



MEMOIRE

Présenté à

L'Ecole Nationale d'Ingénieurs de Sfax
(Département de Génie Informatique et Mathématiques Appliquées)

En vue de l'obtention

Du Diplôme National d'Ingénieur en *Génie Informatiques*

Par

Nour Elimen Yangui

Conception et développement d'une application de gestion d'une salle de sport

Soutenu le 02 juillet 2021, devant la commission d'examen :

Mme. Affef Mdhaffer

Président

M. Bechir Zalila

Examinateur

Mme. Imene Lehyani

Encadrante académique

M. Riadh Ben Halima

Encadrant académique

M. Ahmed Lehyani

Encadrant industriel

DEDICACES

C'est grâce à Dieu que tout est réalisé, et c'est à lui que je rends grâce.

Le reste n'est que dédicaces.

À la mémoire de mes grands-parents

À mes parents **Abdelmajid** et **Jnaina**

Pour leur soutien et leur amour inconditionnel

Que vous trouvez ici l'expression de ma reconnaissance

Que Dieu vous préserve le bonheur, la santé et une longue vie

À ma sœur **MOUNA**

A laquelle, j'espère une vie pleine de joie et de bonheur.

Elle m'a soutenu dans le pire du temps et n'a jamais cessé de croire en moi

À mon frère **Aymen** et sa femme **Youssra**

Pour leurs conseils et leurs encouragements

À ma petite nièce **EMNA**

À tous ceux qui m'ont aidé, de près ou de loin.

À tous ceux qui me sont chers

Je leur dédie ce travail en témoignage de mon grand amour et ma gratitude infinie.

Nour Elimen Yangui

REMERCIEMENTS

C'est avec un grand plaisir que je réserve ces lignes en signe de gratitude et de profonde reconnaissance à tous ceux qui m'ont aidé à réaliser ce travail.

Je tiens tout d'abord à remercier et exprimer toute ma reconnaissance envers mes deux encadrants académiques madame **Imene Lahyani** et monsieur **Riad Ben Halima** pour leur aide précieuse, leurs encouragements et leurs conseils judicieux.

Ma gratitude va également à monsieur **Ahmed Lehyani**, mon encadrant à DK Soft pour sa disponibilité, son soutien et pour ses encouragements continus qui m'ont guidé durant les différentes phases de ce projet.

Mes vifs remerciements s'adressent également à toute l'équipe de DK Soft pour l'accueil chaleureux, l'aide, le temps alloué et les efforts fournis à nous guider et à nous intégrer dans l'environnement de travail.

Enfin, je saisie cette occasion pour remercier tous les membres du jury qui ont accepté de juger ce rapport de projet de fin de mes études tout en espérant qu'ils y trouvent la qualité, la clarté et motivation qu'ils attendent.

Nour Elimen Yangui

TABLE DES MATIERES

Introduction générale	1
CHAPITRE 1 : ETUDE PREALABLE	2
I. Introduction.....	3
II. Présentation de l'organisme d'accueil	3
III. Présentation générale du projet	3
IV. Étude et critique de l'existant	4
IV.1. Présentation de la solution : LogiSport	4
IV.2. Présentation de la solution: Gym Management	5
V. Solution proposée	6
VI. Méthodologie adoptée et Planification du travail	7
VI.1. Méthodologie adoptée « Scrum ».....	7
VI.2. Les rôles Scrum.....	9
VI.3. Sprint planning	9
VI.1. Outil de Planification du travail	10
VII. Conclusion.....	11
CHAPITRE 2 : SPÉCIFICATION DES BESOINS ET ANALYSE	12
I. Introduction.....	13
II. Spécification des besoins	13
II.1. Définition des acteurs	13
II.2. Identification des besoins fonctionnels	13
II.3. Identification des besoins non fonctionnels.....	15
III. Besoins techniques.....	16
III.1. Environnement matériel.....	16
III.2. Environnement logiciel.....	16
IV. Architecture logique de l'application.....	23
IV.1. Etude comparative des patrons architecturaux.....	24

