

RAPPORT DE STAGE DE FIN D'ÉTUDES

Présenté en vue de l'obtention du
Diplôme National de Licence en Science de L'informatique
Spécialité : Computer Science

Par

Asma HACHAICHI et Zeineb HACHAICHI

Création d'un outil de génération automatique des programmes ADL

Encadrants professionnels : **Mr Mohamed BEN BOUZID et Mme Rania SOUAI**

Encadrant académique : **Mr Sahbi ZAHAF**

Réalisé au sein de NeoXam Tunisie



RAPPORT DE STAGE DE FIN D'ÉTUDES

Présenté en vue de l'obtention du
Diplôme National de Licence en Science de L'informatique
Spécialité : Computer Science

Par

Asma HACHAICHI et Zeineb HACHAICHI

**Création d'un outil de génération automatique
des programmes ADL**

Encadrants professionnels : Mr Mohamed BEN BOUZID et Mme Rania SOUAI

Encadrant académique : **Mr Sahbi ZAHAF**

Réalisé au sein de NeoXam Tunisie



J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant professionnel, Mr Mohamed BEN BOUZID et Mme
Rania SOUAI

Signature et cachet



J'autorise l'étudiant à faire le dépôt de son rapport de stage en vue d'une soutenance.

Encadrant académique, Mr Sahbi ZAHAF

Signature

A handwritten signature in blue ink, appearing to read 'Sahbi Zahaf', is shown here.

DÉDICACE

“ Nous dédions ce travail ..

À nos très chers parents Maroua et Yassine, pour tout ce qu'ils ont fait pour nous. Qu'ils trouvent dans ce travail un témoignage de notre profond amour et éternelle reconnaissance. Que dieu vous préserve une longue et heureuse vie.

À notre soeur Khaoula, à qui nous souhaitons une vie pleine de bonheur, de prospérité et de réussite.

À nos proches, qui n'ont cessé de nous soutenir et de nous encourager. Que dieu vous procure bonne santé et longue vie.

À nos amis, pour les bons souvenirs et le beau temps que nous avons passés ensemble, nous ne vous souhaitons que du succès et du bonheur dans vos vies.

Enfin, à la mémoire de toute personne très chère à nos coeurs qui nous a quitté trop tôt pour un monde meilleur.

”

- **Asma et Zeineb Hachaïchi**

REMERCIEMENTS

Nous tenons, avant de présenter notre travail, à exprimer notre grande reconnaissance envers les personnes qui nous ont, de près ou de loin, apporté leurs soutiens. Qu'ils trouvent ici collectivement et individuellement l'expression de toute notre gratitude.

Nous remercions tout d'abord le groupe NeoXam Tunisie pour nous avoir accueilli et donné l'opportunité d'évoluer au sein de leur société.

Nos plus profonds et sincères remerciements vont à :

Notre encadrant académique, Mr. Sahbi ZAHAF pour le temps qu'il a bien voulu consacrer à l'encadrement et le suivi de ce travail, pour ses conseils avisés et la confiance qu'il nous a accordée tout au long de notre stage. Grâce à ses encouragements nous avons pu nous accomplir totalement dans nos missions.

Notre cher professeur, Mr. Moez BEN REGAYA pour ses conseils qu'il nous a prodigué et le partage de son expertise qui nous a beaucoup aidé dans la rédaction de ce rapport.

Nos encadrants professionnels, Mme. Rania SOUAI et Mr. Mohamed BEN BOUZID, qui ont proposé et dirigé ce travail. Nous leur exprimons nos plus profondes reconnaissances pour la confiance qu'ils nous ont accordée et leur aide précieuse, leurs encouragements et leurs conseils continus durant ce stage.

Que les membres de jury trouvent, ici, l'expression de nos remerciements pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail.

- Zeineb et Asma Hachaïchi

TABLE DES MATIÈRES

Introduction générale	1
1 Contexte général	3
Introduction	4
1 Organisme d'accueil	4
1.1 Présentation de la société NeoXam	4
1.2 Historique	4
1.3 Organigramme de NeoXam Tunisie	5
2 Contexte du sujet	5
3 Étude de l'existant	6
3.1 Analyse de l'état actuel	6
3.2 Critique de l'existant	7
3.3 Solutions proposées	7
3.3.1 Création d'un plugin	7
3.3.2 Développement d'une application desktop	8
3.3.3 Développement d'une application web	9
3.4 Solution adoptée	9
4 Choix méthodologique	10
4.1 Processus de développement logiciel	10
4.1.1 Scrum	10
4.1.2 Justification du choix de SCRUM	12
4.2 Langage de modélisation	13
5 Étude technologique	13

TABLE DES MATIÈRES

5.1	Outil de modélisation	13
5.2	Outils de réalisation	14
6	Architecture logique et logicielle	17
	Conclusion	18
2	Planification et capture des besoins	19
	Introduction	20
1	Capture des besoins	20
1.1	Identification des acteurs	20
1.2	Les besoins fonctionnels	20
1.3	Les besoins non fonctionnels	21
1.4	Diagramme de cas d'utilisations global	21
2	Pilotage du projet avec Scrum	22
2.1	Équipe et rôles	22
2.2	Backlog du produit	23
2.3	Planification de la realease	24
	Conclusion	25
3	Sprint1 : Authentification et gestion des utilisateurs et templates	26
	Introduction	27
1	Backlog du Sprint 1	27
2	Diagramme de cas d'utilisation du Sprint 1	28
2.1	Présentation du diagramme de cas d'utilisation	28
2.2	Descriptions textuelles	29
3	Diagrammes de séquences	31
4	Diagramme de classes du sprint1	33
5	Réalisation	34
6	Revue du Sprint1	37
	Conclusion	38
4	Sprint2 : La gestion d'un projet ADL et des composants de la palette	39
	Introduction	40
1	Backlog du sprint 2	40
2	Diagramme de cas d'utilisation du Sprint 2	41

TABLE DES MATIÈRES

2.1	Présentation du diagramme de cas d'utilisation	41
2.2	Descriptions textuelles	42
3	Diagrammes de séquences	46
4	Diagramme de classes du Sprint2	48
5	Modélisation des processus métier	49
6	Réalisation	51
7	Revue du Sprint2	54
	Conclusion	55
5	Sprint3 : Génération du code ADL	56
	Introduction	57
1	Backlog du sprint 3	57
2	Diagramme de cas d'utilisation du Sprint3	58
2.1	Présentation du diagramme de cas d'utilisation	58
2.2	Descriptions textuelles	58
3	Diagrammes de séquences	61
4	Diagramme de classes du Sprint3	62
5	Réalisation	62
6	Revue du sprint3	69
7	Conclusion	70
	Conclusion générale	71
	Références bibliographiques	72

LISTE DES FIGURES

1.1	Locaux de la société NeoXam dans le monde	4
1.2	Organigramme de la société	5
1.3	Affichage de l'écran MAJTIN sur Webintake	6
1.4	Accès à l'éditeur "ADL Editor"	8
1.5	Comparaison entre les solutions	10
1.6	Cycle de vie d'un projet avec Scrum	12
1.7	Schéma récapitulatif des rôles dans Scrum	12
1.8	Logo UML	13
1.9	Visual Paradigm	13
1.10	Logo Citrix Receiver	14
1.11	Logo Trello	14
1.12	Logo Overleaf	14
1.13	Logo Visual Studio Code	15
1.14	Logo Postman	15
1.15	Logo Gitlab	15
1.16	Logo PostgreSQL	16
1.17	Logo Node.JS	16
1.18	Logo React.JS	16
1.19	Logo Express.JS	17
1.20	Architecture logique	17
1.21	Architecture logicielle	18
2.1	Diagramme de cas d'utilisation global	22

LISTE DES FIGURES

2.2	Planification de la release	24
3.1	Diagramme de cas d'utilisation du Sprint1	28
3.2	Diagramme de séquences CU « s'authentifier »	31
3.3	Diagramme de séquences CU « Consulter les utilisateurs »	32
3.4	Diagramme de séquences CU « Modifier utilisateur »	32
3.5	Diagramme de séquences CU « Ajouter utilisateur »	33
3.6	Diagramme de classes du sprint1	33
3.7	Interface d'authentification	34
3.8	Interface bienvenue partie 1	34
3.9	Interface bienvenue partie 2	35
3.10	Interface profil	35
3.11	Pop-up modifier mot de passe	36
3.12	Interface liste utilisateurs	36
3.13	Interface nouvel utilisateur	36
3.14	Pop-up modifier utilisateur	37
3.15	Graphique revue Sprint 1	37
4.1	Diagramme de cas d'utilisation du sprint 2	42
4.2	Diagramme de séquences CU «Consulter un projet ADL»	47
4.3	Diagramme de séquences CU «Créer un nouveau projet ADL»	47
4.4	Diagramme de séquences CU «Supprimer un projet ADL»	48
4.5	Diagramme de séquences CU «Générer squelette ADL»	48
4.6	Diagramme de classes du Sprint2	49
4.7	Diagramme d'activité CU «Ajouter un composant»	50
4.8	Diagramme d'activité CU «Supprimer un composant»	50
4.9	Interface Liste des projets ADL	51
4.10	Pop-up Supprimer projet	51
4.11	Interface Nouveau projet	52
4.12	Interface Liste composants en affichant 4 composants par page	52
4.13	Interface Liste composants en affichant 10 composants par page	53
4.14	Pop-up Ajouter composant	53
4.15	Pop-up Supprimer composant	53
4.16	Interface Modifier composant	54

LISTE DES FIGURES

4.17 Revue du Sprint2	54
5.1 Diagramme de cas d'utilisation du sprint3	58
5.2 Diagramme de séquences CU « Génération du code ADL »	61
5.3 Diagramme de classes du Sprint3	62
5.4 Interface code vide	63
5.5 Interface code avec du code	63
5.6 Schéma de fonctionnement du mode design	64
5.7 Interface design vide	64
5.8 Interface design avec des composants	65
5.9 Pop-up supprimer	65
5.10 Schéma de fonctionnement du générateur ADL	66
5.11 Interface code affichant le code initial	67
5.12 Interface design vide	67
5.13 Interface design	68
5.14 Bouton Valider	68
5.15 Code du fichier généré	68
5.16 Interface code affichant le code après ajout	69
5.17 Interface code affichant le code après ajout	69
5.18 Revue Sprint 3	70

LISTE DES TABLEAUX

2.1	Équipe et rôles	22
2.2	Backlog de produit	23
2.3	La planification des sprints	25
3.1	Backlog Sprint 1	27
3.2	Description textuelle du CU "S'authentifier"	29
3.3	Description textuelle du sous CU "Ajouter un utilisateur"	30
3.4	Description textuelle du sous CU "Modifier un utilisateur"	30
4.1	Backlog Sprint 2	40
4.2	Description textuelle du CU "Créer un nouveau projet ADL"	42
4.3	Description textuelle du CU "Supprimer un projet ADL"	43
4.4	Description textuelle du CU "Dupliquer un composant de la palette"	44
4.5	Description textuelle du CU "Modifier un composant dans la palette"	45
5.1	Backlog Sprint3	57
5.2	Description textuelle du CU "Ajouter un composant dans l'écran"	59
5.3	Description textuelle du CU "Supprimer un composant de l'écran"	59
5.4	Description textuelle du CU "Générer le code ADL"	60

LISTE DES ABRÉVIATIONS

- **ADL** = Application Development Language
- **FS** = File System
- **GP** = Global Portfolio
- **JS** = JavaScript
- **MVC** = Model View Controller
- **UML** = Unified Modeling Language
- **UX** = User eXperience

INTRODUCTION GÉNÉRALE

Au cours de ces dernières années, un changement fondamental est survenu dans la stratégie des entreprises dans lesquelles plusieurs activités se basent sur l'utilisation d'applications web et mobiles.

En effet, un logiciel qui est limité à la réalisation des besoins fonctionnels de ses utilisateurs n'est plus satisfaisant. D'autres exigences sont alors introduites comme la performance, la maintenabilité et l'ergonomie des applications.

Dans ce contexte s'inscrit l'activité de la société « NeoXam » qui conçoit des logiciels pour les sociétés de gestion d'actifs financiers. Dans ce domaine, NeoXam a développé, en utilisant son propre langage de programmation ADL, plusieurs solutions de gestion d'actifs tel que GP, qui est un progiciel configurable composé d'un ensemble d'interfaces appelées écrans.

Malgré la disponibilité de plusieurs logiciels de construction d'interfaces graphiques et de génération automatique de code, l'équipe de « NeoXam » a besoin de s'appuyer sur un outil propre à elle permettant la génération automatique du code en langage ADL à partir d'une interface contenant des composants GP.

C'est dans ce cadre que se situe notre sujet de projet de fin d'études qui vise à concevoir et développer une application web contenant des composants GP et leurs codes en langage ADL adéquat. Cette application va générer automatiquement le code ADL de ces composants dans les nouveaux écrans GP à travers une interface graphique simple en utilisation.

L'objectif de ce sujet est d'augmenter la fiabilité, l'efficacité et faciliter les tâches complexes lors de l'écriture du code pour chaque écran GP.

Le présent rapport est composé de cinq chapitres dont :

- Le premier chapitre *Contexte général* concerne la présentation de l'organisme d'accueil, du cadre du projet, de la problématique ainsi que le diagnostic de l'existant et de la méthodo-

logie adoptée.

- Le deuxième chapitre *Analyse et spécification des besoins* présente les acteurs du système, les besoins fonctionnels et non fonctionnels, le diagramme de cas d'utilisation global et la planification des sprints.
- Le troisième chapitre est dédié au premier sprint intitulé *Authentification et gestion des utilisateurs et des templates*.
- Le quatrième chapitre est dédié au deuxième sprint qui décrit *La gestion d'un projet ADL et des composants de la palette*.
- Le dernier chapitre est dédié au dernier sprint qui est consacré à *L'automatisation du code en langage ADL*.

Enfin nous terminons ce rapport par une conclusion générale qui résume le travail réalisé tout au long de ce projet ainsi que les perspectives envisagées.

CHAPITRE 1

CONTEXTE GÉNÉRAL

Introduction	4
1 Organisme d'accueil	4
1.1 Présentation de la société NeoXam	4
1.2 Historique	4
1.3 Organigramme de NeoXam Tunisie	5
2 Contexte du sujet	5
3 Étude de l'existant	6
3.1 Analyse de l'état actuel	6
3.2 Critique de l'existant	7
3.3 Solutions proposées	7
3.4 Solution adoptée	9
4 Choix méthodologique	10
4.1 Processus de développement logiciel	10
4.2 Langage de modélisation	13
5 Étude technologique	13
5.1 Outil de modélisation	13
5.2 Outils de réalisation	14
6 Architecture logique et logicielle	17
Conclusion	18

Introduction

Dans ce premier chapitre, nous abordons la mise en contexte de notre sujet. Nous commençons par présenter l'organisme d'accueil, NeoXam Tunisie. Ensuite, nous exposons l'étude et critique de l'existant qui nous conduisent à la solution proposée. Nous concluons ce chapitre par l'aspect organisationnel en spécifiant la méthodologie de travail adoptée tout au long du sujet.

1 Organisme d'accueil

1.1 Présentation de la société NeoXam

Ce travail a été réalisé au sein de la société internationale NeoXam dont le siège social est basé en France, et fondé en Tunisie en 2014. NeoXam est un éditeur de progiciels financiers pour les investisseurs et les émetteurs dans le marché financier. Cette dernière est présente dans 14 pays : France, Francfort, Luxembourg, Zurich, Genève, Milan, New York, Boston, HongKong, Shangai, Pékin, Singapour, Tunis et au Cap.

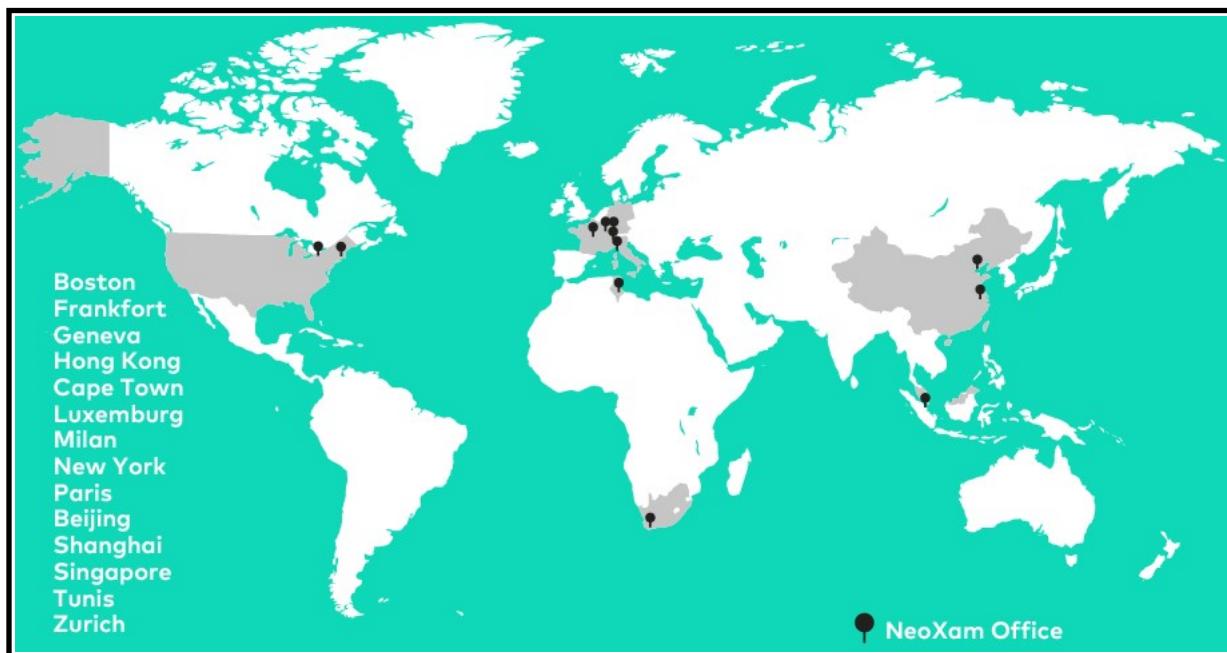


FIGURE 1.1 – Locaux de la société NeoXam dans le monde

1.2 Historique

- 31 Janvier 2014 : Fondation de NeoXam par BlackFin CP et Serge Delpha, PDG NeoXam.
- Octobre 2014 : NeoXam acquiert les solutions « GP3 » et « Decalog » de SunGard.

- Octobre 2014 : NeoXam rachète « Density Tech ».
- Avril 2015 : NeoXam rachète « Nexfi ».
- Juin 2015 : NeoXam rachète « SmartCo ».
- 2016 : NeoXam devient leader des solutions logicielles financières en Europe.
- 2018 : « Cathay Capital » et « Bpifrance » accompagnent NeoXam dans l'accélération de son développement international.
- 2019 : NeoXam achète 100M et renforce ses ores de gestion des données, ses capacités de reporting et son expérience utilisateur.

1.3 Organigramme de NeoXam Tunisie

La figure 1.2 schématise l'organigramme de la société NeoXam Tunisie.

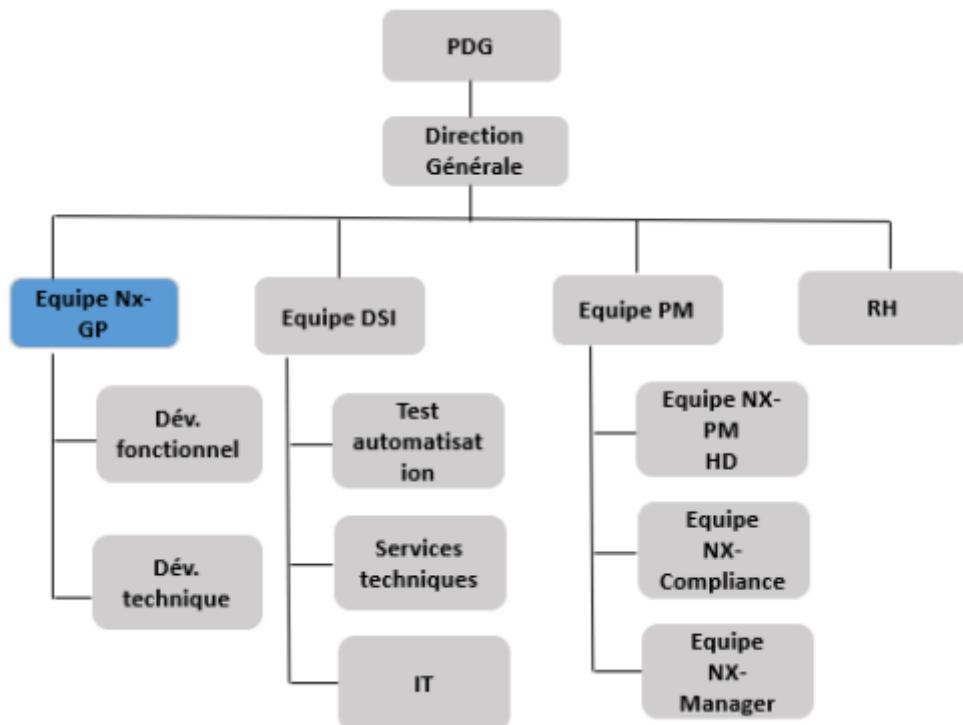


FIGURE 1.2 – Organigramme de la société

2 Contexte du sujet

GP est un progiciel de gestion des actifs financiers utilisé dans les bourses, c'est un ensemble de programmes hautement paramétrables développés en langage spécifique nommé ADL. Afin d'améliorer l'interface du GP qui ressemblait à une console noire et blanc, fut créer le Webintake.

Il s'agit d'une IHM développée en Java qui propose une interface graphique déployable par internet pour rendre les écrans GP plus conviviales.

Webintake représente donc la couche présentation de GP.

