un résumé structuré de cette chronologie :

- **Phase d'analyse :**

- Nous avons commencé par la lecture approfondie du polycopié. Cette étape a été particulièrement exigeante en raison de sa longueur et de la richesse des nouvelles notions qu'il contenait.
- Cette phase d'analyse a nécessité un investissement de temps important pour assimiler les concepts clés du projet et comprendre les spécificités des parties A, B et C.
- Bien que cette étape ait été chronophage, elle a jeté les bases d'une bonne compréhension commune et nous a permis de mieux appréhender la suite du projet.

- **Phase de développement initial :**

- Une fois l'analyse terminée, nous avons entamé le développement du projet. Chacun des membres s'est concentré sur une partie spécifique selon les notions qu'il avait assimilées
- À ce stade, l'accent a été mis sur l'écriture du code, mais sans effectuer de tests systématiques, dans le but de construire une base fonctionnelle pour chaque composant.

- **Détail des parties B et C (à venir) :**

- Nous fournirons une description détaillée des étapes spécifiques liées aux parties B et C dans les sections suivantes, en abordant les difficultés rencontrées, les solutions mises en œuvre, et les leçons apprises.

## 2.2 Déroulé Général du Projet

Le développement du projet a suivi une progression logique, en commençant par les éléments les plus simples pour progressivement aborder les aspects plus complexes :

- **Langage de base :** La première étape a consisté à implémenter un programme *Hello World*. Cela nous a permis de valider l'environnement de développement, de tester les outils de compilation, et de nous familiariser avec les bases du langage Deca, notamment sa syntaxe et sa structure de contrôle.
- **Sans objet :** Les premières fonctionnalités ont été développées . Cette approche a facilité la compréhension des concepts fondamentaux du langage, tels que les expressions, les instructions conditionnelles et les boucles. Elle a également permis d'élaborer les étapes initiales du compilateur, notamment l'analyse lexicale et syntaxique, en se concentrant sur les tokens de base et les règles de grammaire simples.
- **Avec objet :** L'introduction des concepts orientés objet a marqué une étape clé dans le projet. Cela a nécessité d'importants ajustements dans la structure du compilateur pour gérer :

- **Les classes et champs (fields)** : Ajout de la gestion des classes et des attributs avec allocation mémoire et gestion des portées (public, protected).
- **Les méthodes** : Intégration des méthodes avec reconnaissance syntaxique, définition, et gestion des VTables pour les appels dynamiques.
- **L'héritage et le polymorphisme** : Support de l'héritage avec gestion des relations parent-enfant et implémentation du polymorphisme via les VTables.

## 2.3 Étape B : Vérifications Contextuelles

Pour la partie B, la principale difficulté résidait dans la compréhension des attentes et des objectifs à atteindre. De plus, le grand nombre de règles à respecter, combiné à certaines particularités du langage Deca par rapport à Java, a rendu cette étape particulièrement exigeante. Il a fallu consacrer un temps considérable à analyser les spécificités du langage et à adapter nos méthodes de travail pour répondre aux contraintes imposées.

La vérification contextuelle a débuté avec un programme simple *"Hello, World!"*. Bien que ce programme ait été achevé, il a présenté quelques problèmes de tests en raison de l'absence d'implémentation complète de la partie A. Pour contourner cette limitation, nous avons dû créer un programme basé sur un arbre syntaxique et effectuer les tests pour s'assurer de la validité de la partie B, qui était déjà fonctionnelle à ce stade.

Pour surmonter les difficultés rencontrées avec la partie sansobjet, nous avons concentré nos efforts sur l'achèvement de la partie A afin de disposer d'une base de test valide. Cela a également permis de garantir que la partie C puisse avancer. Cependant, l'intégration des programmes avec objet a introduit une complexité supplémentaire pour la partie B. La gestion des relations entre classes, l'héritage et les appels de méthodes a nécessité une refonte partielle de notre code. Enrichir les environnements de typage, tels que *EnvironmentType* et *EnvironmentExp*, s'est révélé crucial pour renforcer la robustesse et la précision des passes de vérification, tout en assurant la cohérence et la fiabilité du système.