IV.2.	Architecture interne d'un microservice	25
V.	Conclusion.....	27
CHAPITRE 3 : MISE EN PLACE DE L'ARCHITECTURE MICRO-SERVICE ET INTEGRATION KEYCLOAK.....		28
I.	Introduction.....	29
II.	Sprint1 : Mise en place de l'architecture Microservice	29
II.1.	Backlog du sprint1.....	29
II.2.	Intégration des microservices	29
III.	Sprint2 : Intégration de Keycloak	32
III.1.	Concepts théoriques	32
III.2.	Backlog du sprint 2.....	35
III.3.	Diagramme de cas d'utilisation.....	36
III.4.	Diagramme de séquence	37
III.5.	Réalisation	38
III.5.1.	Configuration de Keycloak	38
III.5.2.	Paramétrage de l'application.....	41
IV.	Conclusion.....	42
CHAPITRE 4 : TRAITEMENT DES PLANIFICATIONS ET DES INSCRIPTIONS ..		43
I.	Introduction.....	44
II.	Sprint3	44
II.1.	Backlog du sprint 3.....	44
II.2.	Diagramme de cas d'utilisation.....	45
II.3.	Diagramme de séquences	48
II.4.	Diagramme de classes	51
II.5.	Réalisation	51
III.	Sprint 4	54
III.1.	Backlog du sprint 4.....	54

III.2.	Diagramme de cas d'utilisation.....	55
III.3.	Diagramme de séquences	58
III.4.	Diagramme de classes	62
III.5.	Réalisation	63
IV.	Conclusion.....	65
CHAPITRE 5 : TRAITEMENT DE LAMISE EN LOCATION DES TERRAINS ET DES FACTURES.....		66
I.	Introduction.....	67
II.	Sprint5	67
II.1.	Backlog Sprint.....	67
II.2.	Diagramme de cas d'utilisation.....	68
II.3.	Diagramme de séquences	71
II.4.	Diagramme de classes	73
II.5.	Réalisation	74
III.	Sprint6	76
III.1.	Backlog du sprint 6.....	76
III.2.	Diagramme de cas d'utilisation.....	77
III.3.	Diagramme de classes	79
III.4.	Diagramme de séquence	81
III.5.	Réalisation	82
IV.	Sprint 7	85
IV.1.	Backlog du sprint 7.....	86
IV.2.	Diagramme de cas d'utilisation.....	86
IV.3.	Conception UML.....	86
IV.4.	Réalisation	87
V.	Conclusion.....	88
Conclusion générale et perspectives.....		89

BIBLIOGRAPHIE	90
---------------------	----

LISTE DE FIGURES

Figure 1. Logo <i>DK Soft</i>	3
Figure 2. Interface de gestion des cours	4
Figure 3. Interface d'ajout d'une planification	5
Figure 4. Interface d'ajout d'un nouveau membre	5
Figure 5. Processus de Scrum	8
Figure 6. Logo <i>Jira Software</i>	11
Figure 7. Logo <i>Visual Studio Code</i>	17
Figure 8. Logo <i>Eclipse IDE</i>	17
Figure 9. Logo <i>Postman</i>	17
Figure 10. Logo <i>HTML5</i>	18
Figure 11. Logo <i>CSS3</i>	18
Figure 12. Logo <i>TypeScript</i>	19
Figure 13. Logo <i>REST</i>	19
Figure 14. Logo <i>MySQL</i>	19
Figure 15. Logo <i>SpringBoot</i>	20
Figure 16. Logo <i>Hibernate</i>	20
Figure 17. Logo <i>Angular</i>	21
Figure 18. Logo <i>Angular Material</i>	21
Figure 19. Logo <i>UML</i>	22
Figure 20. Logo <i>Draw.io</i>	22
Figure 21. Logo <i>bitbucket</i>	23
Figure 22. Logo <i>slack</i>	23
Figure 23. Architecture frontend de l'application	25
Figure 24. Architecture de la partie backend	26
Figure 25. Architecture globale de l'application	30
Figure 26. Dashboard <i>Eureka</i>	31
Figure 27. API Gateway pattern	31
Figure 28. Fonctionnement SSO.....	32
Figure 29. Fonctionnement OAuth2	33
Figure 30. Fonctionnement OpenIdConnect.....	34
Figure 31. Diagramme de cas d'utilisation du sprint 2.....	36