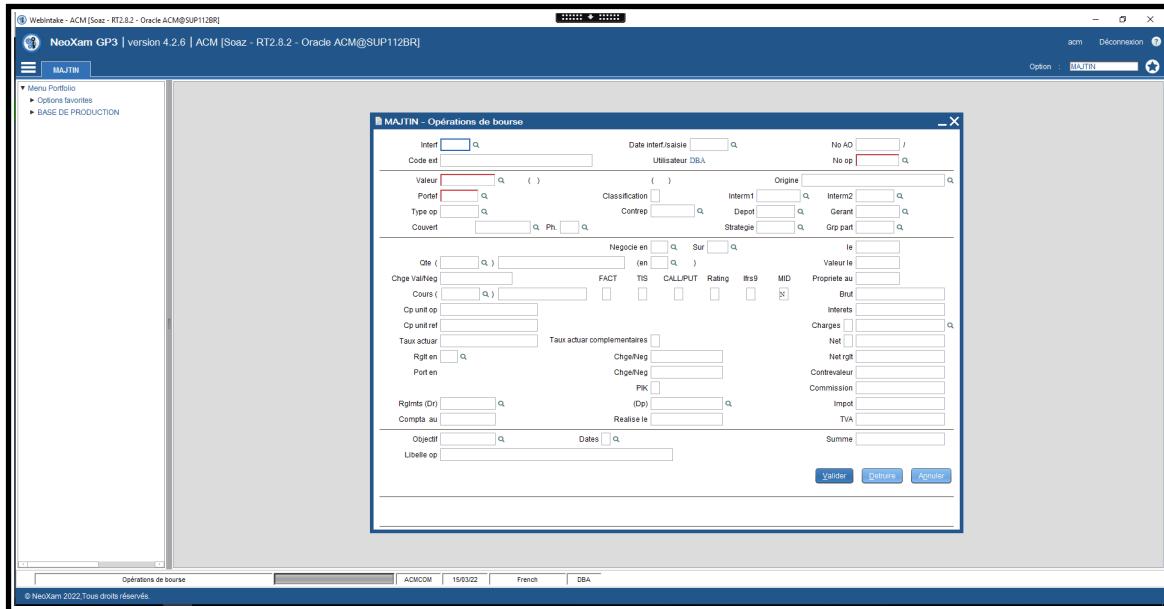


FIGURE 1.3 – Affichage de l'écran MAJTIN sur Webintake

GP3 est opérationnel dans le monde depuis plus de 30 ans et emploie plus de 70 consultants seniors et experts.

GP est composé d'un ensemble d'écrans - chacun représente une procédure - permettant la création d'un portefeuille, création d'une comptabilité, effectuation d'une transaction d'achat ou de vente, la valorisation des portefeuilles, génération des rapports, ...

Chaque nouvelle version de GP contient de nouveaux écrans ou enrichit les écrans déjà existants en y ajoutant de nouvelles fonctionnalités, champs ou options.

3 Étude de l'existant

Après avoir présenté la partie sur laquelle on va travailler : les écrans GP, nous expliquons la méthode existante du développement des écrans GP et les problèmes que nous allons résoudre dans notre sujet.

3.1 Analyse de l'état actuel

Actuellement, pour ajouter ou modifier un écran, les développeurs de NeoXam doivent écrire le code ADL de zéro manuellement, malgré le fait que les squelettes des écrans sont toutes les

mêmes.

Nous avons constaté alors que la création et la modification des nouveaux écrans pour la solution « GP » contiennent plusieurs étapes redondantes qui prennent beaucoup de temps lors de la phase de développement. En effet, il y a plusieurs similarités entre les différents écrans et leurs composants, ce qui rend une grande partie du code réécrite à chaque fois qu'on veut ajouter un écran.

3.2 Critique de l'existant

Dans cette partie, nous discutons les différents inconvénients observés par les équipes de NeoXam lors du processus de modification ou création des écrans GP.

- *Perte de temps* : Les écrans GP possèdent des squelettes presque identiques, ainsi les développeurs vont prendre beaucoup de temps pour réécrire un code qui est déjà existant.
- *Hétérogénéité des codes* : Chaque développeur a sa propre façon de développement. Par conséquent, les codes des écrans ne se ressembleront pas.
- *Problèmes de maintenance* : Avec des codes hétérogènes, les prochains développeurs à faire des modifications dans des écrans existants pourront trouver des difficultés dans la compréhension du code initial.
- *Sources d'erreurs* : La réécriture des codes fait augmenter les sources d'erreurs potentielles.

3.3 Solutions proposées

Suite à l'étude des problèmes évoqués, nous proposons trois solutions. Dans cette partie nous détaillons chaque solution afin de déterminer la meilleure solution à adopter.

3.3.1 Création d'un plugin

Cette solution consiste au développement d'un plugin, en Java, qui doit être intégrable avec Eclipse ADL.

Ce plugin va permettre d'utiliser un concepteur visuel, muni d'une palette contenant des composants, pour concevoir les interfaces et le code ADL sera automatiquement généré. Ce concepteur graphique contient une zone de conception et une palette dans laquelle se trouvent les composants des écrans GP.

Accéssibilité : Pour ouvrir l'éditeur fourni par ce plugin, il suffit de faire un clic droit sur le fichier, et choisir "Ouvrir avec > ADL Editor".

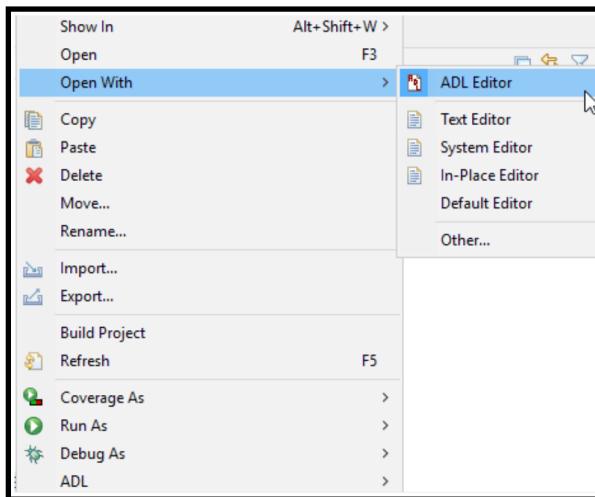


FIGURE 1.4 – Accès à l'éditeur "ADL Editor"

Inconvénients et difficultés :

Bien que cette solution soit complète et permet de couvrir les problèmes cités dans la section du critique de l'existant, elle est difficile à mettre en oeuvre vu sa complexité sur plusieurs niveaux :

- *Séparation artificielle* : Si un auteur de plugin trouve qu'il a besoin d'un plugin qui peut raisonnablement faire deux choses en tandem serré, il devrait implémenter deux plugins et trouver les moyens d'assurer la communication entre eux.
- *Difficultés à maintenir une bonne performance dans la communication entre les plugins* : Bien que chaque plugin fonctionne lorsqu'il est testé seul, les interactions plug-in peuvent entraîner de nouveaux problèmes, avec des bugs apparaissant uniquement avec certaines combinaisons de plug-ins.
- *Maintenabilité* : Le fournisseur du cadre de plugin non seulement doit s'assurer que l'interface de plugin satisfait les cas d'utilisation dentelés, est clair et bien documenté, mais aussi qu'elle peut évoluer.
- *Contrainte de temps* : Cette solution mettra plus de quatre mois à être mise en œuvre.

3.3.2 Développement d'une application desktop

Création d'une application desktop de génération automatique des écrans GP en générant le code ADL des interfaces tout en injectant toutes les bibliothèques et ressources nécessaires.

Cette application doit contenir un éditeur de code dans lequel s'affiche le code du projet actuel et une zone de conception dans laquelle le développeur peut concevoir son écran sans écriture de code.

Inconvénients et difficultés :

Bien que cette solution permet de couvrir les problèmes cités dans la section du critique de l'existant, elle ne permettra pas l'utilisation de l'automatisation dans la production de code pour la raison suivante :

- *Outils d'automatisation* : Manque d'outils d'automatisations pour les applications desktop.
- *Communauté limitée* : Les outils d'automatisations disponibles pour les applications desktop ne sont pas largement utilisés ni maintenus (Exemples : WinAppDriver, Winium, ..).

3.3.3 Développement d'une application web

Dans le but de réduire le temps de développement des écrans et diminuer le nombre d'erreurs dans le code ADL, nous avons décidé de créer une application web de génération automatique des écrans GP en générant le code ADL des interfaces tout en injectant toutes les bibliothèques et ressources nécessaires.

Pour créer un nouvel écran, le développeur n'a qu'à cliquer sur un bouton d'ajout d'écran, et le code de l'écran sera généré automatiquement.

Ce dernier peut également ajouter des composants dans son écran et leur code sera inséré dans le code de l'écran principal.

Cet outil va apporter plein d'avantages aux développeurs sur plusieurs niveaux :

- *Gain de temps* : Il peut enlever beaucoup de temps au niveau des tâches de codage.
- Diminution des sources d'erreurs, vu que le code de base sera automatiquement généré.
- *Homogénéité du code* : Les codes auront une apparence presque identique, ce qui facilitera leur manipulation par les développeurs.
- *Garantie du respect des normes de développement de NeoXam* : Contrairement à un programme automatique, l'être humain peut parfois commettre des erreurs ou oublier les normes et bonnes pratiques de développement de NeoXam.

3.4 Solution adoptée

Après avoir effectuer des recherches de faisabilités pour chacune des solutions présentées précédemment, nous avons décidé d'implémenter la troisième solution *Développement d'une application web*, vu qu'elle est complète, couvre tous les besoins cités, et elle est la solution qui contient le nombre minimal d'inconvénients.

La figure 1.5 ci-dessous présente une comparaison entre les trois solutions proposées :

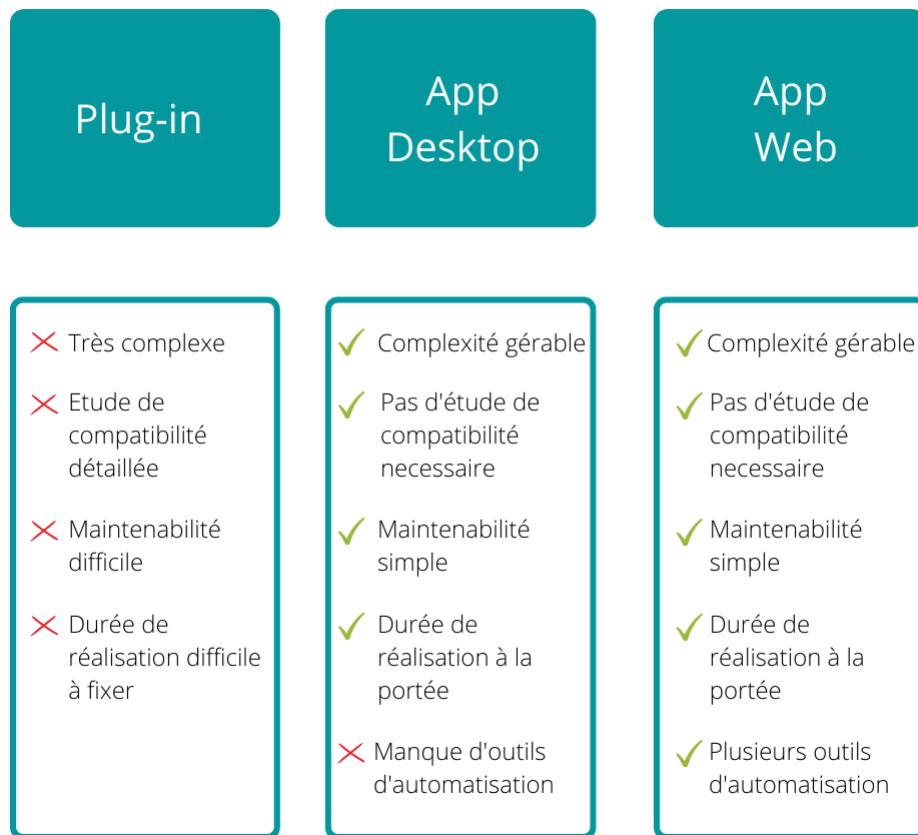


FIGURE 1.5 – Comparaison entre les solutions

4 Choix méthodologique

4.1 Processus de développement logiciel

Une méthodologie est un cadre utilisé pour la planification d'un projet afin d'obtenir un produit final de bonne qualité. Pour notre sujet, le choix de méthode de travail est la méthodologie SCRUM. Ce choix est basé sur des raisons liées à l'instabilité de l'environnement technologique et au fait que le client est souvent dans l'incapacité de définir ses besoins d'une manière exhaustive dès le début du projet.

4.1.1 Scrum

La méthode Scrum est une méthode agile dont le nom est un terme emprunté au rugby qui signifie " la melée ". Elle s'appuie sur le découpage des projets en itérations encore nommées **sprints**.

- **Principes de SCRUM** La méthodologie **Scrum** se caractérise par un ensemble d'étapes pour réussir un projet :
 - *Définir les besoins du client* : Le procès débitera par la formalisation de la vision du client à propos le projet à développer et met à la même longueur d'onde tous les intervenants dans la réalisation du projet.
 - *Créer le product backlog* : Dans un deuxième lieu, nous allons énumérer toutes les tâches nécessaires à la réalisation du produit dans un document appelé **product backlog**. Ce document est modifiable dans n'importe quelle phase de développement du produit.
 - *Définir la durée d'une itération (Sprint)* : Une fois que nous avons spécifié les besoins et listé les tâches fondamentales, on estime le temps nécessaire pour finir ce projet. En fonction du temps estimé et la liste des tâches, nous décidons sur la durée du **Sprint**.
 - *Planification de sprint* : À chaque départ de sprint, un lot de tâches listées dans le **product backlog** sera affecté à l'équipe pour le réaliser pendant cette itération.
 - *Exécution de sprint* : Après la planification du sprint, les tâches seront affectées aux membres de l'équipe. Chaque membre va changer le statut de sa tâche à "En cours" et une fois qu'il la termine, le statut devient "Terminé", puis, il peut choisir une autre tâche avec le statut "À faire" jusqu'à la fin des tâches.
 - *Réunion quotidienne* : Le but de ces réunions est de connaître l'état d'avancement quotidien de l'équipe. Cette réunion dure entre 15 et 30 minutes et se fait soit à la fin de journée soit le matin au début de la journée.
 - *Revue de sprint* : À la fin de chaque sprint, une réunion se fait durant laquelle toute l'équipe fait un bilan sur le travail réalisé et voit les possibilités et les méthodes adéquates qu'ils peuvent adopter pour améliorer le rendement.

La figure 1.6 schématise le cycle de vie d'un projet avec Scrum.

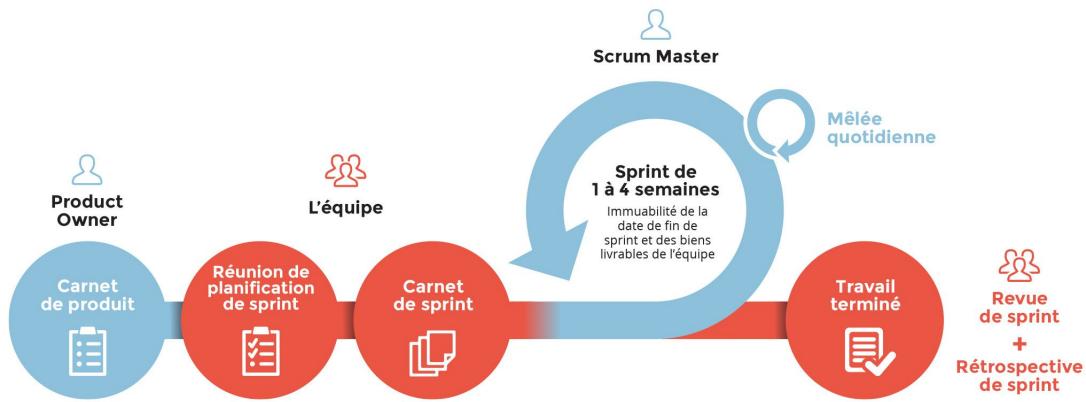


FIGURE 1.6 – Cycle de vie d'un projet avec Scrum

- **Équipes et rôles** Une équipe Scrum est composée d'un Product Owner, d'une équipe de développement et d'un Scrum Master
 - *Product owner* : Responsable de la gestion du backlog produit. Il explique et ordonne le Backlog.
 - *Equipe de développement* : Responsable de la création de l'incrément à la fin du projet. L'équipe se caractérise par l'auto-organisation et la plurodiscipline.
 - *Scrum master* : Le leader-serviteur de l'équipe Scrum. Il partage la prise de décision et a une approche globale sur le projet.

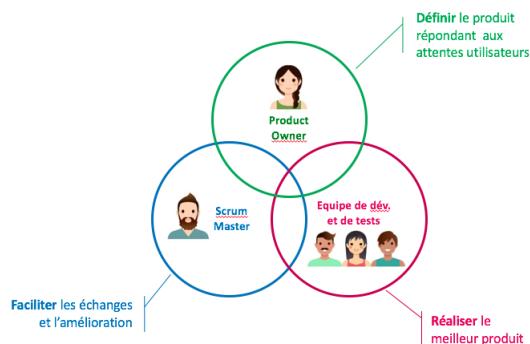


FIGURE 1.7 – Schéma récapitulatif des rôles dans Scrum

4.1.2 Justification du choix de SCRUM

Le choix de Scrum comme méthodologie de pilotage pour notre projet s'est basé sur les atouts de ce dernier. Ils se résument comme suit :

- Une architecture souple laissant place à la créativité. En plaçant le cahier des charges en arrière plan, Scrum évite de frustrer les membres de l'équipe avec un plan et des tâches précises. Être agile, c'est être souple et laisser grandir le projet au fil des sprints.

- Une grande capacité d'adaptation au changement grâce à des itérations courtes.
- Un gain de temps de développement.

4.2 Langage de modélisation

Vu que Scrum se base trop sur la documentation, nous avons besoin d'un langage de modélisation afin de mieux schématiser notre travail.

Dans notre cas, ce langage est UML.



FIGURE 1.8 – Logo UML

L'UML est un langage graphique de modélisation informatique. Ce langage est désormais la référence en modélisation objet, ou programmation orientée objet.

L'UML est constitué de diagrammes qui servent à visualiser et décrire la structure et le comportement des objets qui se trouvent dans un système. Il permet de présenter des systèmes logiciels complexes de manière plus simple et compréhensible qu'avec du code informatique.

L'UML a des applications dans le développement logiciel, mais aussi dans l'industrie (pour modéliser les flux de processus par exemple), dans l'ingénierie ou le marketing. [1]

5 Étude technologique

5.1 Outil de modélisation

- Visual Paradigm



FIGURE 1.9 – Visual Paradigm

Visual Paradigm est un outil utilisé pour créer et gérer l'UML ou les cas de langage de modélisation unifié, qui prend également en charge le groupe de gestion d'objets, y compris la notation de modélisation des processus métier. [2]

5.2 Outils de réalisation

- **Citrix Receiver**



FIGURE 1.10 – Logo Citrix Receiver

Citrix Receiver est un programme qui opère comme une hôte d'applications en ligne. Elle nous permet de travailler à distance sans avoir à installer aucune application sur nos ordinateurs [3].

- **Trello**



FIGURE 1.11 – Logo Trello

Trello est un outil de collaboration qui organise les projets en tableaux. En un coup d'œil, Trello indique l'environnement de travail. C'est comme un tableau blanc, rempli de listes de notes autocollantes, chaque note étant une tâche pour l'utilisateur et toute l'équipe [4]

- **Overleaf**



FIGURE 1.12 – Logo Overleaf

Overleaf est un éditeur en ligne qui permet de stocker, éditer, gérer et compiler des projets Latex en ligne, où chaque projet peut être partagé entre plusieurs utilisateurs, en temps réel [5].

- **Visual Studio Code**



FIGURE 1.13 – Logo Visual Studio Code

Visual Studio Code est un éditeur de code multiplateforme, open source et gratuit développé par Microsoft. Il supporte plusieurs langages de programmation tout en donnant la possibilité de télécharger les langages non inclus par défaut [6]

- **Postman**



FIGURE 1.14 – Logo Postman

Postman est une plateforme collaborative de développement des API. Il permet la création, le test et la modification des API. Nous avons ainsi testé les services Rest implémentés dans le BackEnd par cet outil [7]

- **Gitlab**



FIGURE 1.15 – Logo Gitlab

GitLab est une plateforme open source de collaboration et de gestion de versions destinée aux développeurs de logiciels. Le développement de notre application étant réalisé sur deux ordinateurs, donc nous avons bénéficié de la fonctionnalité de gestion de versions de GitLab [8].

- **PostgreSQL**



FIGURE 1.16 – Logo PostgreSQL

PostgreSQL un système de gestion de base de données relationnelle orienté objet puissant et open source qui est capable de prendre en charge en toute sécurité les charges de travail de données les plus complexes. Ce SGBD a été fondé par une communauté de développeurs et d'entreprises [9].

- **Node.JS**



FIGURE 1.17 – Logo Node.JS

Node.js est un environnement d'exécution JavaScript open source et multiplateforme. Cet outil convient à de nombreux types de projets. Node.js exécute le moteur JavaScript V8 en dehors du navigateur, qui est au cœur de Google Chrome. Cela permet à Node.js de bien fonctionner [10].

- **React.JS**



FIGURE 1.18 – Logo React.JS

React est une bibliothèque JavaScript open source et libre. L'objectif principal de cette bibliothèque est de rendre la création d'applications Web d'une seule page plus facile en créant des composants liés à l'état et en générant une page HTML à chaque changement d'état [11].

- Express.JS



FIGURE 1.19 – Logo Express.JS

Express.js est un framework d'application Web back-end pour Node.js, publié en tant que logiciel gratuit et open-source sous la licence MIT, conçu pour créer des applications Web et des API [12].

6 Architecture logique et logicielle

- **Architecture logique**

L'architecture logique adoptée dans notre projet est de type 3-tiers, composée des trois couches suivantes :

- **Couche Client** : Cette couche contient les interfaces permettant à l'utilisateur d'interagir avec l'application.
- **Couche Serveur** : Cette couche représente le serveur de l'application qui héberge toutes les couches de l'application à développer.
- **Couche Base de données** : Cette couche représente la base de données de notre site qui sera développée avec PostgreSQL.



FIGURE 1.20 – Architecture logique

- **Architecture logicielle**

L'organisation du code source est assurée par le biais du modèle MVC. C'est une solution reconnue permettant de séparer l'affichage des informations (Views), les actions de l'utilisateur (Controllers) et l'accès aux données (Models). Son rôle est donc de segmenter la logique du code en trois parties :

- **Modèle :** Représente le cœur de l'application qui permet de récupérer des données brutes stockées dans notre base de données.
- **Vue :** C'est la partie qui s'occupe des interactions avec l'utilisateur.
- **Contrôleur :** Permet de gérer le dynamique de l'application en analysant les requêtes et fait ainsi le lien entre l'utilisateur et le reste de l'application.