Les étapes majeures de l'implémentation de la vérification contextuelle incluaient :

- La création et l'enregistrement des symboles et types dans une table adaptée.
- La validation des types et relations lors de la seconde passe sur l'arbre syntaxique.
- La décoration finale de l'arbre avec les informations de typage complètes pour garantir la conformité avec la grammaire contextuelle.

Grâce à une organisation rigoureuse et une collaboration constante, cette étape, bien qu'exigeante, a été menée à bien.

## 2.4 Étape C : Génération de Code Assembleur

Dans un premier temps, lors de la phase initiale de la partie C, on a procédé à la génération du code sans considérer la gestion des objets. Par la suite, on a abordé la

gestion de la pile dans le cadre d'une situation de manque de registres, en mettant en œuvre des mécanismes adaptés pour optimiser l'allocation et l'utilisation de la pile, garantissant ainsi l'efficacité du processus. Une fois cette étape achevée, on a généré le code des classes de manière indépendante, sans prendre en compte l'héritage, afin d'assurer une première version fonctionnelle de la structure de données.

Dans une étape suivante, on s'est concentré sur la gestion des classes seules, excluant l'utilisation de la directive `extends`, tout en intégrant la gestion du point d'entrée `main`. Cette phase a permis de lier efficacement les différentes classes entre elles tout en assurant la conformité avec la structure du programme. L'ensemble de ces étapes a garanti une gestion optimale des éléments fondamentaux du langage, tout en préparant le terrain pour l'ajout de fonctionnalités avancées telles que l'héritage.

## 2.5 Tests de Validation

La gestion des tests de validation dans notre projet a suivi une chronologie marquée par des ajustements progressifs dans notre méthodologie.

- **Premiers tests** : Comme mentionné précédemment, nous avons débuté avec un test simple pour le programme *"Hello, World!"*. Cependant, les parties B et C ayant progressé plus rapidement que la partie A, nous avons dû créer nos propres tests pour générer un *"Hello, World!"* en sortie, afin de pallier l'absence d'une base de test fournie par la partie A.
- **Tests pour la partie sans objet** : Un incident lié à une mauvaise gestion de *push* a entraîné la suppression d'une grande partie des tests de cette étape, ce qui a eu un impact négatif sur la qualité et le rendu final de cette partie.
- **Méthodologie initiale** : Au départ, nous avons adopté une approche où les tests étaient réalisés uniquement à la fin de chaque implémentation. Cette méthodologie s'est avérée problématique, car elle rendait la détection des erreurs plus difficile et ralentissait les ajustements nécessaires.
- **Changement de méthodologie** : Dans la dernière semaine du projet, nous avons modifié notre approche en intégrant les tests au fur et à mesure de l'implémentation. Cette nouvelle méthodologie a permis de vérifier rapidement la validité de notre code, d'identifier les erreurs plus tôt, et d'enrichir notre base de tests de manière significative. Cela a grandement amélioré la qualité et la fiabilité de notre rendu final.

Ce processus a mis en évidence l'importance d'intégrer les tests de validation dès les premières étapes du développement, une leçon que nous retiendrons pour les projets futurs.

## 2.6 Documentation

Tout au long de notre travail, en plus du développement du code, nous avons également des documents à produire afin de structurer et d'expliquer nos choix. Nous avons adopté une approche progressive pour compléter ces documents au fur et à mesure de l'avancement du projet.

Par exemple, le document de conception a été enrichi au fil de l'implémentation des parties B et C. Il a permis d'expliquer en détail nos algorithmes, nos méthodologies, ainsi que les solutions apportées aux différents problèmes rencontrés. Ce processus d'itération nous a aidés à maintenir une trace claire et précise de nos décisions techniques et à assurer une meilleure cohérence dans l'ensemble du projet.

Cette méthode nous a permis de ne pas attendre la fin du projet pour produire une documentation complète, tout en garantissant que celle-ci reflète fidèlement notre démarche et nos résultats.