Figure 32. Diagramme de séquence du cas d'utilisation « S'authentifier avec login et mot de passe »	38
Figure 33. Realm « master »	38
Figure 34. Interface de création d'un compte admin	39
Figure 35. Création d'un realm	39
Figure 36. Ajout client « fitness-app ».....	40
Figure 37. Interface d'authentification par défaut	40
Figure 38. Contenu du dossier themes.....	40
Figure 39. Interface Themes dans keycloak	41
Figure 40. Interface d'authentification.....	41
Figure 41. Interface de paramétrage de l'application	42
Figure 42. Diagramme de cas d'utilisation détaillé du sprint 3	46
Figure 43. Diagramme de séquence du cas d'utilisation « créer cours »	48
Figure 44. Diagramme de séquence du cas d'utilisation « planifier quotidiennement ».....	50
Figure 45. Diagramme de classes du sprint 3.....	51
Figure 46. Interface d'ajout d'un nouveau cours.....	52
Figure 47. Interface des planifications	52
Figure 48. Interface de visualisation de informations d'une planification	52
Figure 49. Interface d'ajout d'une nouvelle planification	53
Figure 50. Calendrier des planifications avec filtre cours	53
Figure 51. Diagramme de cas d'utilisation détaillé du sprint 4	55
Figure 52. Diagramme de séquence « Créer pack »	59
Figure 53. Diagramme de séquence « affecter une inscription à un adhérent »	61
Figure 54. Diagramme de classe du sprint 4	62
Figure 55. Interface de visualisation des packs	63
Figure 56. Interface de visualisation de détails d'un pack	63
Figure 57. Interface de gestion des adhérents	64
Figure 58. Interface de visualisation de informations d'un adhérent	64
Figure 59. Interface de visualisation de l'historique d'abonnement d'un adhérent	64
Figure 60. Interface d'ajout d'un nouvel abonnement	65
Figure 61. Diagramme de cas d'utilisation détaillé du sprint 5	68
Figure 62. Diagramme de séquence du cas d'utilisation « créer terrain ».....	71
Figure 63. Diagramme de séquence « affecter une location à un client »	72
Figure 64. Diagramme de classe du sprint 5	73

Figure 65. <i>Interface de gestion des terrains</i>	74
Figure 66. <i>Interface d'ajout d'un nouveau terrain</i>	74
Figure 67. <i>Calendrier des réservations d'un terrain</i>	75
Figure 68. <i>Interface de gestion des réservations de terrain</i>	75
Figure 69. <i>Diagramme de cas d'utilisation détaillé du sprint 6</i>	77
Figure 70. <i>Diagramme de classes résultant</i>	80
Figure 71. <i>Diagramme de séquence du cas d'utilisation « Etablir une facture »</i>	81
Figure 72. <i>Liste des factures</i>	82
Figure 73. <i>Liste des factures avec filtre « état facture » & « compagnie »</i>	82
Figure 74. <i>Interface de création d'une facture</i>	83
Figure 75. <i>Interface de création d'une facture</i>	83
Figure 76. <i>Interface de visualisation d'une facture</i>	84
Figure 77. <i>Interface de paiement d'une facture</i>	85
Figure 78. <i>Interface historique de paiement</i>	85
Figure 79. <i>Diagramme de cas d'utilisation sprint6</i>	86
Figure 80. <i>Interface de visualisation des revenues</i>	87
Figure 81. <i>Interface de consultation des notifications</i>	87
Figure 82. <i>Interface de consultation des statistiques</i>	88