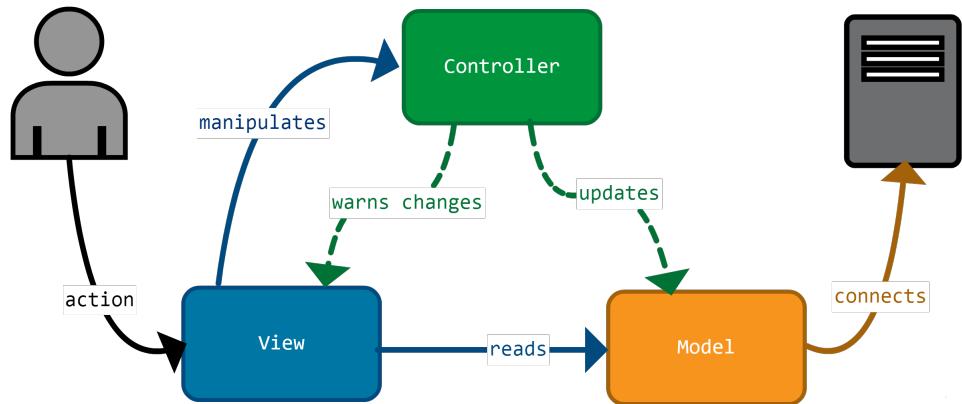


FIGURE 1.21 – Architecture logicielle

Conclusion

Dans ce premier chapitre, nous avons mis notre projet dans son contexte afin de déterminer les étapes nécessaires pour sa conception et sa réalisation. Dans le deuxième chapitre, nous présentons l'analyse et spécification des besoins fonctionnels et non fonctionnels, ainsi que le diagramme des cas d'utilisation.

CHAPITRE 2

PLANIFICATION ET CAPTURE DES BESOINS

Introduction	20
1 Capture des besoins	20
1.1 Identification des acteurs	20
1.2 Les besoins fonctionnels	20
1.3 Les besoins non fonctionnels	21
1.4 Diagramme de cas d'utilisations global	21
2 Pilotage du projet avec Scrum	22
2.1 Équipe et rôles	22
2.2 Backlog du produit	23
2.3 Planification de la realease	24
Conclusion	25

Introduction

Dans ce deuxième chapitre, nous démarrons la première phase de la méthodologie SCRUM nommée *Sprint 0*.

Le but de ce chapitre est de premièrement donner une idée approfondie sur notre projet, identifier ses différents acteurs, besoins fonctionnels et non fonctionnels.

Nous élaborons ensuite le backlog, réalisons le diagramme de cas d'utilisation et présentons la planification des sprints.

Cette phase a une importance capitale pour le bon déroulement du projet car elle nous permet de mieux comprendre les demandes et les attentes de l'entreprise.

1 Capture des besoins

1.1 Identification des acteurs

Un acteur est une entité externe qui interagit directement avec le système.

Les différents acteurs de notre application sont :

- **Le développeur :** C'est l'acteur qui peut créer des écrans ADL et les manipuler tel qu'il souhaite. Il a une autorité limitée sur certaines fonctionnalités de l'application.
- **L'administrateur :** C'est un utilisateur autorisé à utiliser les opérations offertes par l'application pour le développeur, en plus de la possibilité de gérer les comptes et les permissions des autres utilisateurs.

1.2 Les besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités du système. Ce sont les nécessités spécifiant un comportement d'entrée/sortie du système.

En effet, le système doit permettre à l'utilisateur de :

- **Gérer les projets ADL :** Les développeurs ont la possibilité de créer, modifier ou supprimer un projet ADL.
- **Saisir du code ADL :** Les développeurs peuvent éditer le code ADL dans la zone de texte de la partie source.
- **Fournir une partie graphique :** Cette fonctionnalité offre aux développeurs l'option de concevoir des écrans GP dans la zone de conception de la partie graphique.

- **Fournir une palette :** Les développeurs peuvent ajouter des composants dans leurs écrans à partir de la palette.
- **Gérer les composants de la palette :** Les développeurs peuvent ajouter de nouveaux composants dans la palette de composants.
- **Générer du code ADL :** Chaque édition de la partie graphique va modifier le code ADL correspondant selon les changements apportés.
- **Gérer les templates d'écrans GP :** Les administrateurs peuvent créer, modifier ou supprimer des templates d'écrans GP.
- **Gérer les utilisateurs :** Les administrateurs peuvent ajouter, modifier ou supprimer des utilisateurs.

1.3 Les besoins non fonctionnels

Un besoin non fonctionnel est un besoins spécifiant des propriétés du système, telles que les contraintes liées à l'environnement et à l'implémentation, et les exigences en matière de performances, de dépendances de plate-forme, de facilité de maintenance, d'extensibilité et de fiabilité.

Dans notre cadre de travail, l'application doit répondre aux contraintes suivantes :

- **La performance :** L'application doit être performante sur le plan de réponse de système suite à une action dans un délai précis.
- **L'extensibilité :** L'application doit être extensible, ce qui indique qu'il y a une possibilité d'ajouter ou de modifier quelques fonctionnalités.
- **La sécurité :** C'est le critère le plus important pour une application, l'accès doit être limité aux personnes physiques identifiées par un nom d'utilisation et un mot de passe.
- **Convivialité :** Les interfaces de l'application doivent être faciles à utiliser et ergonomiques pour optimiser l'UX.

1.4 Diagramme de cas d'utilisations global

Avant tout développement, il convient de répondre à la question *A quoi va servir le nouveau système ?*, nous établissons le diagramme de cas d'Utilisation 2.1 pour répondre à cette question.

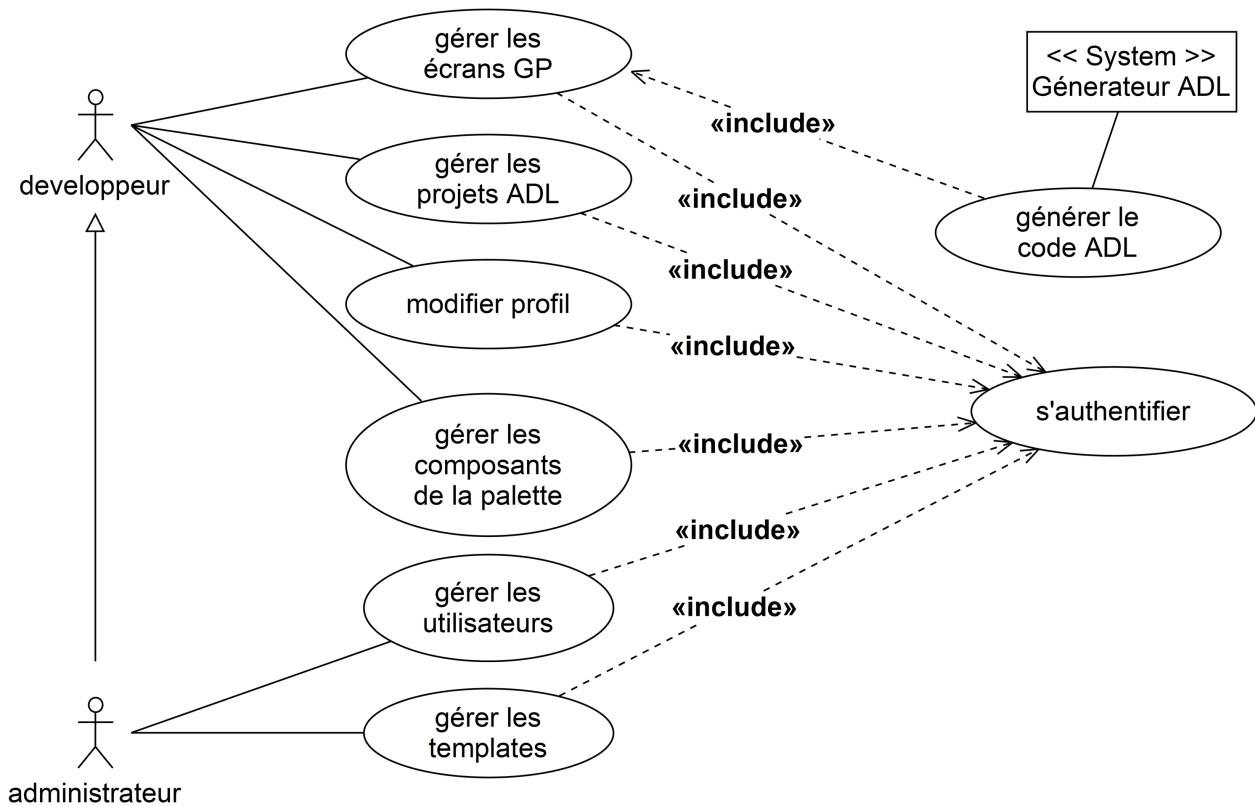


FIGURE 2.1 – Diagramme de cas d'utilisation global

2 Pilotage du projet avec Scrum

2.1 Équipe et rôles

Dans les projets SCRUM, l'équipe joue un rôle important dans l'optimisation de la productivité et l'assurance de l'avancement du travail.

TABLE 2.1 – Équipe et rôles

Rôle	Personne	Tâches demandées
Product Owner	Mr Mohamed BEN BOUZID	<ul style="list-style-type: none"> → Fixer la vision et les exigences du produit; → Fixer les priorités
Scrum Master	Mme Rania SOUAI	<ul style="list-style-type: none"> → Maintenir le product backlog à jour; → Guider l'équipe de développement
Équipe	Zeineb HACHAICHI Asma HACHAICHI	<ul style="list-style-type: none"> → Estimer la charge des users stories; → Définir les tâches techniques; → Développer le produit

2.2 Backlog du produit

Comme le montre le tableau 2.2, le backlog Scrum est conçu pour capturer toutes les exigences des clients auxquelles nous devons répondre. Par conséquent, il contient une liste de fonctions qui nécessitent notre intervention.

Tous les besoins inclus dans le backlog Scrum sont classés par priorité (haute, moyenne et faible) indiquant l'ordre de leur réalisation.

Cet ordonnancement est basé sur la valeur ajoutée que la fonction apporte au produit.

TABLE 2.2 – Backlog de produit

ID	En tant que	Je voudrais (User Story)	Priorité
1	Développeur	M'authentifier	élevée
2	Développeur	Gérer mon profil	moyenne
3	Développeur	Créer un nouveau projet ADL	élevée
4	Développeur	Gérer les projets ADL	moyenne
5	Développeur	Consulter l'écran de mon projet ADL	élevée
6	Développeur	Consulter les composants de la palette	élevée
7	Développeur	Gérer les composants de la palette	moyenne
8	Développeur	Ajouter des composants dans mon écran à partir de la palette	élevée
9	Développeur	Supprimer des composants de mon écran	moyenne
10	Développeur	Modifier l'ordre des composants dans mon écran	faible
11	Développeur	Gérer le code ADL correspondant à mon projet via un éditeur de code	élevée
12	Développeur	Générer automatiquement le code de mon écran	élevée
13	Développeur	Sauvegarder le code de mon projet	élevée
14	Administrateur	Gérer la liste des développeurs	faible
Page suivante			

TABLE 2.2 – Backlog de produit

ID	En tant que	Je voudrais (User Story)	Priorité
15	Administrateur	Gérer la liste des administrateurs	moyenne
16	Administrateur	Gérer les projets des développeurs	moyenne
17	Administrateur	Gérer les templates	moyenne

2.3 Planification de la realease

La planification de la release permet d'orchestrer les sprints en fonction des priorités définies dans le backlog ainsi que la capacité de travail de l'équipe.

La figure 2.2 présente une séquence de sprints à venir, avec une vision du contenu prévu (les éléments de backlog de produit) de ces sprints

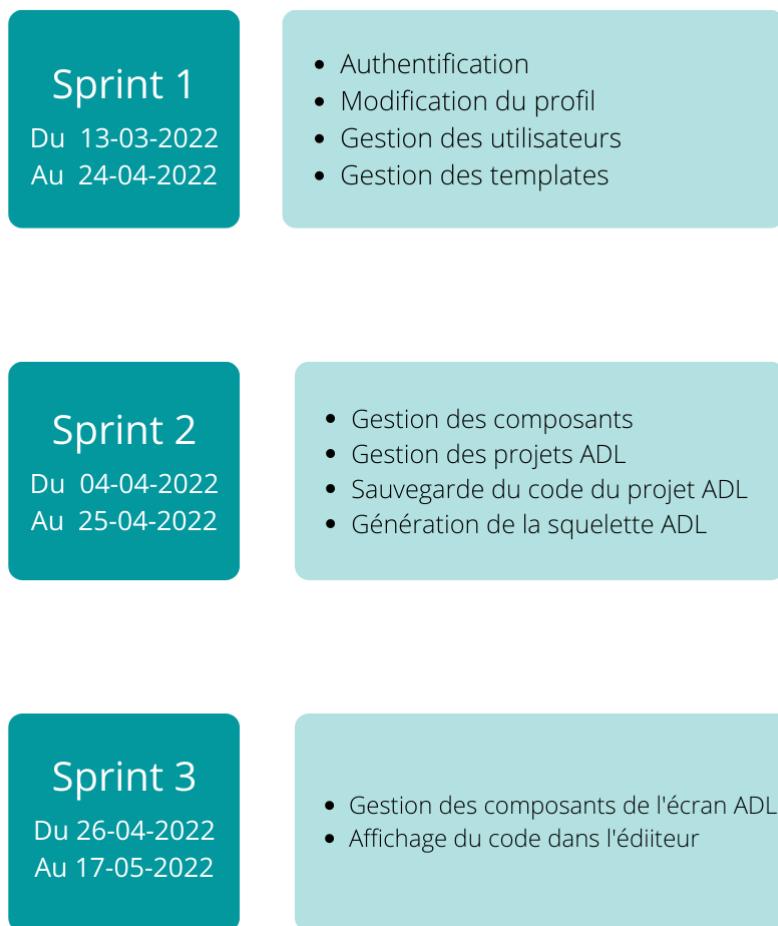


FIGURE 2.2 – Planification de la release

Le tableau 2.3 ci-dessous présente la planification de la release :

TABLE 2.3 – La planification des sprints

Sprint	Objectif	ID user story
1	Authentification et gestion des utilisateurs et templates	1, 2, 14, 15, 16, 17
2	Gestion d'un projet ADL et des composants de la palette	3, 4, 5, 6, 7, 11, 13
3	Gestion des écrans GP et génération du code ADL	8, 9, 10, 11

Conclusion

Dans ce chapitre, nous avons dégagé les besoins fonctionnels et non fonctionnels, ce qui nous a conduit à l'élaboration du diagramme de cas d'utilisation. Ensuite, nous avons présenté notre *product backlog* qui nous a permis de planifier les sprints à la fin.

CHAPITRE 3

SPRINT1 : AUTHENTIFICATION ET GESTION DES UTILISATEURS ET TEMPLATES

Introduction	27
1 Backlog du Sprint 1	27
2 Diagramme de cas d'utilisation du Sprint 1	28
2.1 Présentation du diagramme de cas d'utilisation	28
2.2 Descriptions textuelles	29
3 Diagrammes de séquences	31
4 Diagramme de classes du sprint1	33
5 Réalisation	34
6 Revue du Sprint1	37
Conclusion	38

Introduction

Pour l'élaboration du premier sprint, nous commençons par la planification du sprint suivi par la schématisation des différents diagrammes relatifs à cette partie et des captures représentant les interfaces graphiques le travail effectué durant le Sprint 1.

1 Backlog du Sprint 1

Nous estimons un travail quotidien de six heures par jour pendant trois semaines pour la réalisation de ce sprint.

TABLE 3.1 – Backlog Sprint 1

User story	ID	Tâches	Estimation(/h)
En tant qu'administrateur, je peux gérer les utilisateurs.	1.1	Partie backend : Création des model, controller et route de la table «Développeur».	12
	1.2	Partie frontend : Ajouter les interfaces de la partie administrateur.	12
	1.3	Tester le fonctionnement de la partie frontend et backend.	9
En tant que développeur ou administrateur, je peux m'authentifier.	2.1	Partie frontend : Ajouter l'interface d'authentification.	3
	2.2	Intégrer et tester le fonctionnement de la partie frontend et backend.	6
En tant que développeur ou administrateur je peux modifier mon profil.	3.1	Ajouter une interface permettant la modification des informations de l'utilisateur.	7
			Page suivante

TABLE 3.1 – Backlog Sprint 1

User story	ID	Tâches	Estimation(/h)
	3.2	Intégration frontend et backend pour l'affichage des données et l'enregistrement des modifications.	13
En tant qu'administrateur, je peux gérer les templates des écrans GP.	4.1	Étude sur les projets ADL afin d'extraire les différents templates	18
	4.2	Partie backend : Création des model, controller et route de la table «Template».	10

2 Diagramme de cas d'utilisation du Sprint 1

2.1 Présentation du diagramme de cas d'utilisation

La figure 3.1 présente le diagramme de cas d'utilisation relatif au sprint 1.

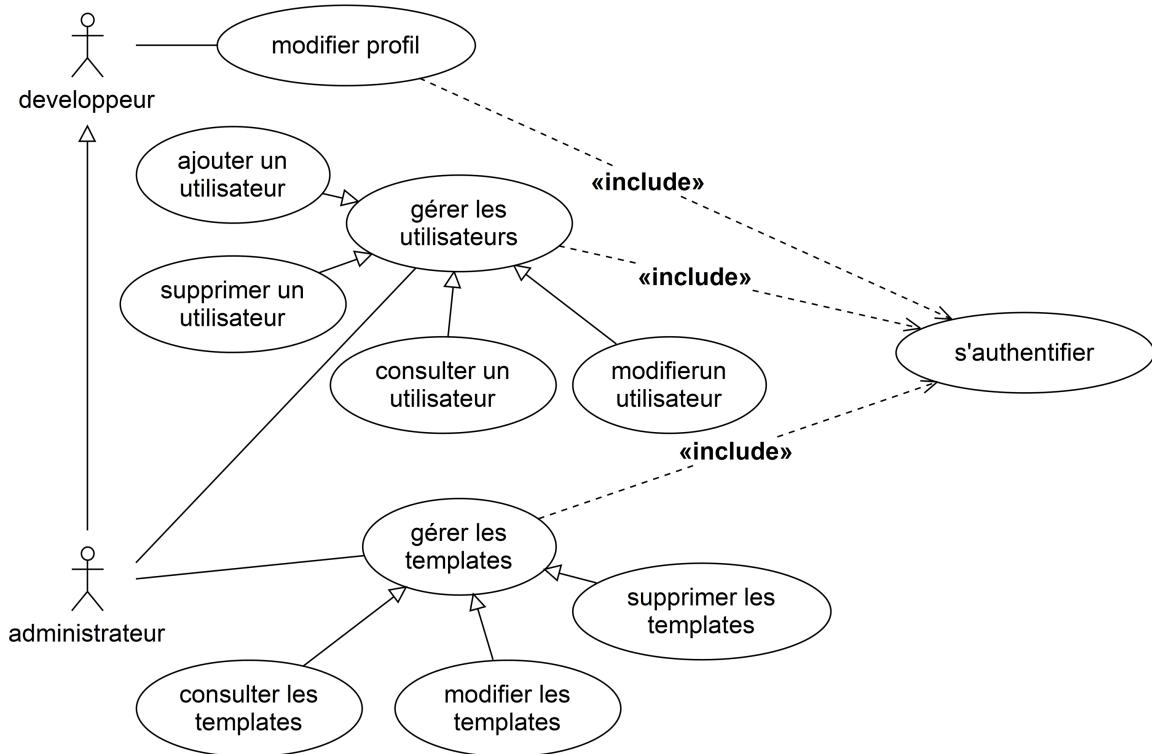


FIGURE 3.1 – Diagramme de cas d'utilisation du Sprint1

2.2 Descriptions textuelles

Dans cette partie, nous analysons les cas et sous cas d'utilisations représentés dans le diagramme du Sprint 1.

- **Description textuelle du CU "S'authentifier"**

TABLE 3.2 – Description textuelle du CU "S'authentifier"

Cas d'utilisation	S'authentifier
Acteurs	Développeur/Administrateur
Pré-condition	L'utilisateur doit être inscrit dans la base de données et doit connaître ses identifiants
Post-condition	L'utilisateur accède à la page de bienvenue
Scénario nominal	<p>❶ Le système affiche le formulaire de connexion.</p> <p>❷ L'utilisateur saisit ses identifiants</p> <p>❸ L'utilisateur valide le formulaire</p> <p>❹ Le système vérifie les informations saisies</p> <p>❺ Le système affiche la page de bienvenue</p>
Scénario alternatif	<p>Les données saisies sont incomplètes :</p> <p>❻ le système affiche un message d'erreur</p> <p>❼ le cas d'utilisation reprend à la deuxième étape du Scénario nominal</p> <p>Les données saisies sont invalides :</p> <p>❾ le système affiche un message d'erreur</p> <p>❿ le cas d'utilisation reprend à la deuxième étape du Scénario nominal</p>

Chaque utilisateur (Développeur/Administrateur) doit tout d'abord s'authentifier pour accéder à l'application. Pour s'authentifier, il faut saisir un login et mot de passe qui seront testés par le système et devront être enregistré dans la base des données. Si les identifiants saisis sont correctes, l'utilisateur sera redirigé par le système vers l'interface suivante qui est la page de bienvenue. Dans le cas contraire, un message d'erreur sera affiché.

- **Description textuelle du sous CU "Ajouter un utilisateur"**

TABLE 3.3 – Description textuelle du sous CU "Ajouter un utilisateur"

Cas d'utilisation	Ajouter un utilisateur
Acteur	Administrateur
Résumé	L'administrateur peut ajouter un nouvel utilisateur.
Pré-condition	Administrateur authentifié.
Scénario principal	<p>Pour créer un nouveau projet ADL :</p> <ul style="list-style-type: none"> ❶ L'administrateur accède à la liste des utilisateurs. ❷ L'administrateur clique sur le bouton «Ajouter utilisateur». ❸ L'administrateur remplit les champs et clique sur «Ajouter». ❹ Le système ajoute l'utilisateur dans la base de données et affiche un message de succès.
Post-condition	Utilisateur ajouté.

Pour ajouter un utilisateur (Développeur/Administrateur), l'administrateur doit tout d'abord s'authentifier. Ensuite, il accède à la liste des utilisateurs et clique sur "Ajouter" pour que le système lui affiche un formulaire à remplir. Une fois le formulaire rempli, le système vérifie le format des champs saisis et insère une nouvelle ligne dans la table des utilisateurs.