## **2.7 Impact des Vacances sur la Cadence de Travail**

Le projet s'est déroulé sur une période d'un mois, incluant deux semaines de vacances. La première semaine de vacances a été prise en compte comme une pause totale, sans avancement significatif. Cela a notablement ralenti la progression du travail, affectant la cadence globale et la capacité de l'équipe à maintenir un rythme soutenu. Pour compenser, des réunions intensives ont été organisées avant et après cette période afin de réaligner les efforts sur les échéances prévues.

## **3 Analyse Critique**

Le projet a révélé à la fois des points forts significatifs et des défis notables. Parmi les éléments positifs, la cohésion de l'équipe a été un atout majeur. Une communication fluide, soutenue par des réunions régulières et des canaux clairs d'échange, a permis de résoudre rapidement les problèmes rencontrés et d'avancer efficacement. Cette dynamique d'équipe a contribué à maintenir un niveau élevé de motivation et à renforcer la collaboration.

Sur le plan technique, le respect strict des conventions de codage a favorisé la lisibilité et la maintenabilité du code. La mise en place d'une base de tests exhaustive a également joué un rôle crucial en assurant la qualité des livrables. Chaque étape du projet a été validée par des tests rigoureux, ce qui a permis de détecter et de corriger les erreurs avant qu'elles ne deviennent critiques.

Cependant, le projet n'a pas été exempt de difficultés. La synchronisation des tâches s'est révélée complexe en raison des nombreuses dépendances entre les étapes. Cela a conduit à des retards dans certaines phases critiques, notamment lors de l'intégration des différentes parties du compilateur.

Pour l'avenir, plusieurs améliorations peuvent être envisagées. Une planification plus flexible, intégrant des marges pour les imprévus, permettrait de mieux gérer les dépendances entre les tâches. De plus, certains membres de l'équipe ont rencontré des problèmes de configuration du projet, en particulier sur des machines Mac, ce qui a occasionné des retards et des difficultés techniques. Une meilleure prise en charge des environnements spécifiques ou une documentation plus complète aurait pu réduire ces obstacles et améliorer la fluidité des échanges.



## **4 Leçons Apprises**

### **4.1 Gestion de Projet**

La gestion de projet a été l'un des principaux domaines d'apprentissage de cette expérience. Nous avons constaté qu'une communication régulière et structurée est essentielle pour maintenir la coordination et éviter les malentendus. Les réunions hebdomadaires et les bilans d'étapes ont permis de détecter rapidement les obstacles et d'ajuster les priorités en conséquence. De plus, nous avons appris que la flexibilité dans le planning est indispensable, car certaines dépendances mal anticipées entre les tâches ont engendré des retards. Enfin, l'utilisation d'outils comme Jira et GitLab a grandement contribué à une gestion efficace des tâches et à une meilleure visibilité sur l'avancement global.

### **4.2 Aspects Techniques**

Sur le plan technique, ce projet nous a permis de perfectionner notre approche de développement incrémental. Tester chaque module séparément avant son intégration a limité les erreurs et amélioré la stabilité du système. Par ailleurs, nous avons renforcé nos compétences dans la conception modulaire du code, ce qui a simplifié les ajustements et amélioré la maintenabilité. Les défis liés à la génération de code, à la gestion des environnements de typage, et à l'intégration des concepts orientés objet nous ont donné une meilleure compréhension des techniques avancées de compilation. En outre, cette expérience a mis en évidence l'importance d'une documentation claire et détaillée pour faciliter la collaboration et garantir la pérennité du projet.

## **5 Conclusion**

Le projet Génie Logiciel a été une expérience enrichissante, tant sur le plan technique qu'humain. L'équipe a su relever les défis liés à la complexité du compilateur et respecter les échéances imposées. Les difficultés rencontrées ont permis de renforcer nos compétences en gestion de projet, en communication et en développement logiciel. Ce bilan servira de base pour améliorer nos futurs travaux collaboratifs.