LISTE DES TABLEAUX

Tableau 1. <i>Tableau comparatif des solutions existantes</i>	6
Tableau 2. <i>Planification des sprints</i>	10
Tableau 3. <i>Environnement matériel utilisé</i>	16
Tableau 4. <i>Tableau comparatif des architectures</i>	24
Tableau 5. <i>Backlog du sprint 1</i>	29
Tableau 6. <i>Backlog du sprint 2</i>	35
Tableau 7. <i>Description textuelle du cas d'utilisation « S'authentifier via login et mot de passe »</i>	36
Tableau 8. <i>Backlog du sprint 3</i>	44
Tableau 9. <i>Description textuelle du cas d'utilisation « Créer cours »</i>	46
Tableau 10. <i>Description textuelle du cas d'utilisation « planifier quotidiennement »</i>	47
Tableau 11. <i>Backlog du sprint 4</i>	54
Tableau 12. <i>Description textuelle du cas d'utilisation « Créer pack »</i>	56
Tableau 13. <i>Description textuelle du cas d'utilisation « affecter une inscription à un adhérent »</i>	57
Tableau 14. <i>Backlog du sprint 5</i>	67
Tableau 15. <i>Description textuelle du cas d'utilisation « Créer terrain »</i>	69
Tableau 16. <i>Description textuelle du cas d'utilisation « Affecter une location à un client »</i> ...70	70
Tableau 17. <i>Backlog du sprint 6</i>	76
Tableau 18 : <i>Description textuelle du cas d'utilisation « Etablir une facture »</i>	77
Tableau 19. <i>Description textuelle du cas d'utilisation « Encaisser par espèce »</i>	78
Tableau 20. <i>Backlog du sprint 7</i>	86

LISTE DES ABRÉVIATIONS

REST : Representational State Transfer

HTML : HyperText Markup Language

CSS : Cascading Style Sheets

UML : Unified Modeling Language

SSO : Single Sign On

OIDC : OpenId Connect

IAM : Identity and Access Management

BMI : Body Mass Index

OTP : One Time Password

Introduction générale

Personne ne peut nier que la révolution technologique a grandement servi l'homme dans son environnement et aussi a amélioré de façon considérable son niveau de vie. En effet, toute entreprise est prête à investir des sommes substantielles dans la mise en œuvre des logiciels pour améliorer ses services.

C'est l'exemple des salles de sports qui cherchent à utiliser diverses applications web, mobiles pour aboutir à une meilleure gestion et optimiser la diffusion des informations en interne.

Dans ce contexte, se situe notre contribution par le biais de ce projet de fin d'études proposé par l'entreprise Dk Soft, qui vise à concevoir et développer une application web basée sur l'architecture microservices.

Cette solution assurera la gestion des services de la salle de sport à savoir la gestion des cours, des entraîneurs, la planification, la préparation des abonnements des clients, la mise en location des terrains, la génération des factures et le suivi de leurs paiements.

Le présent rapport décrit en détails la progression du projet s'étalant sur cinq chapitres comme suit : Le premier chapitre décrit l'entreprise d'accueil, la présentation du sujet et la méthodologie de travail adoptée. Le deuxième chapitre présente les spécifications des besoins et analyse qui est la première phase du cycle SCRUM durant laquelle nous préparons notre plan de travail. Le troisième chapitre est consacré à la réalisation des deux premiers sprints relatifs à la mise en place de l'architecture microservices de notre application ainsi que l'intégration de keycloak et le paramétrage de la salle de sport. Le quatrième chapitre est dédié à la réalisation du troisième sprint qui traite la gestion des planifications et le quatrième sprint qui traite la gestion des inscriptions. Le cinquième chapitre sera consacré à la présentation de la réalisation des tâches du cinquième sprint qui consiste à développer la partie liée à la mise en location des terrains, le sixième sprint sera consacré pour la génération des factures et le suivi des paiements et le dernier sprint qui se focalise sur l'établissement des revenus, la gestion des notifications ainsi que la génération des statistiques.