- **Description textuelle du sous CU "Modifier un utilisateur"**

TABLE 3.4 – Description textuelle du sous CU "Modifier un utilisateur"

Cas d'utilisation	Modifier un utilisateur
Acteur	Administrateur
Résumé	L'administrateur peut modifier les données d'un utilisateur.
Page suivante	

TABLE 3.4 – Description textuelle sous du CU "Modifier utilisateur"

Cas d'utilisation	Modifier utilisateur
Pré-condition	Administrateur authentifié.
Scénario principal	<p>Pour créer un nouveau projet ADL :</p> <ul style="list-style-type: none"> ❶ L'administrateur accède à la liste des utilisateurs. ❷ L'administrateur sélectionne l'icône «Modifier» de l'utilisateur souhaité. ❸ L'administrateur modifie les champs et clique sur «Valider». ❹ Le système enregistre les modifications et affiche un message de succès.
Post-condition	Utilisateur modifié.

L'administrateur peut modifier les informations personnelles des développeurs. Il suffit de choisir un développeur de la liste et cliquer sur "Modifier utilisateur". Les données saisies s'enregistrent dans la base de données après validation.

3 Diagrammes de séquences

Dans cette partie, nous présentons les diagrammes de séquences détaillés des cas d'utilisation du Sprint 1.

- Diagramme de séquences CU « s'authentifier »

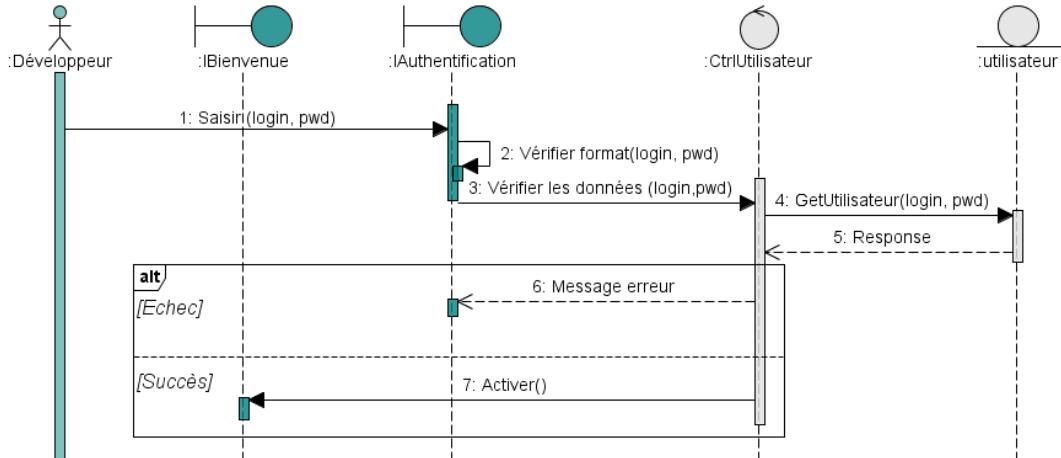


FIGURE 3.2 – Diagramme de séquences CU « s'authentifier »

- Diagramme de séquences CU « Consulter utilisateur »

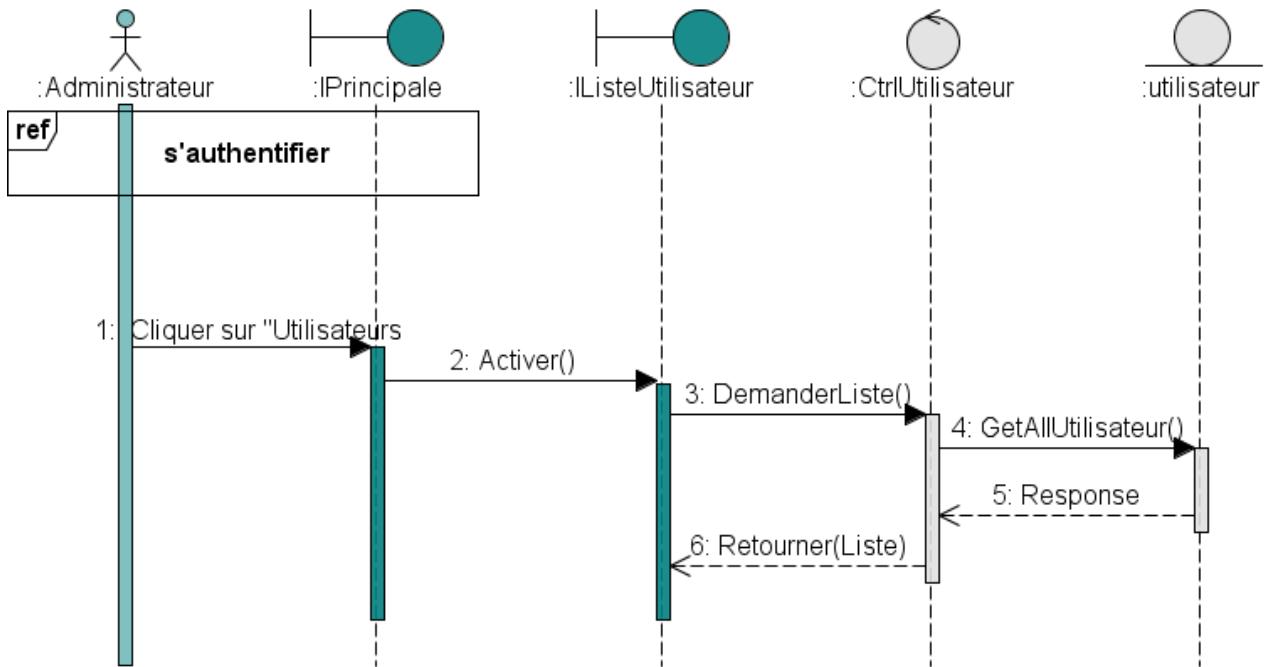


FIGURE 3.3 – Diagramme de séquences CU « Consulter les utilisateurs »

- Diagramme de séquences CU « Modifier utilisateur »

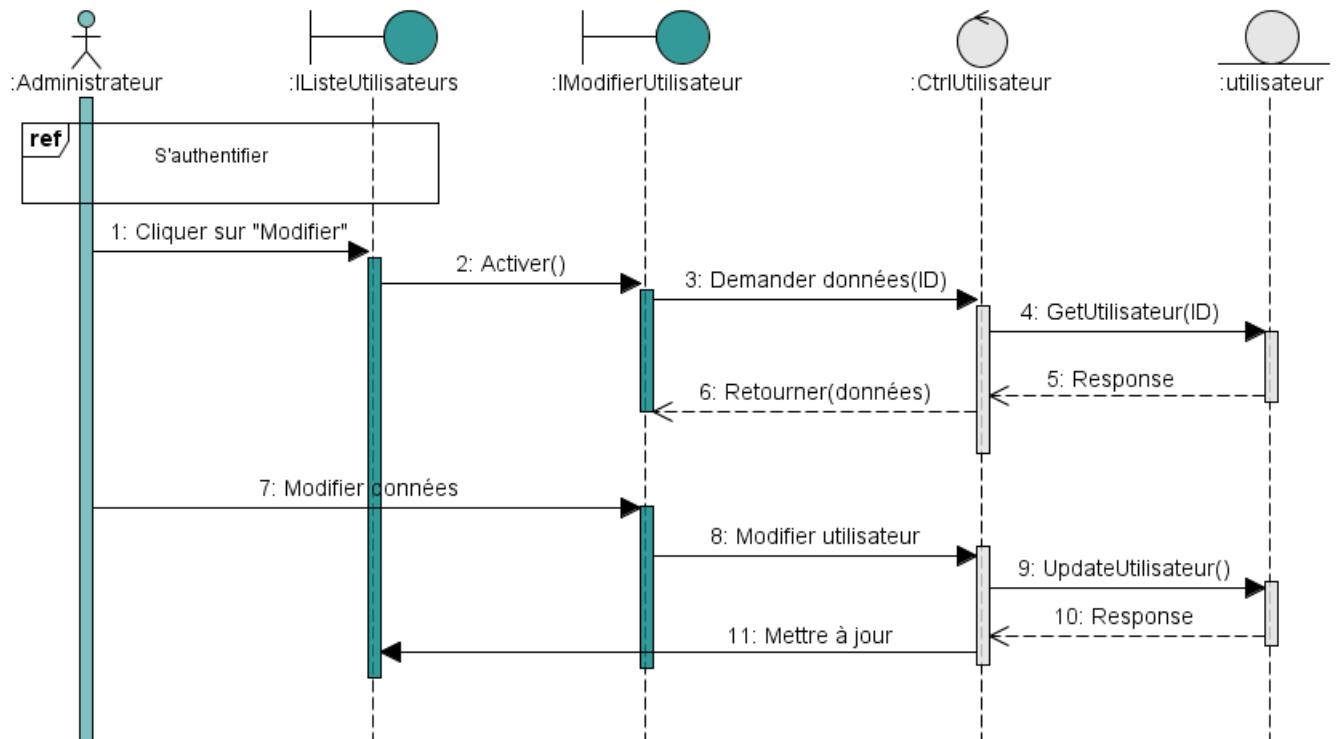


FIGURE 3.4 – Diagramme de séquences CU « Modifier utilisateur »

- Diagramme de séquences CU « Ajouter utilisateur »

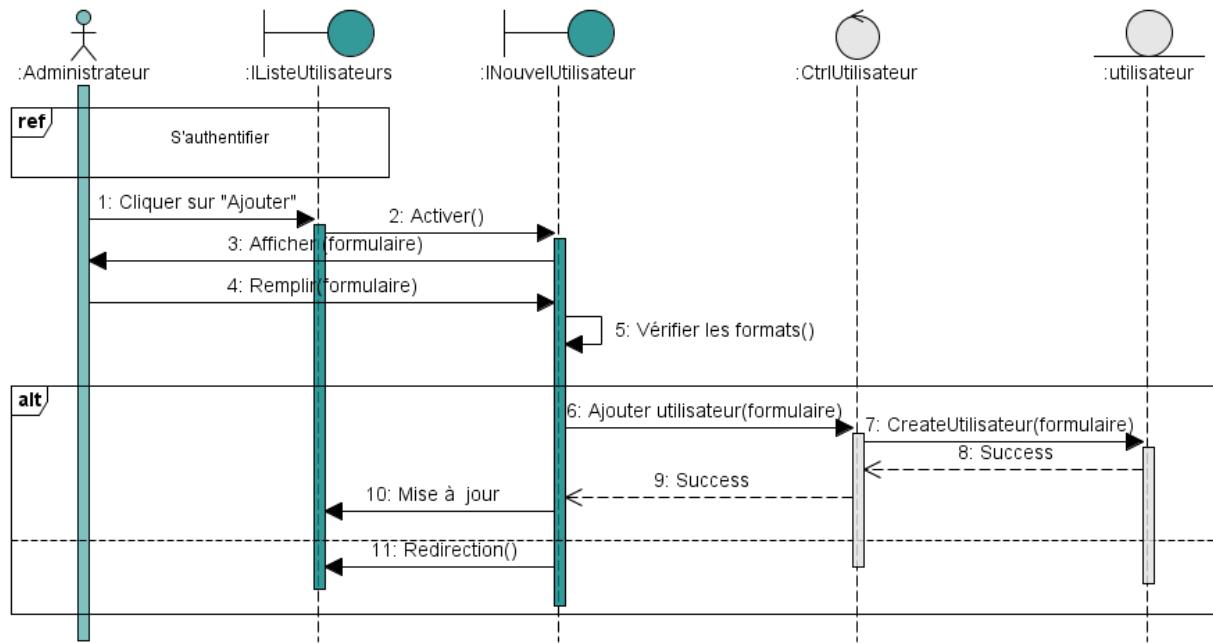


FIGURE 3.5 – Diagramme de séquences CU « Ajouter utilisateur »

4 Diagramme de classes du sprint1

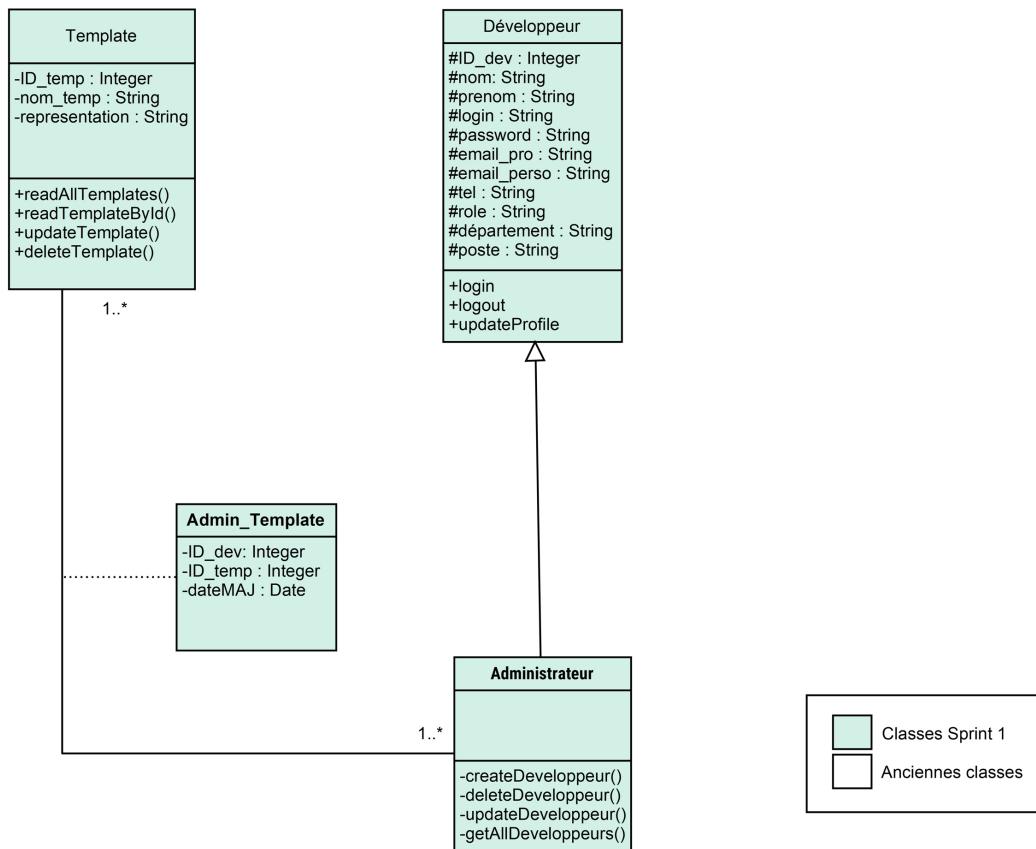


FIGURE 3.6 – Diagramme de classes du sprint1

5 Réalisation

- Interface d'authentification

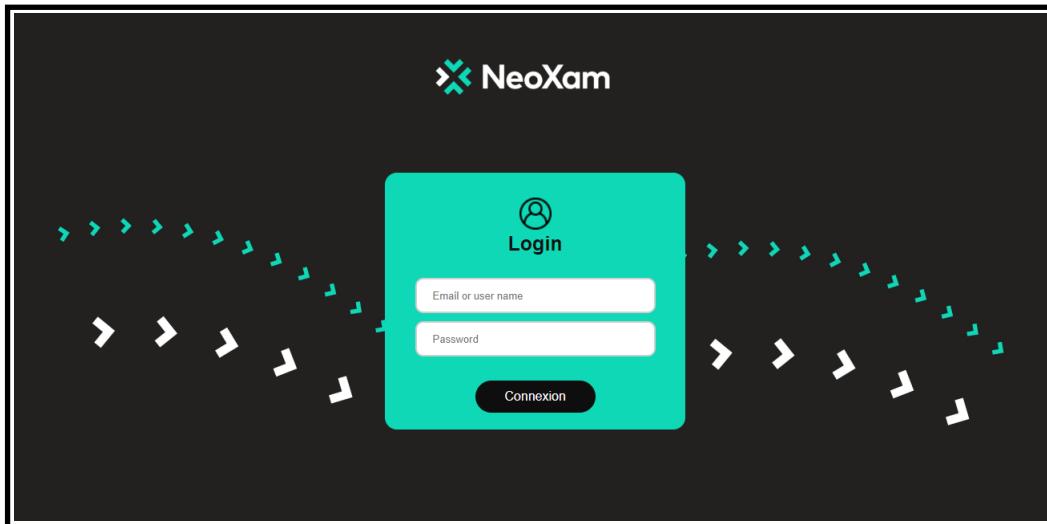


FIGURE 3.7 – Interface d'authentification

L'utilisateur commence par s'authentifier en entrant ses données dans les champs de l'interface présentée dans la figure 3.7 ci-dessus.

- Interface de bienvenue

Une fois l'authentification réussie, l'utilisateur sera redirigé vers la page de bienvenue.

A partir de cette page, il peut commencer le développement en choisissant l'option *Démarrer*, ou bien il peut consulter les informations relatives aux produits de NeoXam : GP3 et GP4.



FIGURE 3.8 – Interface bienvenue partie 1



FIGURE 3.9 – Interface bienvenue partie 2

- **Interface profil**

L'utilisateur peut accéder à l'interface profil présenté dans la figure 3.10 ci-dessous, où il peut voir ses informations personnelles.

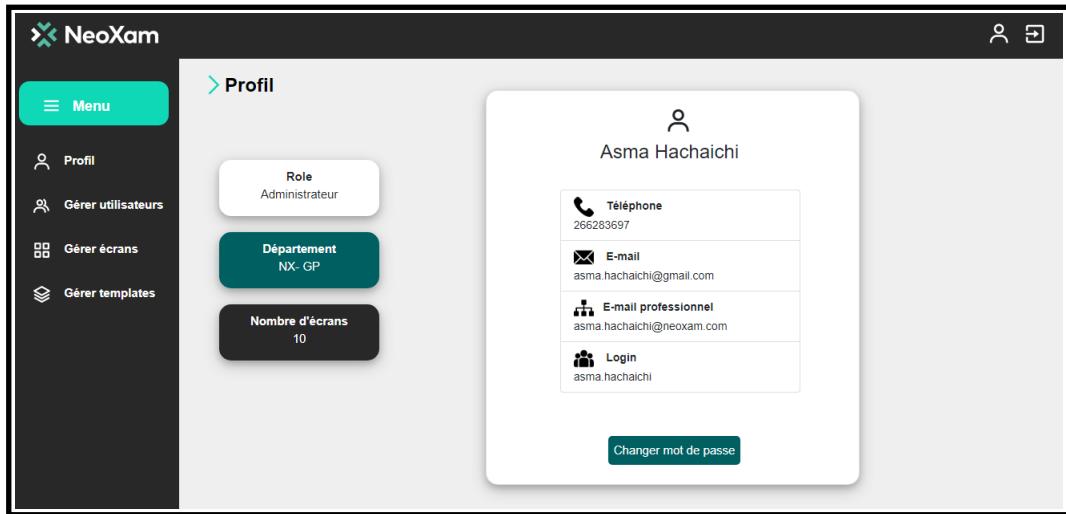


FIGURE 3.10 – Interface profil

- **Pop-up modifier mot de passe**

Après avoir accéder au profil de l'utilisateur, ce dernier peut modifier son mot de passe en cliquant sur "Changer mot de passe". Suite à cette action, la pop-up suivante s'affiche :



FIGURE 3.11 – Pop-up modifier mot de passe

- **Interface liste utilisateurs**

L'interface liste utilisateurs est accessible aux administrateurs. Elle leurs permet de consulter la liste de tous les utilisateurs(Développeurs/Administrateurs).

FIGURE 3.12 – Interface liste utilisateurs

- **Interface nouvel utilisateur**

Seul l'administrateur peut ajouter un nouvel utilisateur à partir de l'interface présenté dans la figure 3.13 :

FIGURE 3.13 – Interface nouvel utilisateur

- **Pop-up modifier utilisateur**

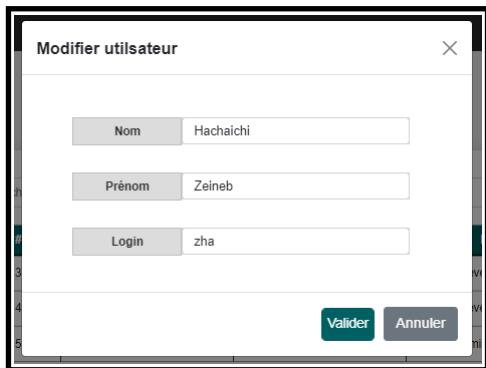


FIGURE 3.14 – Pop-up modifier utilisateur

6 Revue du Sprint1

Le revue de sprint est représenté par un graphique d'évolution permet de visualiser l'état d'avancement réel du projet par rapport à l'état d'avancement estimé au début du sprint.

Il comporte deux axes :

- Nombre de jours
- Nombre d'heure travaillé

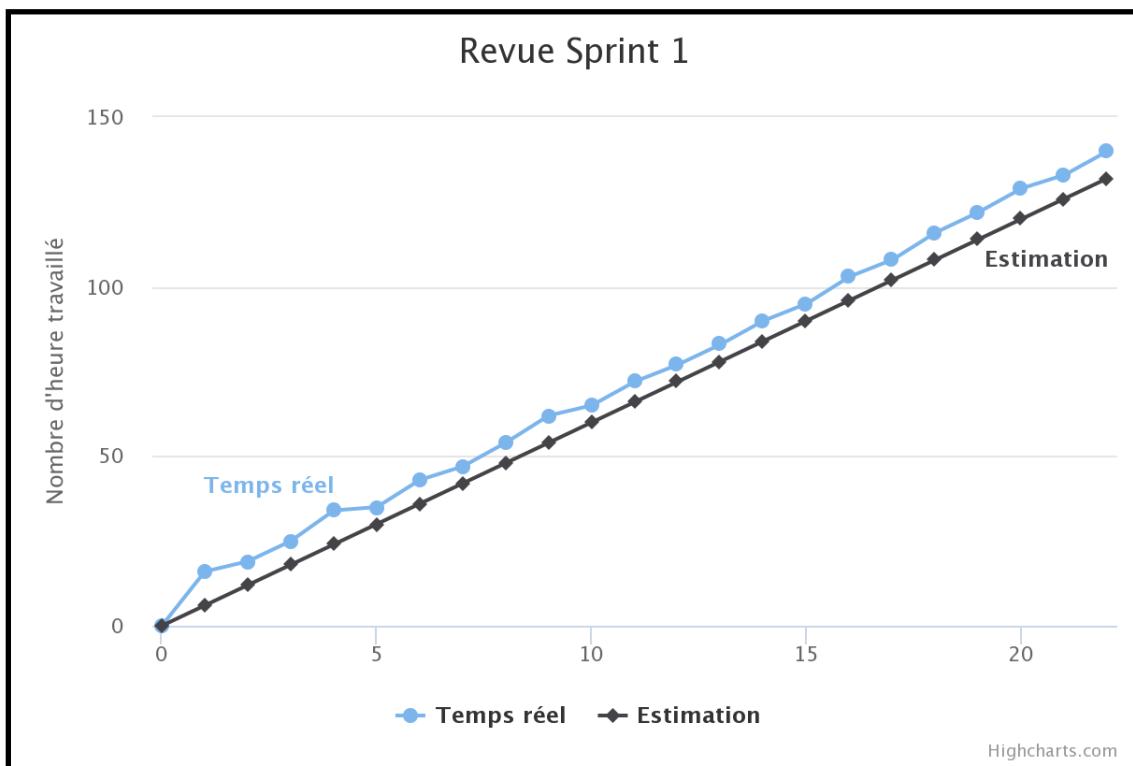


FIGURE 3.15 – Graphique revue Sprint 1

D'après le graphe 3.15 ci-dessus, la courbe du temps réel (représentée en bleue) est au dessus de la droite idéale (représentée en noir).