Enfin, nous clôturons notre rapport par une conclusion générale exposant une synthèse du travail ainsi que quelques perspectives.

CHAPITRE 1 : ETUDE PREALABLE

I. Introduction

Dans ce premier chapitre, nous allons donner un aperçu sur le cadre général de notre projet. Nous commençons par une présentation de l'organisme d'accueil. Ensuite, nous abordons l'étude de l'existant. Enfin, nous terminerons ce chapitre par l'énoncé de la méthodologie de gestion de projet adoptée.

II. Présentation de l'organisme d'accueil

DK Soft [1] est une Entreprise de Services du Numérique (ESN), il s'agit d'une structure souple, réactive et qui propose des solutions sur mesure dans le domaine des nouvelles technologies et du système d'information. La figure 1 présente le logo de DK Soft.

DK Soft fournit un développement de logiciels personnalisés et de haute qualité à ses clients, les aidant à accélérer le temps de déploiement, à réduire les coûts et à accroître la compétitivité.



Figure 1. Logo DK Soft

III. Présentation générale du projet

Ce projet intitulé « Conception et développement d'une application de gestion d'une salle de sport » est effectué dans le cadre du projet de fin d'études pour l'obtention du diplôme national d'ingénieur en Génie Informatique.

Ce projet est réalisé chez la société DK Soft au sein de l'équipe JEE qui s'intéresse au développement des applications Web.

Notre projet vise à développer une application web destinée à une salle de sport à base de l'architecture microservice afin d'assurer les réservations des salles, la préparation des

abonnements des clients, la mise en location des terrains, la génération des factures et le suivi de leurs paiements.

Avant de démarrer l'élaboration de la solution nous sommes amenés à faire une étude de l'existant afin d'explorer les solutions existantes.

IV. Étude et critique de l'existant

L'étude de l'existant est une étape primordiale pour la réalisation d'un projet. Cette étape a pour objectif de détecter et d'analyser les points forts et les points faibles des solutions existantes.

Les analyses sur les solutions existantes permettent de bien définir les besoins et les fonctionnalités que nous pouvons intégrer dans notre application.

IV.1. Présentation de la solution : LogiSport

LogiSport [2] solution développée par LogiSam permet d'informatiser la gestion d'une salle de sport. Cette solution offre les fonctionnalités suivantes :

- Gérer les salles (volume, occupation)
- Planning et charge des entraîneurs
- Suivi des adhérents (Informations personnelles, abonnements, cartes d'abonnés).
- Pointage