C'est assez normal puisque durant cette période, nous prenons connaissance de nouvelles technologies.

Conclusion

Au cours de ce chapitre, nous avons réalisé le premier Sprint intitulé « Authentification et gestion des utilisateurs et des templates » en élaborant sa conception. Dans le chapitre suivant, nous entamerons le développement du Sprint 2.

CHAPITRE 4

SPRINT2 : LA GESTION D'UN PROJET ADL ET DES COMPOSANTS DE LA PALETTE

Introduction	40
1 Backlog du sprint 2	40
2 Diagramme de cas d'utilisation du Sprint 2	41
2.1 Présentation du diagramme de cas d'utilisation	41
2.2 Descriptions textuelles	42
3 Diagrammes de séquences	46
4 Diagramme de classes du Sprint2	48
5 Modélisation des processus métier	49
6 Réalisation	51
7 Revue du Sprint2	54
Conclusion	55

Introduction

Ce chapitre est dédié à la réalisation et la présentation du sprint 2, intitulé «Gestion d'un projet ADL et des composants de la palette». Dans cette partie nous présenterons l'analyse et la spécification des besoins pour ce sprint ainsi que la conception en utilisant les diagrammes de cas d'utilisation, de classe, de séquence et d'activité. Nous terminons par des captures d'écran pour mieux expliquer la réalisation de ce sprint.

1 Backlog du sprint 2

Nous estimons un travail quotidien de six heures pendant quatre semaines pour la réalisation de ce sprint.

TABLE 4.1 – Backlog Sprint 2

User story	ID	Tâches	Estimation(/h)
En tant que développeur, je peux gérer un projet ADL.	1.1	Backend : Création des model, controller et route de la table «ProjetADL».	12
	1.2	Frontend : Ajouter l'interface de création d'un nouveau projet ADL.	3
	1.3	Recherche et développement de la squelette du code en langage ADL	15
	1.4	Backend : Automatisation de la génération d'un fichier texte contenant le code ADL de la squelette du projet.	12
	1.5	Intégrer et tester le fonctionnement de la partie frontend et backend.	12

Page suivante

TABLE 4.1 – Backlog Sprint 2

User story	ID	Tâches	Estimation(/h)
En tant que développeur, je peux gérer la palette des composantes ADL.	2.1	Backend : Création des model, controller et route de la table «Composant».	12
	2.2	-Fontend : Création de l'interface de la liste des composants. -Intégration : Affichage de la liste des composant à partir de la base de données.	6
	2.3	Recherche et développement du code ADL des composants GP.	12
	2.4	-Frontend : Ajouter l'interface de création d'un nouveau composant ADL.	6

2 Diagramme de cas d'utilisation du Sprint 2

Dans cette section, nous présentons le diagramme de cas d'utilisation raffiné du Sprint 2 et les descriptions textuelles relatives aux cas d'utilisations.

2.1 Présentation du diagramme de cas d'utilisation

La figure 4.1 présente le diagramme de cas d'utilisation du Sprint 2.

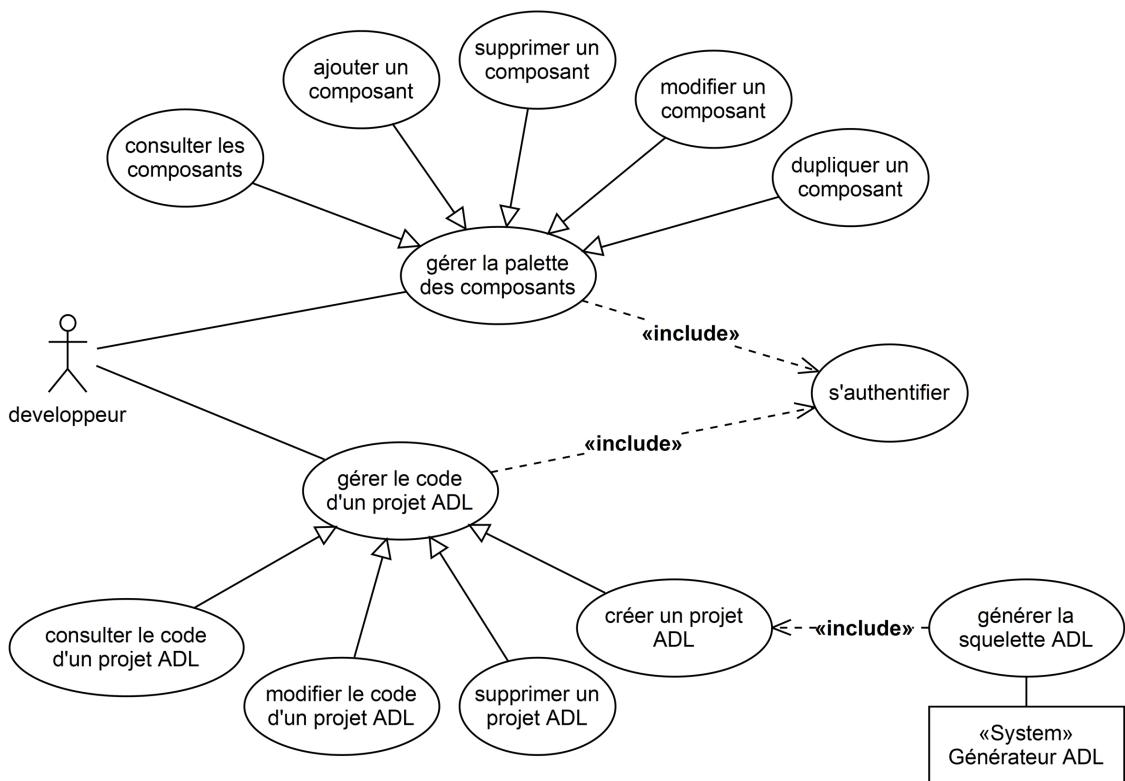


FIGURE 4.1 – Diagramme de cas d'utilisation du sprint 2

2.2 Descriptions textuelles

Dans cette partie, nous analysons les cas et sous cas d'utilisations représentés dans le diagramme du Sprint 2.

- **Description textuelle du CU "Créer un nouveau projet ADL"**

TABLE 4.2 – Description textuelle du CU "Créer un nouveau projet ADL"

Cas d'utilisation	Créer un nouveau projet ADL
Acteurs	Développeur
Résumé	Le développeur peut créer un nouveau projet ADL.
Pré-condition	Développeur authentifié.
Post-condition	Génération automatique d'un fichier texte contenant le code ADL du projet.
Page suivante	

TABLE 4.2 – Description textuelle du CU "Créer un nouveau projet ADL"

Cas d'utilisation	Créer un nouveau projet ADL
Scénario principal	<ul style="list-style-type: none"> ❶ Le développeur clique sur "Nouveau" dans la barre de navigation ❷ Le développeur remplit les champs du formulaire affiché ❸ Le système génère un fichier contenant la squelette basique d'un projet ADL ❹ Le système ajoute le chemin du fichier dans la base de données

Chaque développeur doit tout d'abord être authentifié pour créer un nouveau projet ADL. Pour créer un nouveau projet ADL, il faut choisir l'option de création d'un nouveau projet dans la barre de navigation, remplir le formulaire affiché et cliquer sur le bouton "Créer". Ainsi, le nouveau projet ADL sera créé et ajouté à notre base de données.

• **Description textuelle du CU "Supprimer un projet ADL"**

TABLE 4.3 – Description textuelle du CU "Supprimer un projet ADL"

Cas d'utilisation	Supprimer un projet ADL
Acteurs	Développeur
Résumé	Le développeur peut créer un nouveau projet ADL.
Acteurs	Développeur
Résumé	Le développeur peut supprimer un projet existant.
Pré-condition	Développeur authentifié.
Post-condition	Suppression du projet ADL de la base de données.
Page suivante	

TABLE 4.3 – Description textuelle du CU "Supprimer un projet ADL"

Cas d'utilisation	Supprimer un projet ADL
Scénario principal	<ul style="list-style-type: none"> ❶ Le développeur clique sur "Ouvrir" dans la barre de navigation ❷ Le développeur choisit un projet existant ❸ Le développeur clique sur l'icône de suppression ❹ Le développeur confirme son choix. ❺ Le système supprime le projet de la base de données.
Scénario alternatif	<p>Le développeur annule son choix :</p> <p>❻ L'opération sera annulée si le développeur clique sur le bouton "Annuler".</p>

Chaque développeur doit tout d'abord être authentifié pour supprimer un projet ADL.

Pour supprimer un projet ADL, il faut choisir l'option qui permet d'ouvrir un projet existant, choisir le projet et cliquer sur le bouton "Supprimer".

Si le développeur confirme la suppression, le projet ADL sera supprimé de la base de données.

Dans le cas contraire, l'opération sera annulée et aucun projet ne sera supprimé.

- **Description textuelle du CU "Dupliquer un composant de la palette"**

TABLE 4.4 – Description textuelle du CU "Dupliquer un composant de la palette"

Cas d'utilisation	Dupliquer un composant de la palette
Acteurs	Développeur
Pré-condition	Développeur authentifié.
Post-condition	Composant dupliqué
Page suivante	

TABLE 4.4 – Description textuelle du CU "Dupliquer un composant de la palette"

Cas d'utilisation	Dupliquer un composant de la palette
Scénario nominal	<p>❶ Le développeur clique sur "Palette" dans la barre de navigation latérale.</p> <p>❷ Le développeur choisit le composant et clique sur l'icône de duplication.</p> <p>❸ Le développeur valide son choix.</p> <p>❹ Le système duplique ce composant et l'ajoute dans la base de données.</p>
Scénario alternatif	<p>Le développeur annule son choix :</p> <p>❺ L'opération sera annulée si le développeur clique sur le bouton "Annuler".</p>

Chaque développeur doit tout d'abord être authentifié pour dupliquer un composant de la palette.

Pour dupliquer un composant de la palette, il faut accéder à la palette, choisir la componante et cliquer sur le bouton "Dupliquer".

Si le développeur valide la duplication, un nouveau composant, identique au composant choisi, sera créé et enregistré dans la base de données.

Dans le cas contraire, l'opération sera annulée et aucun composant ne sera créé.

- **Description textuelle du CU "Modifier un composant dans la palette"**

TABLE 4.5 – Description textuelle du CU "Modifier un composant dans la palette"

Cas d'utilisation	Modifier un composant dans la palette
Acteurs	Développeur
Résumé	Le développeur peut modifier un composant dans la palette.
Pré-condition	Développeur authentifié.
Post-condition	Composant modifié
Page suivante	

TABLE 4.5 – Description textuelle du CU "Modifier un composant dans la palette"

Cas d'utilisation	Modifier un composant dans la palette
Scénario principal	<p>❶ Le développeur clique sur "Palette" dans la barre de navigation latérale.</p> <p>❷ Le développeur choisit le composant et clique sur l'icône de modification.</p> <p>❸ Le développeur modifie les champs de son choix.</p> <p>❹ Le développeur valide les modifications effectuées.</p> <p>❺ Le système met à jour les données de ce composant dans la base de données.</p>
Scénario alternatif	<p>Le développeur annule les modifications effectuées :</p> <p>❻ L'opération sera annulée si le développeur clique sur le bouton "Annuler".</p>

Chaque développeur doit tout d'abord être authentifié pour modifier un composant dans la palette.

Pour modifier un composant dans la palette, il faut accéder à la palette, choisir la composante et cliquer sur le bouton "Modifier" et remplir le formulaire avec les données souhaitées.

Si le développeur valide les modifications effectuées, les données du composant seront mises à jour dans la base de données.

Dans le cas contraire, l'opération sera annulée et aucune donnée ne sera modifiée.

3 Diagrammes de séquences

Dans cette partie, nous présentons les diagrammes de séquences détaillés des cas d'utilisation du Sprint 2.

- **Diagramme de séquences CU «Consulter un projet ADL»**

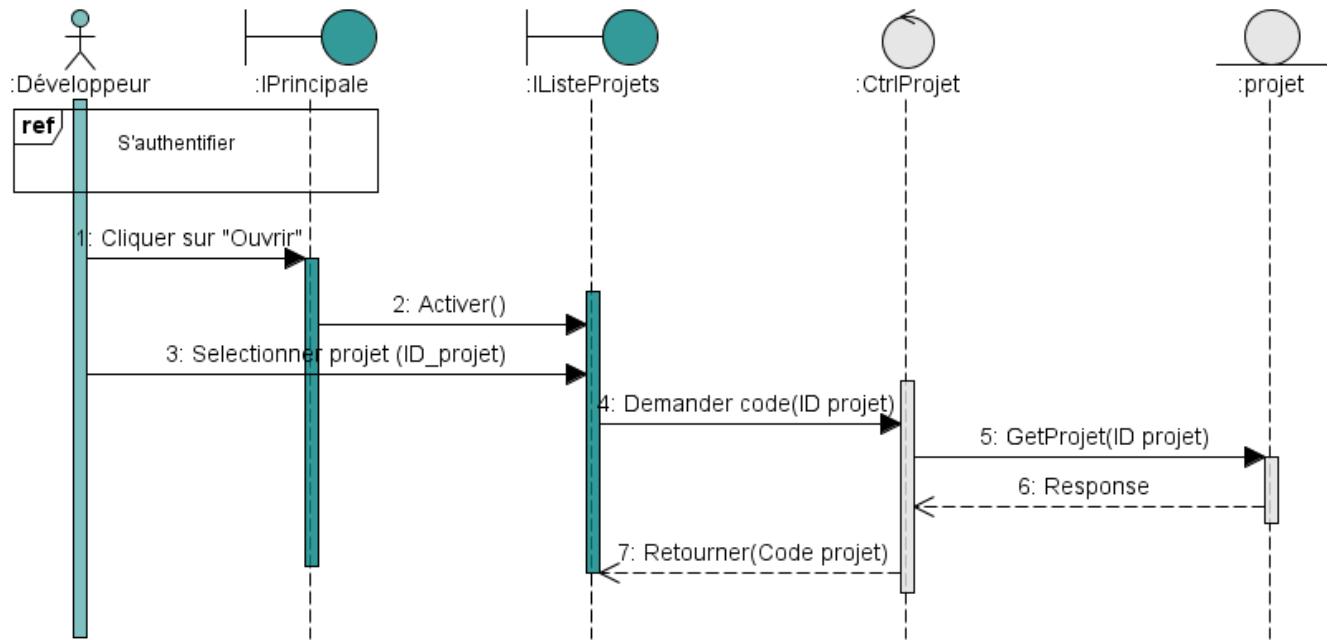


FIGURE 4.2 – Diagramme de séquences CU «Consulter un projet ADL»

- **Diagramme de séquences CU «Créer un nouveau projet ADL»**

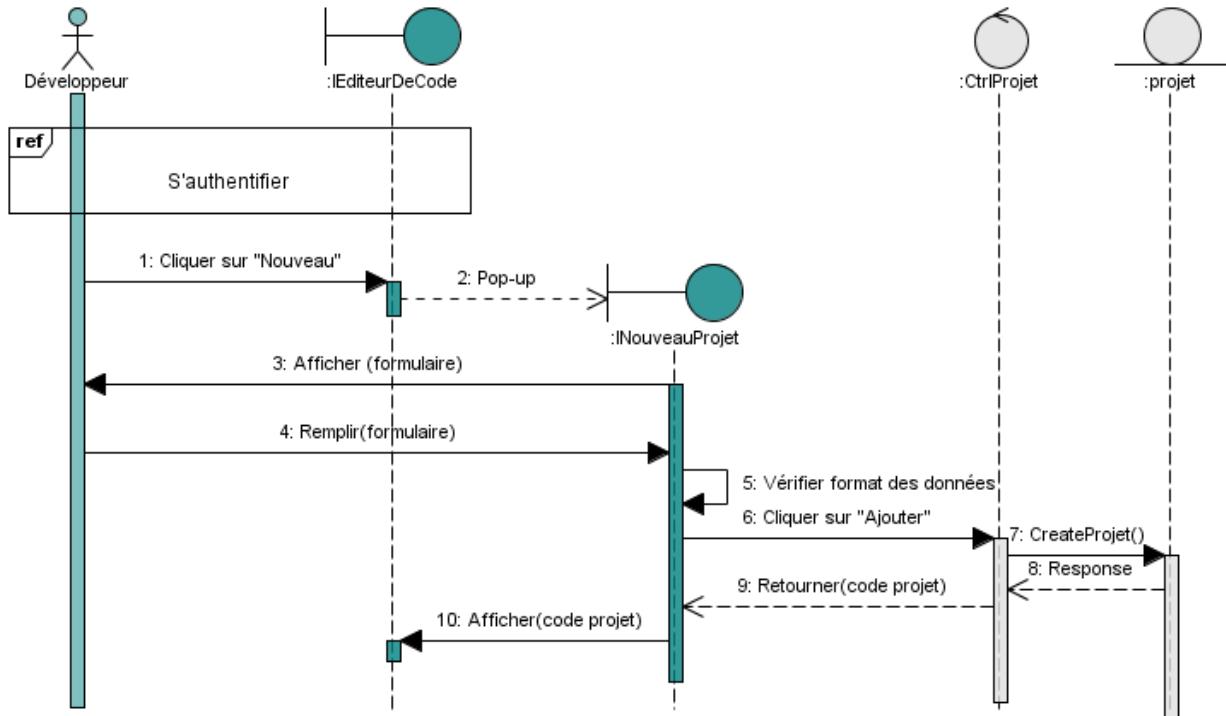


FIGURE 4.3 – Diagramme de séquences CU «Créer un nouveau projet ADL»

- **Diagramme de séquences CU «Supprimer un projet ADL»**

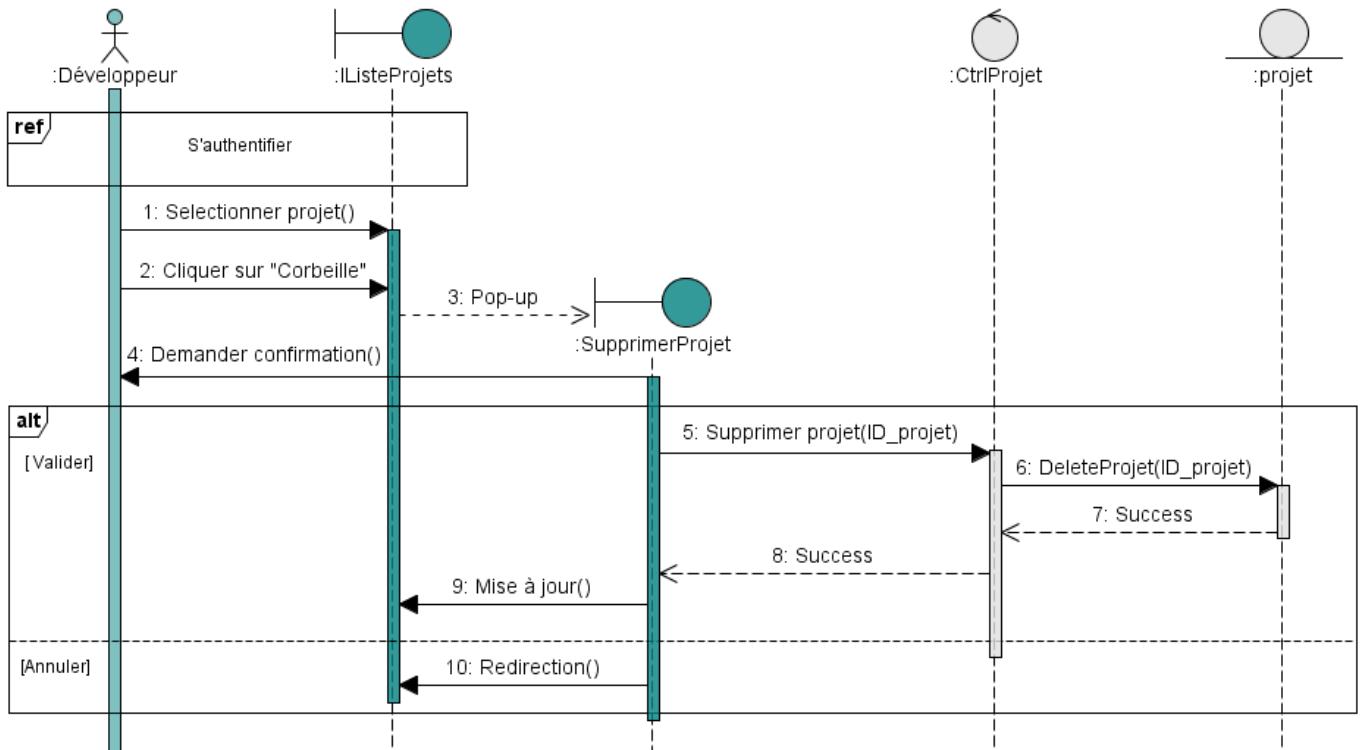


FIGURE 4.4 – Diagramme de séquences CU «Supprimer un projet ADL»

- Diagramme de séquences CU «Générer squelette ADL»

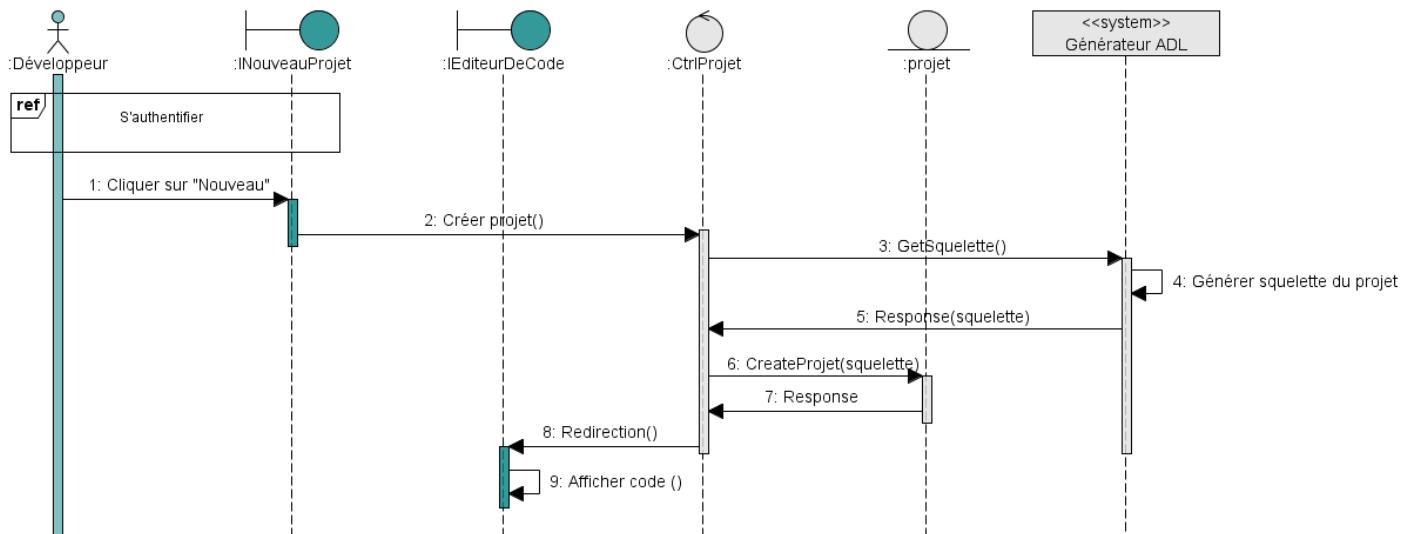


FIGURE 4.5 – Diagramme de séquences CU «Générer squelette ADL»

4 Diagramme de classes du Sprint2

La figure 4.6 présente le diagramme de classes relatif au Sprint2.

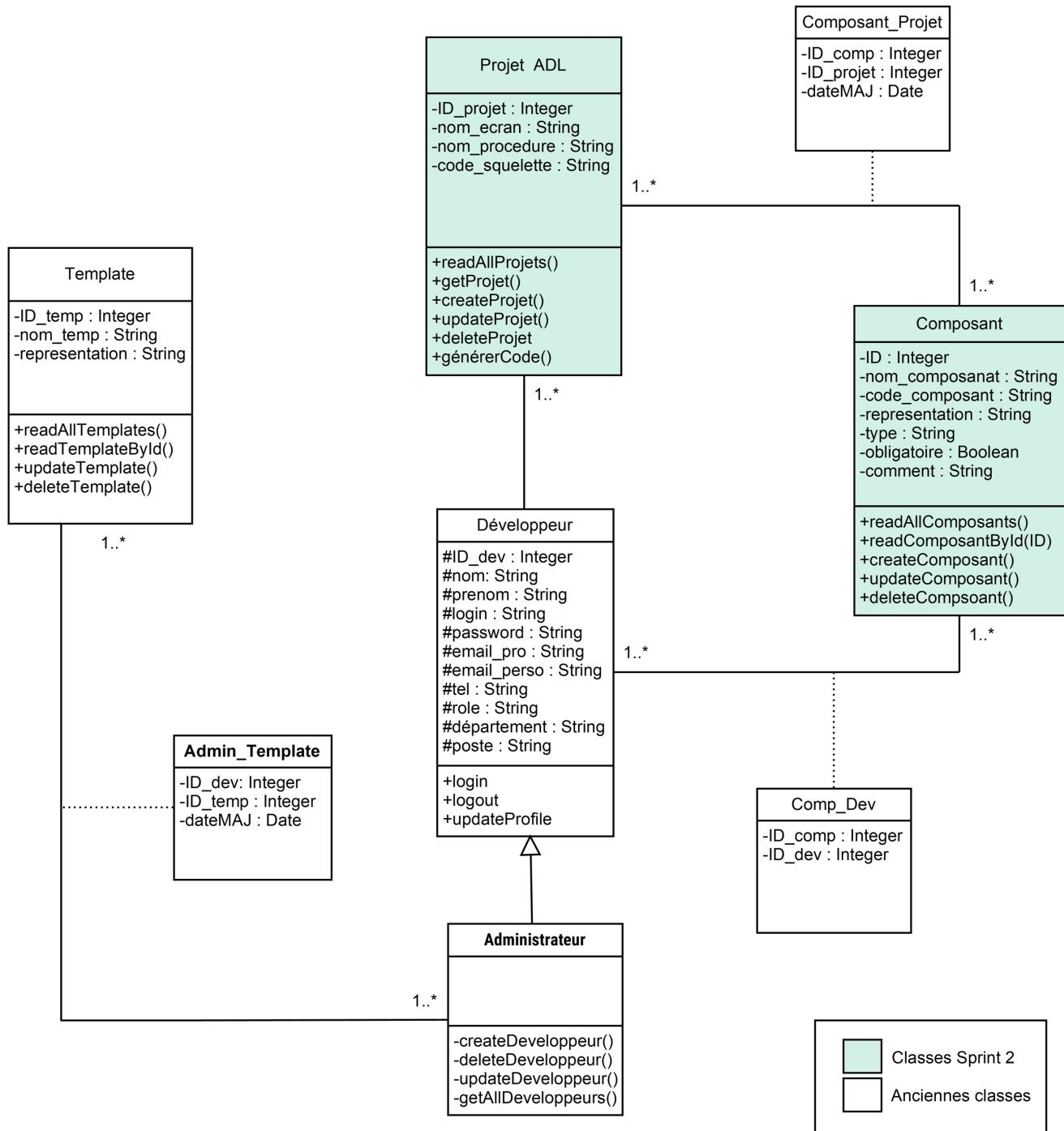


FIGURE 4.6 – Diagramme de classes du Sprint2

5 Modélisation des processus métier

- Diagramme d'activité CU «Ajouter un composant»

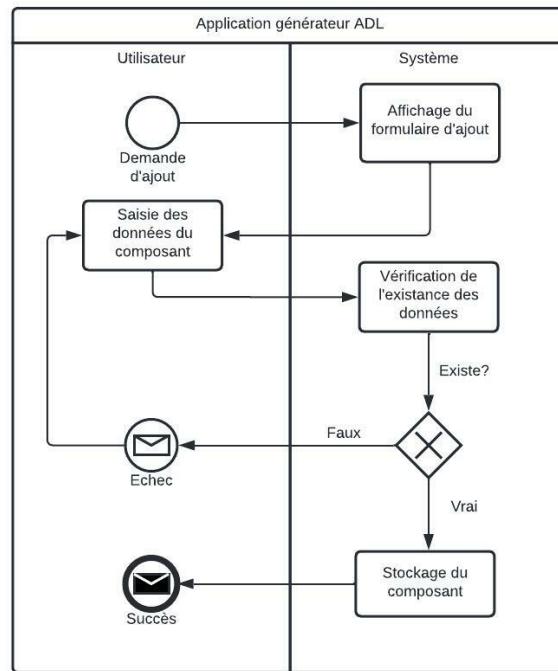


FIGURE 4.7 – Diagramme d'activité CU «Ajouter un composant»

- **Diagramme d'activité CU «Supprimer un composant»**

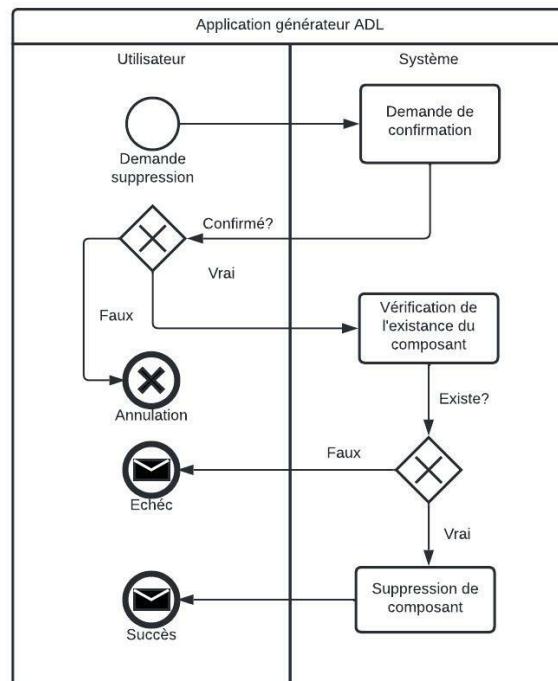


FIGURE 4.8 – Diagramme d'activité CU «Supprimer un composant»

6 Réalisation

- **Gérer un projet ADL**

Pour gérer un projet, le développeur clique sur «Ouvrir» dans la barre de navigation. Ainsi, la liste des projets ADL lui sera affichée comme le montre la figure 4.9.

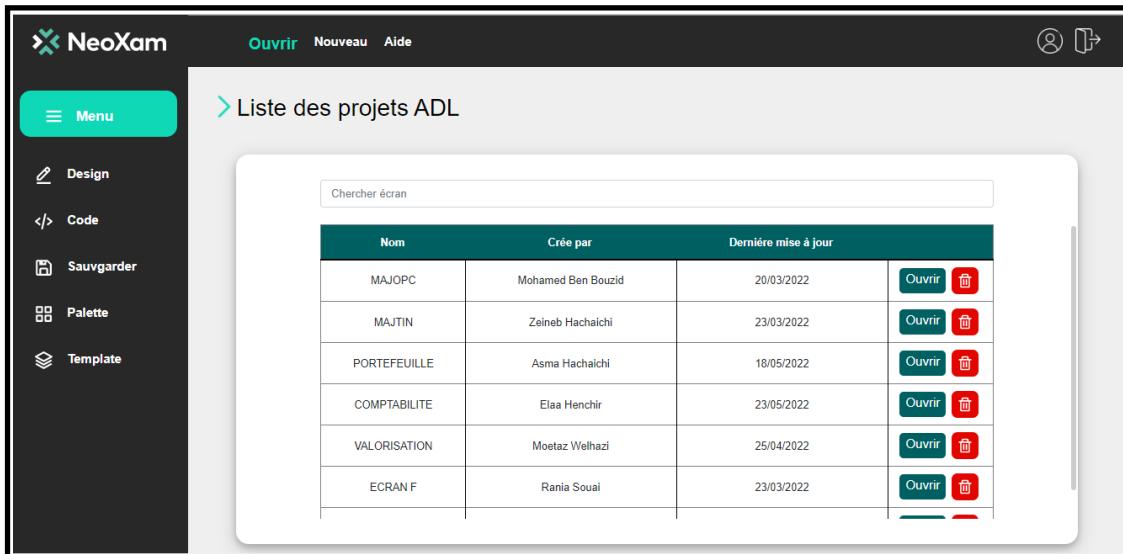


FIGURE 4.9 – Interface Liste des projets ADL

Pour supprimer un projet ADL, le développeur clique sur l'icône de la corbeille, une fenêtre de validation s'affichera comme suit :



FIGURE 4.10 – Pop-up Supprimer projet

Le développeur peut aussi créer un nouveau projet ADL en choisissant l'option «Nouveau» dans la barre de navigation.

Il doit ensuite remplir le formulaire suivant :

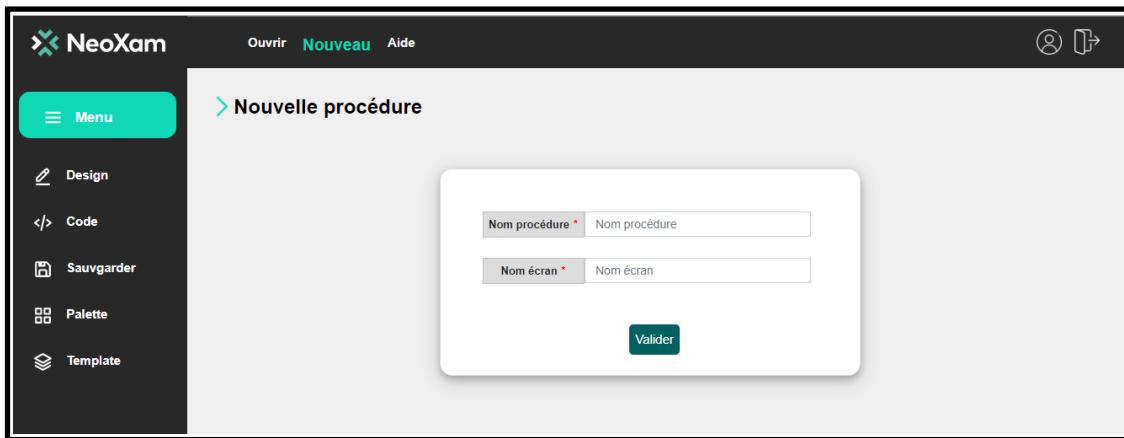


FIGURE 4.11 – Interface Nouveau projet

- **Gérer la palette des composants**

Pour la gestion de la palette, il faut accéder à l'interface «Liste composant», via le bouton *Palette* dans la barre de navigation latérale.

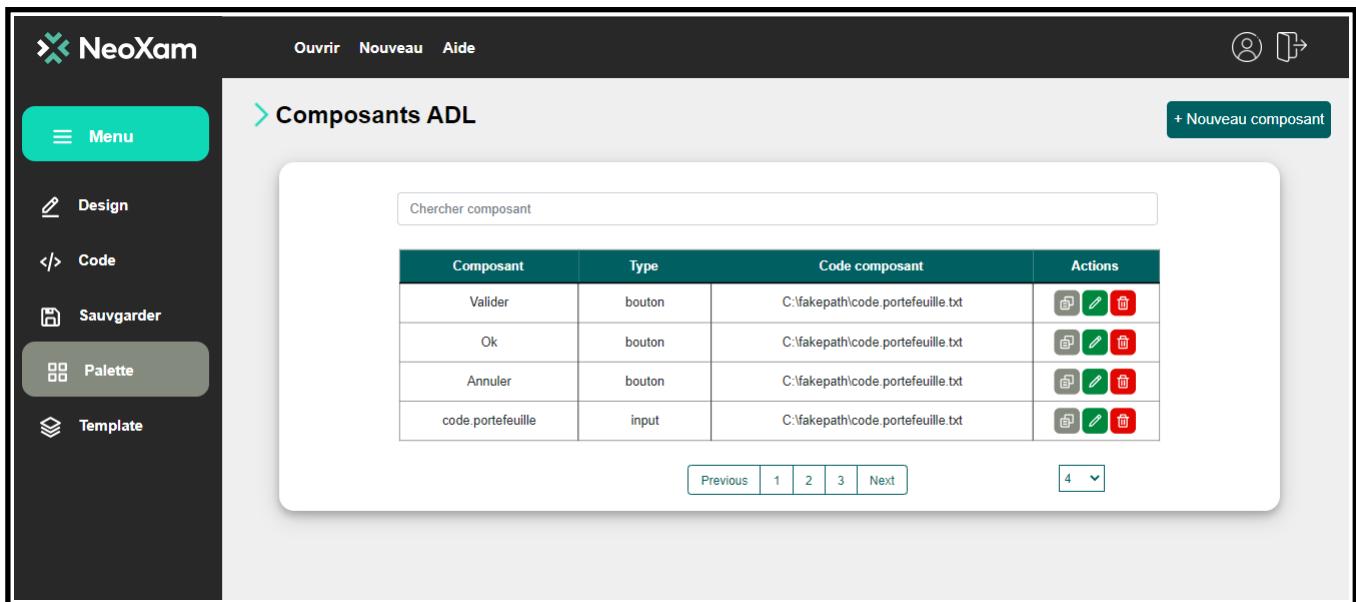


FIGURE 4.12 – Interface Liste composants en affichant 4 composants par page

Le développeur peut modifier le nombre de composants qu'il souhaite voir par page (maximum 20 composants par page).

Composant	Type	Code composant	Actions
Valider	button	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Ok	button	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Annuler	button	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
code.portefeuille	input	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
code.valo	input	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Date.debut	date	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Date.fin	date	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Date.compta	date	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Date.historique	date	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]
Jaccepte	checkbox	C:/fakepath/code.portefeuille.txt	[Edit, Save, Delete]

Previous 1 2 Next 10

FIGURE 4.13 – Interface Liste composants en affichant 10 composants par page

Pour ajouter un composant, le développeur doit cliquer sur le bouton *+ Nouveau composant* et remplir les champs de la pop-up dans la figure 4.14.

FIGURE 4.14 – Pop-up Ajouter composant

Pour supprimer un composant, le développeur doit cliquer sur l'icône de la corbeille. Ainsi, une pop-up de confirmation sera affichée.

FIGURE 4.15 – Pop-up Supprimer composant

Pour modifier un composant, le développeur doit cliquer sur l'icône du stylo et remplir le formulaire présenté dans la figure 4.16.

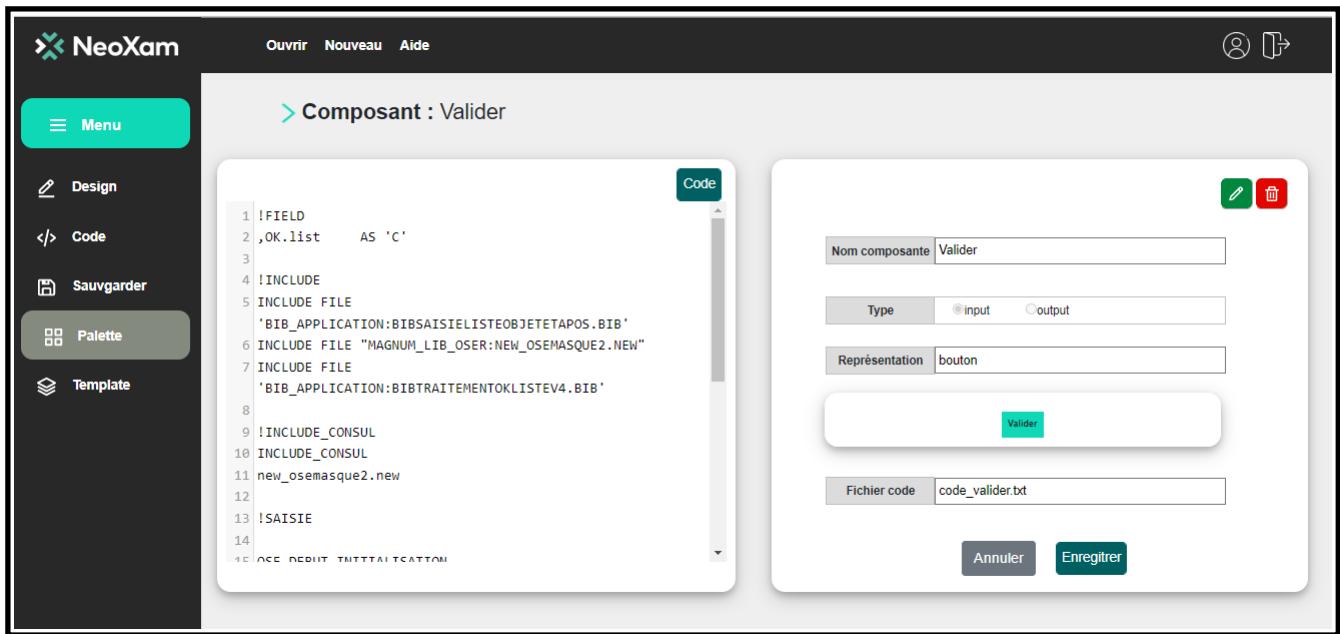


FIGURE 4.16 – Interface Modifier composant

7 Revue du Sprint2

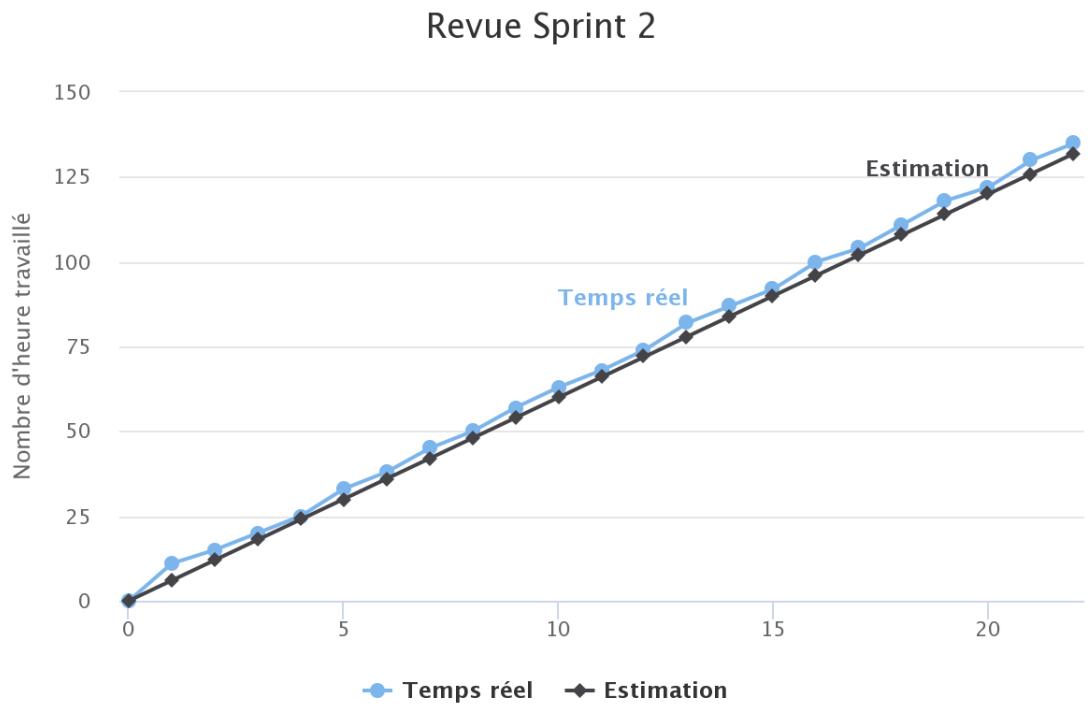


FIGURE 4.17 – Revue du Sprint2

D'après le graphe 3.15 ci-dessus, la courbe du temps réel (représentée en bleue) est légèrement au dessus de la droite idéale (représentée en noir).

C'est assez normal puisque durant cette période, nous prenons connaissance de nouvelles tâches embarquées dans le sprint courant.

Conclusion

Au cours de ce quatrième chapitre, nous avons exposé la conception et la réalisation du sprint2 : "La gestion d'un projet ADL et des composants de la palette". Le chapitre suivant est réservé pour la présentation du dernier sprint.

CHAPITRE 5

SPRINT3 : GÉNÉRATION DU CODE ADL

Introduction	57
1 Backlog du sprint 3	57
2 Diagramme de cas d'utilisation du Sprint3	58
2.1 Présentation du diagramme de cas d'utilisation	58
2.2 Descriptions textuelles	58
3 Diagrammes de séquences	61
4 Diagramme de classes du Sprint3	62
5 Réalisation	62
6 Revue du sprint3	69
7 Conclusion	70

Introduction

Ce chapitre est dédié pour la réalisation et la présentation du sprint 3, intitulé "Génération du code ADL", dans lequel nous présentons l'analyse et les spécifications des besoins pour ce dernier sprint ainsi que la conception en utilisant les diagrammes de cas d'utilisation, de classes et de séquences.

1 Backlog du sprint 3

Nous estimons un travail quotidien de six heures pendant quatre semaines pour la réalisation de ce sprint.

TABLE 5.1 – Backlog Sprint3

User story	ID	Tâches	Estimation (/h)
En tant que développeur, je peux ajouter un composant dans un écran GP.	1.1	Backend : Recherche sur la manipulation de fichiers en NodeJS avec fs.	12
	1.2	Backend : Création des controllers de gestion de fichier et de génération de code.	24
	1.4	Frontend : Ajouter l'interface design et intégrer la fonction d'ajout par click.	18
	1.3	Intégrer et tester le fonctionnement de la partie frontend et backend	12

Page suivante

TABLE 5.1 – Backlog Sprint3

User story	ID	Tâches	Estimation (/h)
En tant que développeur, je peux supprimer un composant d'un écran GP.	2.1	Backend : Développement de la fonction de suppression de code ADL.	12
	2.2	Intégrer et tester le fonctionnement de la partie frontend et backend	12

2 Diagramme de cas d'utilisation du Sprint3

2.1 Présentation du diagramme de cas d'utilisation

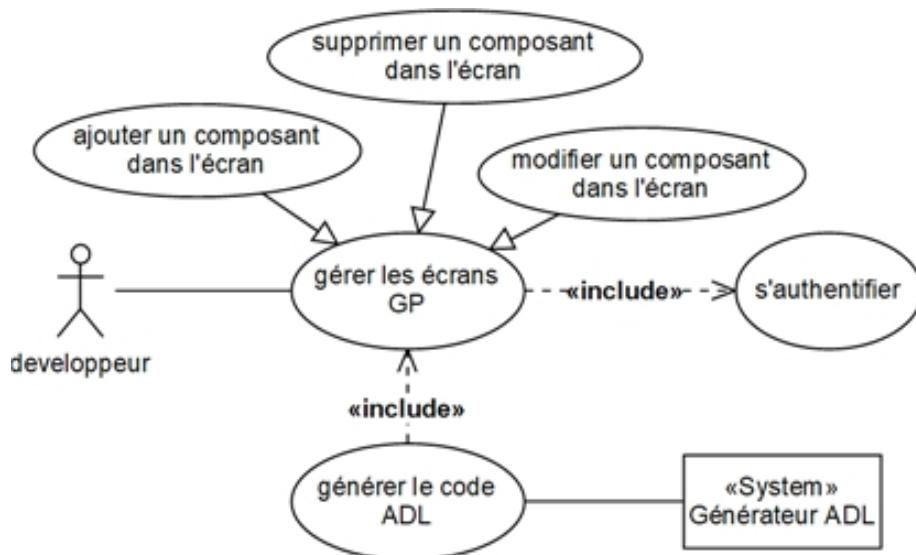


FIGURE 5.1 – Diagramme de cas d'utilisation du sprint3

2.2 Descriptions textuelles

- **Description textuelle du sous cas d'utilisation "Ajouter un composant dans l'écran"**

Le tableau ci-dessous contient la description textuelle du cas d'utilisation "Ajouter un composant dans l'écran".

TABLE 5.2 – Description textuelle du CU "Ajouter un composant dans l'écran"

Cas d'utilisation	Ajouter un composant dans l'écran
Acteurs	Développeur/Administrateur
Résumé	Le développeur peut ajouter un composant dans l'écran à partir de la zone de conception.
Pré-condition	Développeur authentifié.
Post-condition	Le composant s'ajoute dans l'écran
Scénario principal	<p>Pour ajouter un composant :</p> <ul style="list-style-type: none"> ❶ Le développeur accède à la partie "Design" ❷ Le développeur clique sur un composant de la palette ❸ Le composant s'ajoute dans l'écran affiché dans la zone de conception

Pour ajouter un composant dans l'écran, le développeur doit tout d'abord s'authentifier. Ensuite, il accède à la partie design et clique sur un composant de la palette. Ainsi, le composant s'ajoutera et s'affichera dans la zone de conception.

• **Description textuelle du sous cas d'utilisation "Supprimer un composant de l'écran"**

Le tableau ci-dessous contient la description textuelle du cas d'utilisation "Supprimer un composant de l'écran".

TABLE 5.3 – Description textuelle du CU "Supprimer un composant de l'écran"

Cas d'utilisation	Supprimer un composant de l'écran
Acteurs	Développeur
Résumé	Le développeur peut supprimer un composant de l'écran à partir de la zone de conception.
Pré-condition	Développeur authentifié.
Post-condition	Le composant est retiré de l'écran
Page suivante	

TABLE 5.3 – Description textuelle du CU "Supprimer un composant de l'écran"

Cas d'utilisation	Supprimer un composant de l'écran
Scénario principal	<p>Pour supprimer un composant de l'écran :</p> <ul style="list-style-type: none"> ❶ Le développeur accède à la partie "Design" ❷ Le développeur clique sur le composant dans l'écran qu'il désire supprimer ❸ Le système demande une confirmation ❹ Le développeur valide son choix ❺ Le système retire le composant de l'écran
Scénario Alternatif	<p>❻ L'opération sera annulée si le développeur clique sur "Annuler"</p>

Chaque développeur doit tout d'abord être authentifié pour supprimer un composant de l'écran.

Pour supprimer un composant de l'écran, il faut cliquer sur le composant qu'il désire supprimer.

Si le développeur confirme la suppression, le composant sera retiré de l'écran.

Dans le cas contraire, l'opération sera annulée et aucun composant ne sera supprimé.

- **Description textuelle du sous cas d'utilisation "Générer le code ADL"**

Le tableau ci-dessous contient la description textuelle du cas d'utilisation "Générer le code ADL".

TABLE 5.4 – Description textuelle du CU "Générer le code ADL"

Cas d'utilisation	Générer le code AD
Acteurs	Développeur / Générateur ADL
Résumé	Générateur doit générer le nouveau code de l'écran.
Pré-condition	Développeur authentifié.
Post-condition	Le fichier contenant le code de l'écran est mis à jour en ajoutant le code du nouveau composant
Page suivante	

TABLE 5.4 – Description textuelle du CU "Générer le code ADL"

Cas d'utilisation	Générer le code ADL
Scénario principal	<p>Pour générer le code ADL :</p> <ul style="list-style-type: none"> ❶ Le développeur ajoute un composant de la palette dans l'écran ❷ Le développeur valide l'ajout ❸ Le générateur ADL copie le code du composant ❹ Le générateur ADL met à jour le code de l'écran en ajoutant le code du composant ❺ Le système enregistre le nouveau fichier du code

Pour que le code soit généré, le développeur doit être authentifié.

Ce dernier commence par ajouter un composant dans l'écran.

Ensuite, le développeur valide son ajout pour que le générateur ADL ajoute le code ADL du composant choisi au code de l'écran GP.

3 Diagrammes de séquences

- **Diagramme de séquences CU « Génération du code ADL »**

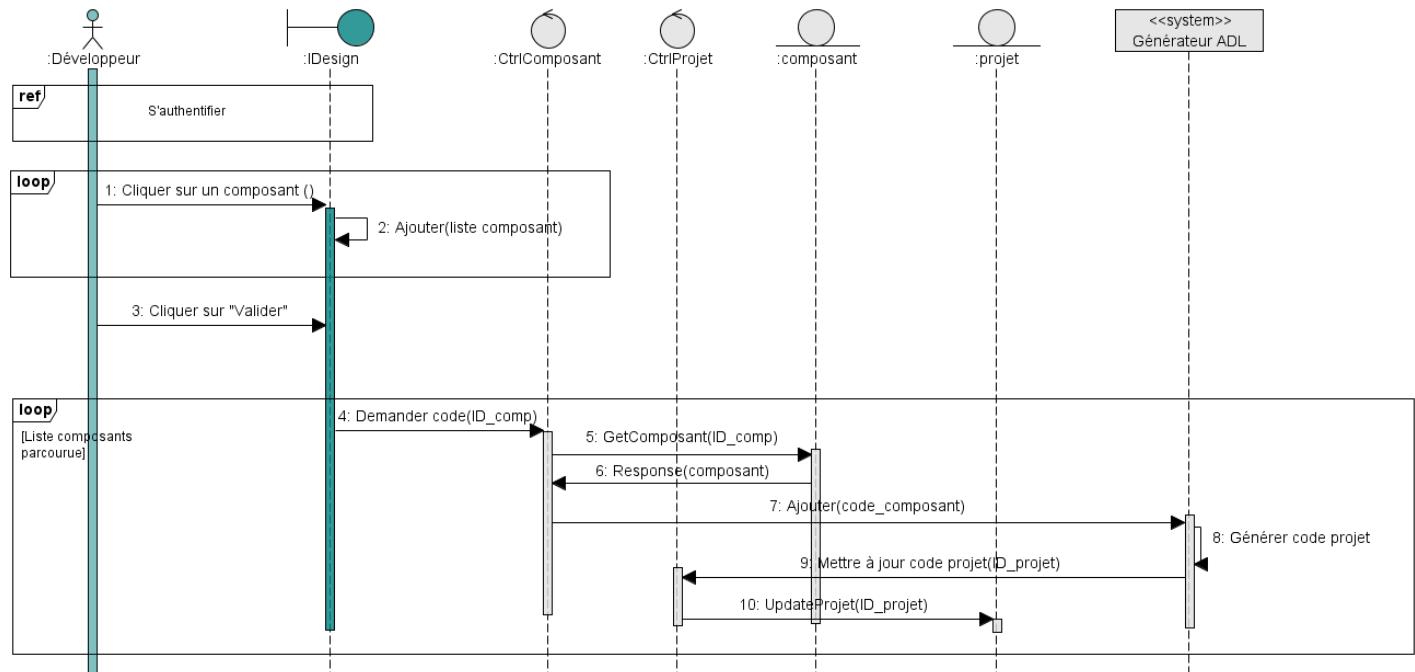


FIGURE 5.2 – Diagramme de séquences CU « Génération du code ADL »

4 Diagramme de classes du Sprint3

La figure présente le diagramme de classes relatif au sprint3.

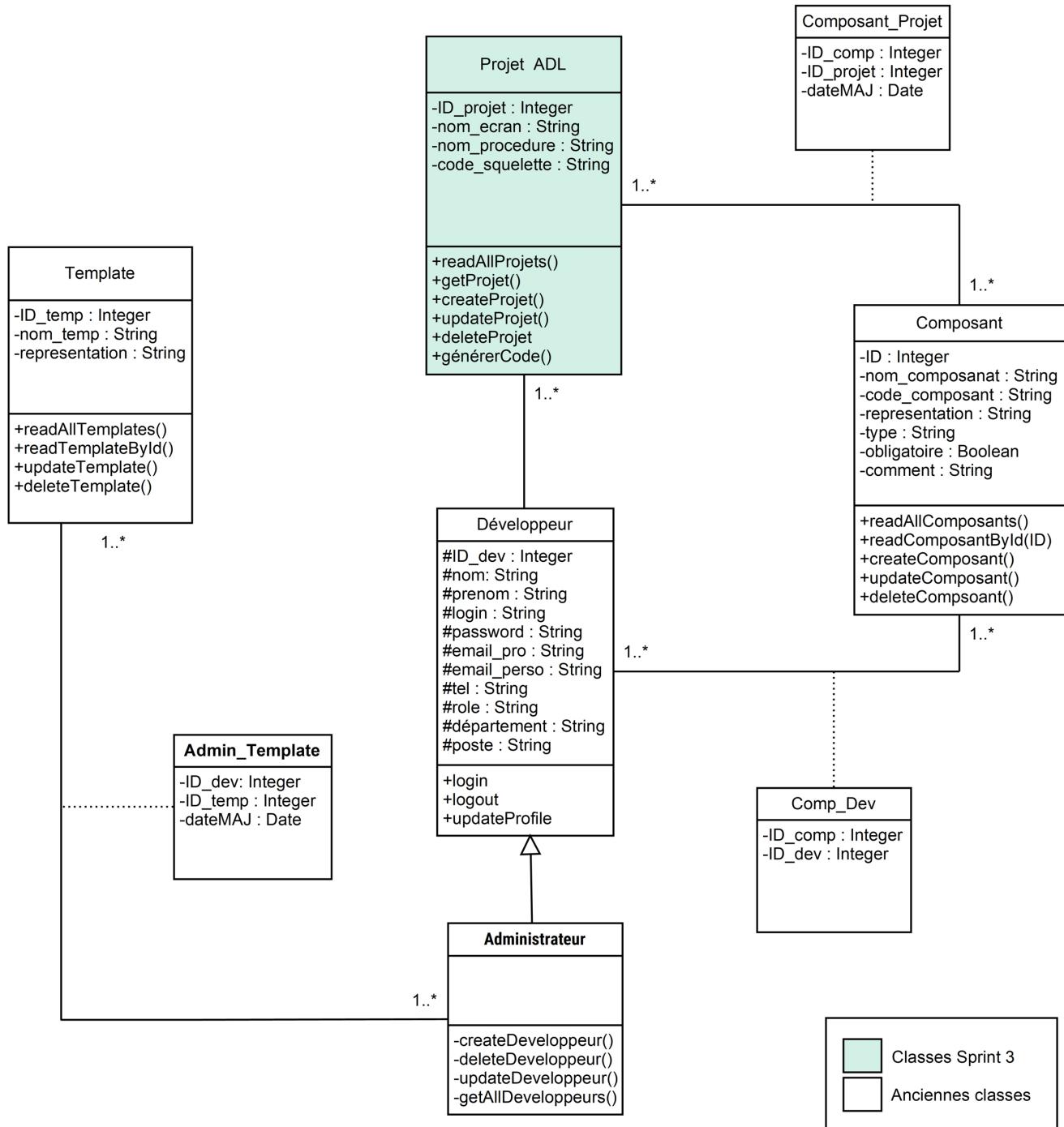


FIGURE 5.3 – Diagramme de classes du Sprint3

5 Réalisation

Dans cette partie, nous allons présenter quelques captures des différents modules réalisés au cours du Sprint 3.

- **Interface Code**

Cette interface contient un éditeur de code dans lequel s'affiche le code du projet ADL.

Le développeur peut modifier le code de son projet à travers cet éditeur.

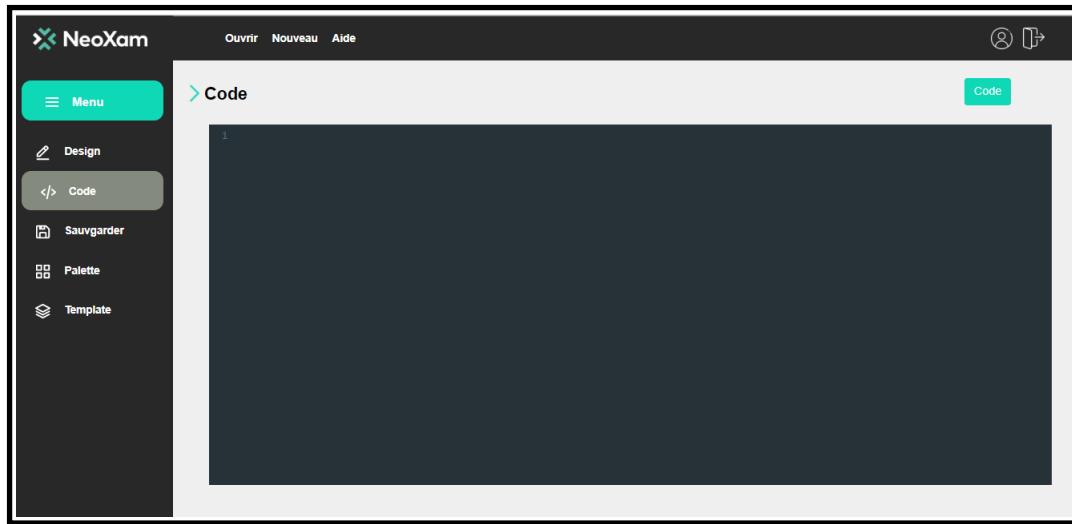


FIGURE 5.4 – Interface code vide

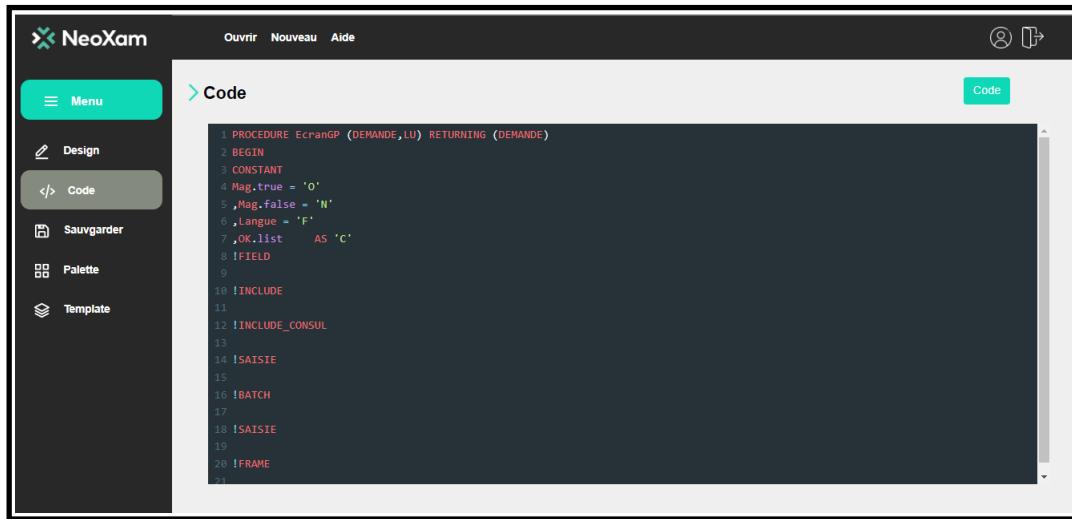


FIGURE 5.5 – Interface code avec du code

- **Interface Design**

Cette interface contient une zone de conception et une palette.

Dans la zone de conception s'affiche l'écran correspondant au projet ADL.

Le développeur peut ajouter des composants dans l'écran à partir de la palette à droite.

- **Schéma du fonctionnement du mode design**

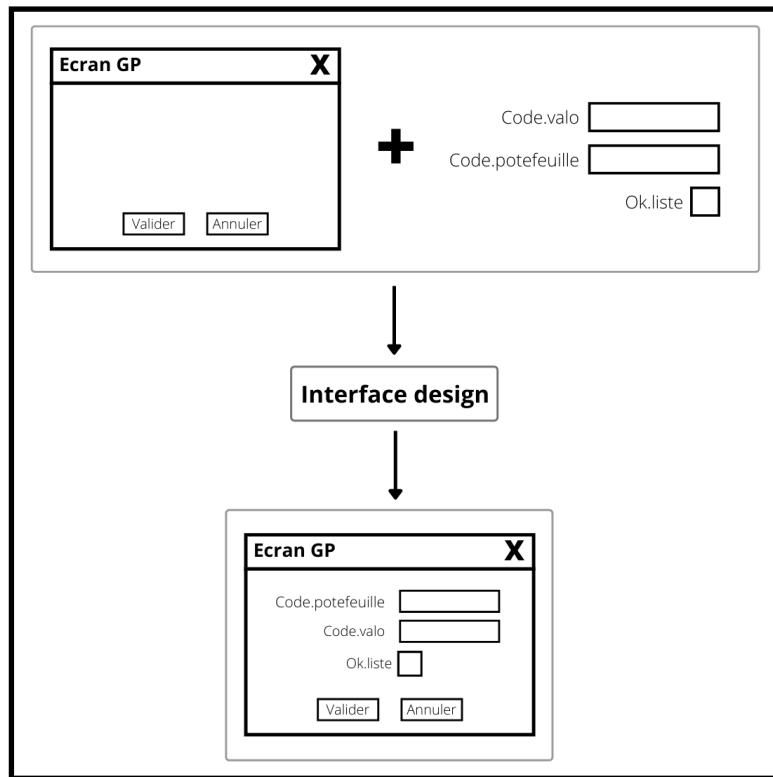


FIGURE 5.6 – Schéma de fonctionnement du mode design

La figure 5.7 présente l'interface "Design" contenant un écran initialement vide.

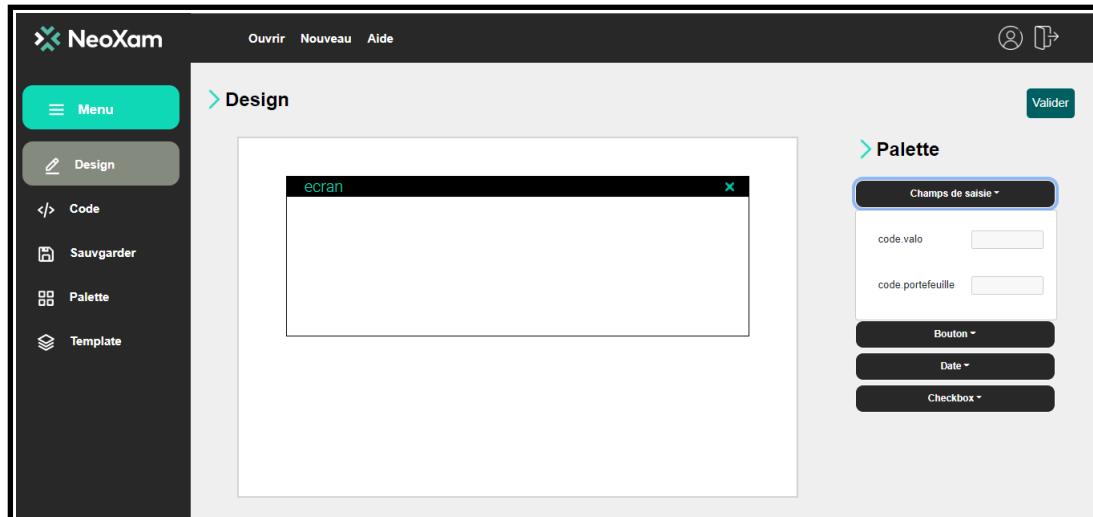


FIGURE 5.7 – Interface design vide

La figure 5.8 présente la même interface avec un écran content des composants de la palette.

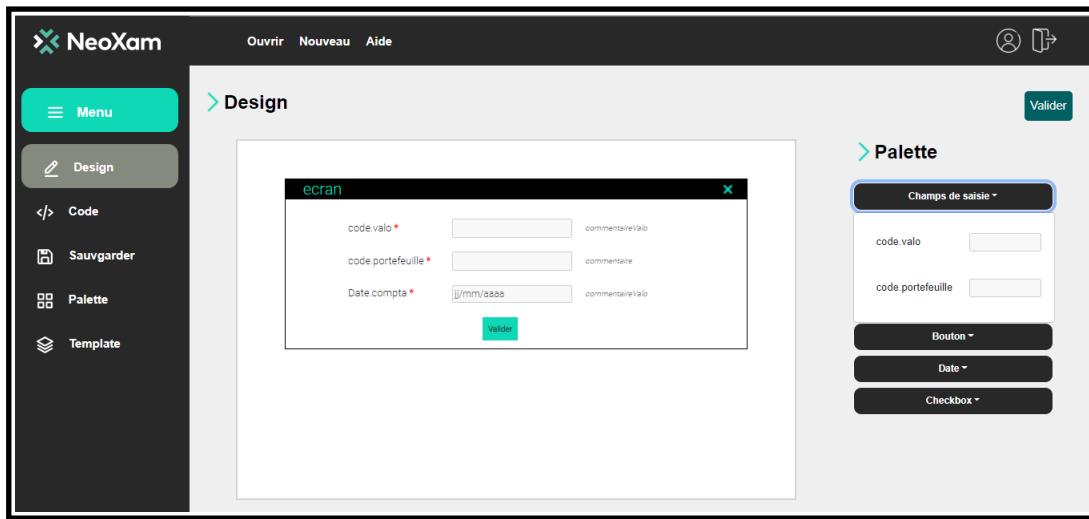


FIGURE 5.8 – Interface design avec des composants

Le développeur peut aussi supprimer un composant de l'écran.



FIGURE 5.9 – Pop-up supprimer

- **Générer le code ADL**

Explication du raisonnement

Afin de générer le code correctement, il nous a fallu en premier lieu de faire des recherches approfondies sur les écrans ADL et leurs composants.

Ensuite, nous avons défini un code ADL standard pour chaque composant GP.

En effet, chaque code correspondant à un composant contient plusieurs sections, dont on peut citer parmi les plus répondues :

- FIELD : La section dans laquelle on déclare les variables
- INCLUDE : La section dans laquelle on importne les bibliothèques nécessaires
- OSE.INPUT : La section dans laquelle on met les conditions de saisie

Notre générateur est supposé insérer chaque partie du code dans sa section correspondante comme indiqué dans la figure 5.10.

Afin de localiser chaque partie dans le code ADL, nous avons utilisé des «tags» sous forme de commentaires (en ADL, un commentaire commence par un point d'exclamation :!This

(is a comment).

En début de chaque section, nous ajoutons un *tag* contenant le nom de cette partie.

Exemple : Exemple de code pour la section INCLUDE

```

INCLUDE

INCLUDE FILE 'BIB_APPLICATION:BIBSAISIELISTEOBJETETAPOS.BIB'
INCLUDE FILE "MAGNUM_LIB_OSER:NEW_OSEMASQUE2.NEW"
INCLUDE FILE 'BIB_APPLICATION:BIBTRAITEMENTOKLISTEV4.BIB'
```

Schéma du fonctionnement du générateur ADL

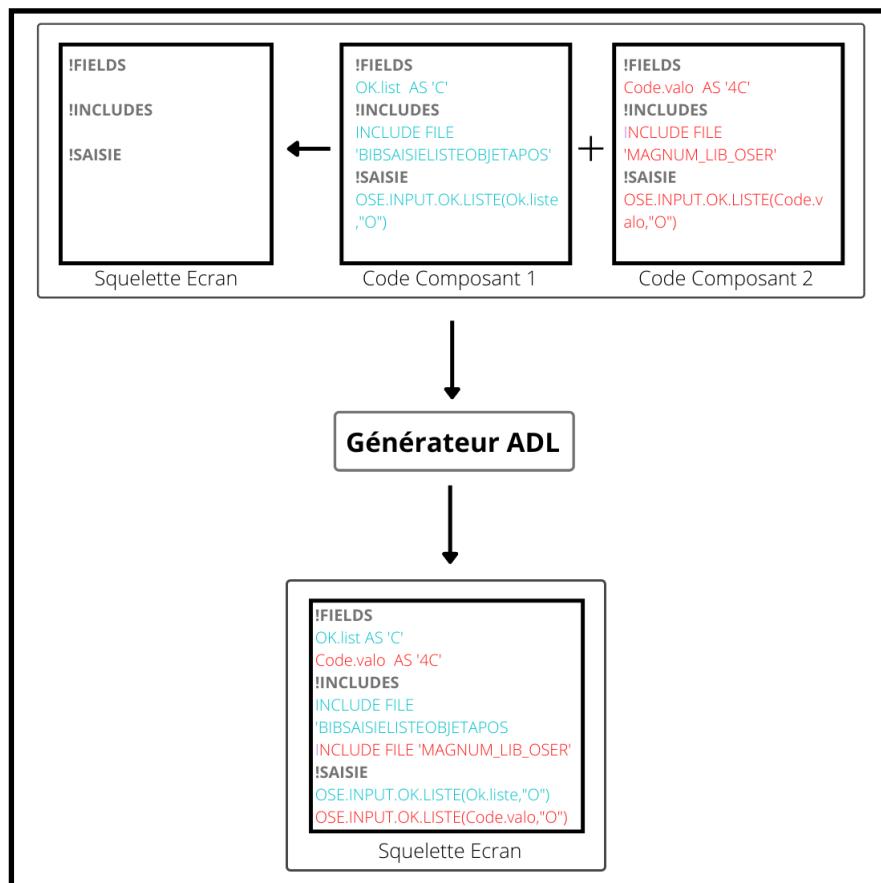


FIGURE 5.10 – Schéma de fonctionnement du générateur ADL

Les étapes détaillées

Les figures ci-dessous montrent les étapes de cette opération :

1. Code initial de l'écran

La création d'un nouveau projet engendre la génération du code suivant qui correspond à

la squelette standard d'un code ADL.

```
PROCEDURE EcranGP (DEMANDE,LU) RETURNING (DEMANDE)
BEGIN
CONSTANT
Mag.true = 'O'
, Mag.false = 'N'
, Langue = 'F'
!FIELD
,OK.list AS 'C'

!INCLUDE
!INCLUDE_CONSUL
!SAISIE
!BATCH
!SAISIE
!FRAME
|
```

FIGURE 5.11 – Interface code affichant le code initial

2. Écran initial

L'écran correspondant au code de la squelette, initialement vide, est le suivant :

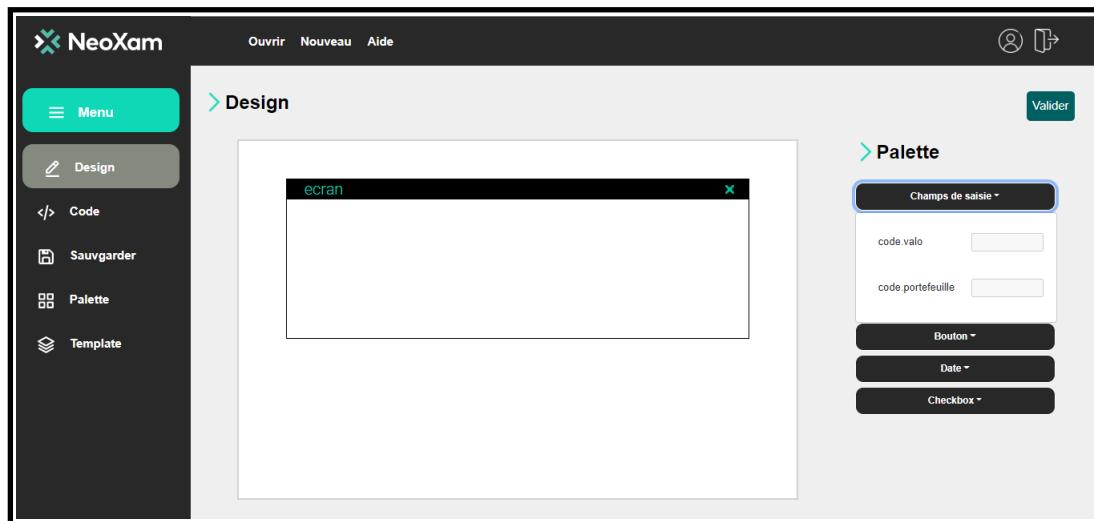


FIGURE 5.12 – Interface design vide

3. Ajouter un composant à l'écran

Ensuite, le développeur ajoute un composant à l'écran à partir de la palette :

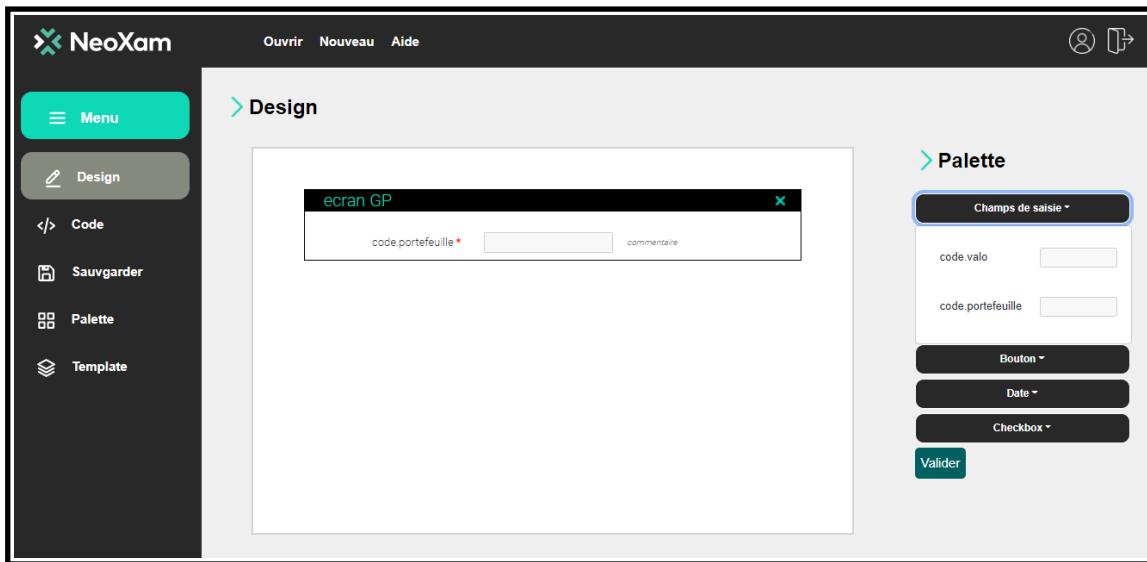


FIGURE 5.13 – Interface design

4. Valider l'ajout

Enfin, le développeur doit valider son choix en cliquant sur le bouton valider :



FIGURE 5.14 – Bouton Valider

5. Fichier généré

Le code du projet est re-généré après l'intégration des codes des composants ajoutés :

```

PROCEDURE EcranGP (DEMANDE,LU) RETURNING (DEMANDE)
BEGIN
CONSTANT
Mag.true = 'O'
, Mag.false = 'N'
, Langue = 'F'
!FIELD
,OK.liste AS 'C'

FIELDS
FRAME FORMAT Ecran GP
!INCLUDE
INCLUDE FILE 'BIB_APPLICATION:BIBSAISIELISTEOBJETETAPOS.BIB'
INCLUDE FILE "MAGNUM_LIB_OSER:NEW_OSEMASQUE2.NEW"
INCLUDE FILE 'BIB_APPLICATION:BIBTRAITEMENTOKLISTEV4.BIB'

!INCLUDE_CONSUL
new_osemasque2.new

!SAISIE
DSE.DEBUT.INITIALISATION
MOVE 'O' TO OK.LISTE
DSE.FIN.INITIALISATION OSE.INPUT.OK.LISTE(Ok.liste,"O")
DSE.DEBUT.CONTROLE
TRAITEMENT.OK.LISTE (OK.LISTE, TEMP, SLO.OBJET, 1)
DSE.FIN.CONTROLE

!BATCH
[IHM]
[FRAME]
"OK (O/N) : %%"
[FRAME_HELP]
!SAISIE

```

FIGURE 5.15 – Code du fichier généré

6. Code de l'écran après l'ajout du composant

Le développeur peut ainsi afficher le code généré dans la partie "Code" :

The screenshot shows the NeoXam application window. The left sidebar has buttons for 'Menu', 'Design', 'Code' (which is selected), 'Sauvgarder', 'Palette', and 'Template'. The main area is titled 'Code' and contains the following PL/I code:

```
1 PROCEDURE EcranGP (DEMANDE,LU) RETURNING (DEMANDE)
2 BEGIN
3 CONSTANT
4 Mag.true = 'O'
5 Mag.false = 'N'
6 Langue = 'F'
7 OK.list AS 'C'
8 !FIELD
9
10 FIELDS
11 FRAME FORMAT Ecran GP
12 !INCLUDE
13 INCLUDE FILE 'BIB_APPLICATION:BIBSAISIELISTEOBJETETAPOS.BIB'
14 INCLUDE FILE "MAGNUM_LIB_OSER:NEW_OSEMASQUE2.NEW"
15 INCLUDE FILE 'BIB_APPLICATION:BIBTRAITEMENTOKLISTEV4.BIB'
16
17 !INCLUDE_CONSUL
18 new_osemasque2.new
19
20 !SAISIE
21 OSE_DEBUT_INITIALISATION
```

FIGURE 5.16 – Interface code affichant le code après ajout

The screenshot shows the NeoXam application window. The left sidebar has buttons for 'Menu', 'Design', 'Code' (which is selected), 'Sauvgarder', 'Palette', and 'Template'. The main area is titled 'Code' and contains the following PL/I code:

```
21 OSE_DEBUT_INITIALISATION
22 MOVE '0' TO OK.LISTE
23 OSE_FIN_INITIALISATION OSE.INPUT.OK.LISTE(OK.liste,"0")
24 OSE_DEBUT.CONTROLE
25 TRAITEMENT.OK.LISTE (OK.LISTE, TEMP, SLO.OBJET, 1)
26 OSE.FIN.CONTROLE
27
28 !BATCH
29
30 [IHM]
31
32 [FRAME]
33 " OK (O/N) : %%"
34
35 [FRAME_HELP]
36 !SAISIE
37 !FRAME
38 FRAME ECRAN FROM Ecran GP FRAME.AREA 2 TO 21
39 DATA '#' OR '%'
40 DATA.NAMES
41 FIELD.CONTROL
```

FIGURE 5.17 – Interface code affichant le code après ajout

6 Revue du sprint3

La figure ci-dessous représente le graphique d'avancement du sprint 3 :

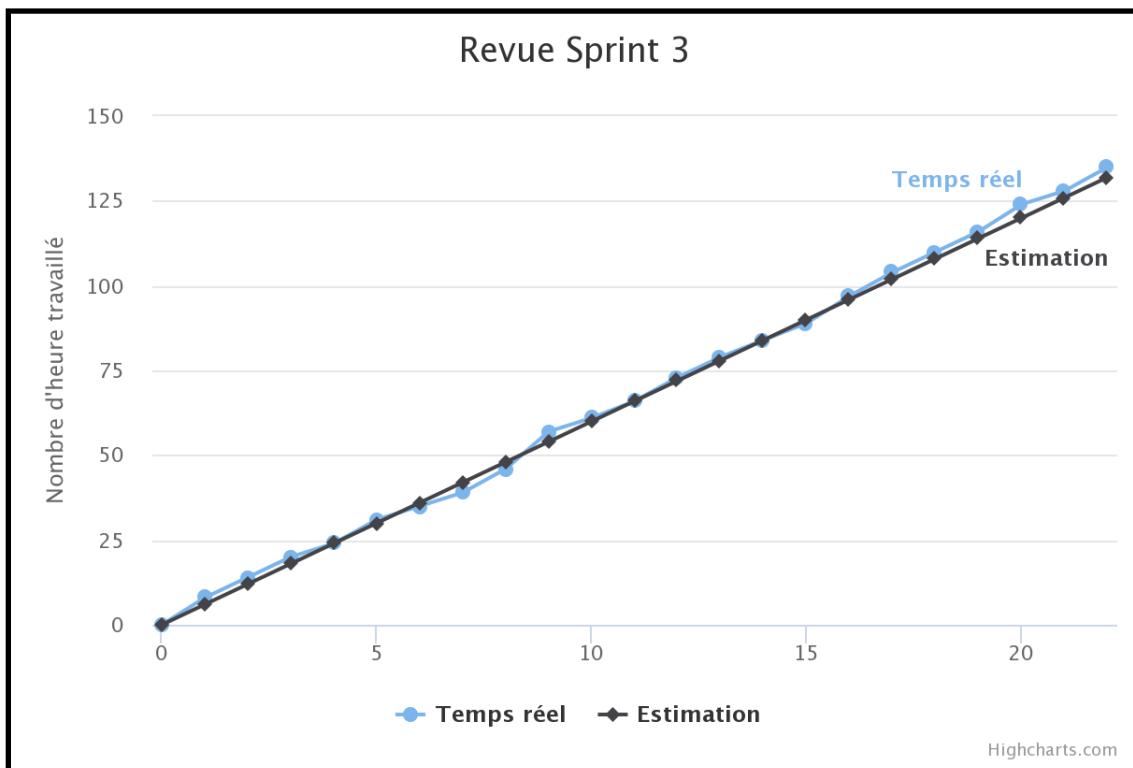


FIGURE 5.18 – Revue Sprint 3

D'après le graphe, la courbe du temps réel (représentée en bleue) est presque confondue avec la droite idéale (représentée en noir).

Cela veut dire que notre estimation en nombre d'heure de travail pour le sprint 3 a été bien fixée et plus correcte que celle des sprint précédents.

7 Conclusion

Dans ce chapitre, nous avons terminé le dernier sprint de notre application en traitant les détails de la réalisation de notre sprint et en montrant les différentes interfaces.

CONCLUSION GÉNÉRALE

Nous avons réalisé une application web qui permet la génération automatique de code ADL à travers une interface graphique simple et facile à utiliser.

Pour réaliser cette application, on s'est appuyé sur Scrum, UML, React JS et Node JS. Il nous a fallu en premier lieu de faire des recherches approfondies dans le produit GP et ses composants, ainsi que le langage de programmation propriétaire de NeoXam : ADL.

Ensuite, nous avons développé et paramétré l'éditeur de code qui est nécessaire pour la manipulation directe du code ADL. Par ailleurs, nous avons développé les squelettes standards ADL des composants et des écrans GP.

Enfin, nous avons développé les fonctions de génération automatique de code.

Cet outil va apporter plein d'avantages aux développeurs sur plusieurs niveaux :

- Gain de temps.
- Diminution des sources d'erreurs.
- Homogénéité du code.

Ce stage nous a été très enrichissant puisqu'il nous a permis de découvrir et d'adopter de nouvelles technologies, de s'auto-former et de mettre en pratique des connaissances théoriques acquises au cours de nombreuses années d'études à l'Institut Supérieur d'Informatique et de les améliorer.

Comme suite à notre travail, nous envisageons d'élaborer un compilateur de code qui permettra de compiler le code ADL généré dans le même environnement de développement afin d'améliorer la fiabilité et la certitude du processus effectué.

RÉFÉRENCES BIBLIOGRAPHIQUES

- [1] Uml : qu'est-ce que c'est? <https://www.futura-sciences.com/tech/definitions/informatique-uml-3979/>, 2018. [En ligne; Accès le 02-25-2022].
- [2] Visual Paradigm. <https://www.educba.com/visual-paradigm/>.
- [3] Citrix. <https://www.citrix.com/fr-fr/products/receiver.html>.
- [4] Trello. <https://outilscollaboratifs.com/2012/01/trello-outil-dorganisation-collaboratif-2/>.
- [5] OverLeaf. <https://bioinfo-fr.net/creez-vos-documents-collaboratifs-latex>.
- [6] Visual Studio. <https://framalibre.org/content/visual-studio-code>.
- [7] Postman. <https://practicalprogramming.fr/postman>.
- [8] GitLab. <https://www.blogdumoderateur.com/tools/gitlab/>.
- [9] PostgreSQL. <https://www.oracle.com/fr/database/definition-postgresql.html>.
- [10] Basarat Ali Syed and Martin Bean. *Beginning Node.js*. Springer, 2014.
- [11] AM Vipul and Prathamesh Sonpatki. *ReactJS by Example-Building Modern Web Applications with React*. Packt Publishing Ltd, 2016.
- [12] ExpressJS. <https://expressjs.com/fr/>.

إنشاء أداة للتوليد التلقائي للبرامج

يتم تقديم هذا المشروع كجزء من تدريينا في نهاية الدراسة للحصول على درجة الرخصة الوطنية في : «علوم الكمبيوتر» يهدف المشروع المسمى «إنشاء أداة للتوليد التلقائي للبرامج» إلى تسهيل مهمة المطورين من خلال إنشاء رمز تلقائياً لهم عبر واجهة رسومية سهلة الاستخدام

كلمات مفاتيح :

ReactJS, NodeJS, ADL , التوليد

Création d'un outil de génération automatique des programmes ADL

Ce projet est présenté dans le cadre de notre stage de fin d'études au sein de l'entreprise «NeoXam Tunisie», pour l'obtention du Diplôme National de Licence en Science de L'informatique : «Computer Science». Le projet a pour objectif de faciliter la tâche des développeurs en leur générant automatiquement du code ADL via une interface graphique.

Mots clés : Automatisation, Génération, ReactJS, NodeJS, ADL

Creation of a tool for automatic generation of ADL programs

This project is presented as part of our end-of-study internship within the company «NeoXam Tunisie», for obtaining the National License Degree in : «Computer Science» . The project aims to facilitate the task of developers by automatically generating ADL code for them via a graphical interface.

Keywords : Automation, Generation, ReactJS, NodeJS, ADL

Intitulé et adresse de l'entreprise

Entreprise : NeoXam Tunisie

Adresse : Immeuble Nida, Pôle Technologique El Ghazala, 2088, Ariana

Tél : 71 116 300

Email : tn.recruitment@neoxam.com