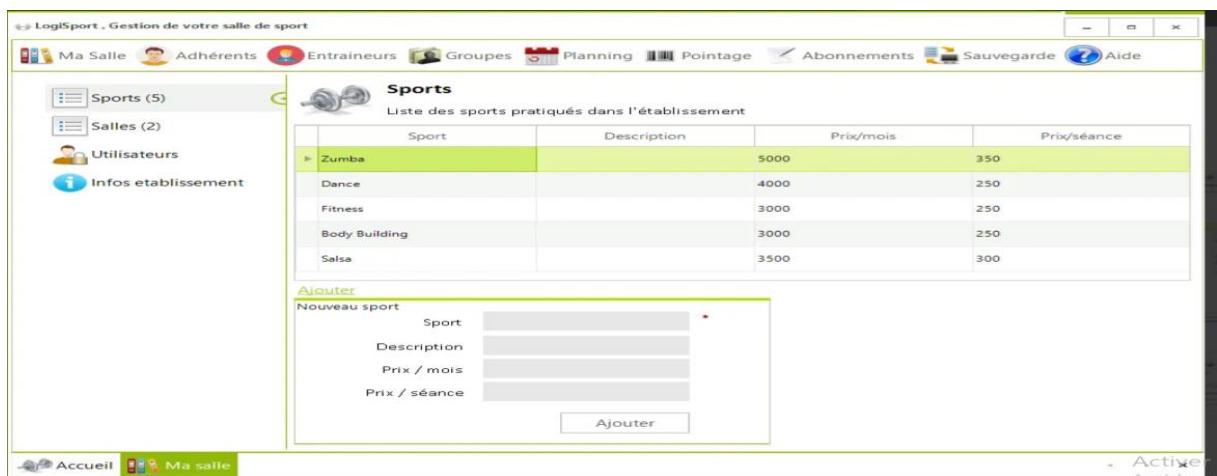


Figure 2. Interface de gestion des cours

IV.2. Présentation de la solution: Gym Management

Gym Management [3] est une solution disponible chez Oddo qui contient les fonctionnalités suivantes :

- Gestion des membres
- Gestion des entraîneurs
- Gestion des exercices
- Gestion des planifications par cours
- Formulaire de calcul de BMI

The screenshot shows the 'Workouts / Workout Plan' section of the Gym Management application. At the top, there are tabs for 'Members', 'Workout', 'Diet', 'Report', and 'Configuration'. On the right, there are user icons and a dropdown for 'Mitchell Admin'. Below the tabs, there are buttons for 'EDIT' and 'CREATE'. The main area displays a 'Workout Plan 1' with days set to Monday, Wednesday, and Friday. A table lists exercises: Dumbbell Chest-Supported Row (Chest, Pec-Deck, 10.00 sets, 2.00 repeats, 15.00 Kgs) and Running (Legs, Treadmill, 0.00 sets, 0.00 repeats, 0.00 Kgs). There are also buttons for 'Action' and '1 / 2'.

Figure 3. Interface d'ajout d'une planification

The screenshot shows the 'Trainers / David' section of the Gym Management application. At the top, there are tabs for 'Members', 'Workout', 'Diet', 'Report', and 'Configuration'. On the right, there are user icons and a dropdown for 'Mitchell Admin'. Below the tabs, there are buttons for 'EDIT' and 'CREATE'. The main area displays a profile for 'David' with an 'Active' status and '0 Members'. It includes tabs for 'Work Information', 'Private Information', and 'HR Settings'. Under 'Contact Information', it shows work address (Ascetic Business Solution LLP, 1725 Slough Ave, Scranton 18540, Pennsylvania PA, India), work location (Scranton), work email (david@odoo.com), and work phone. Under 'Position', it shows department (Ascetic Business Solution LLP), job position (Trainer), job title (Manager), coach (None), working hours (Standard 40 Hours/Week), timezone (Europe/Brussels), and specialist (Body Building).

Figure 4. Interface d'ajout d'un nouveau membre

Nous allons dresser un tableau comparatif entre les solutions existantes comme illustré par le tableau 1.

Tableau 1. Tableau comparatif des solutions existantes

	Gym Management	LogiSport
Gestion des membres	✓	✓
Gestion des entraîneurs	✓	✓
Traitement des planifications	✓	✓
Gestion des salles des cours	✗	✓
Gestion des abonnements par pack	✗	✗
Génération des factures	✗	✗
Suivi de paiement	✗	✗
Génération des recettes	✗	✗
Gestion des compagnies	✗	✗
Gestion du contrôle d'accès	✗	✓
Mise en location des terrains	✗	✗
Génération des statistiques	✓	✗
Payante	✓	✓

Il est bien clair que les solutions Gym Management et LogiSport présentent des fonctionnalités qui facilitent la gestion d'une salle de sport, néanmoins ces solutions sont d'une part payantes d'autre part ils présentent quelques limites et ne peuvent pas atteindre tous les besoins de la salle de sport qui représente notre client tels que la gestion des salles des cours, la gestion des abonnements par pack, la gestion des compagnies, la mise en location des terrains, la génération des factures, le suivi de paiement ainsi et la génération des recettes.

V. Solution proposée

Afin de remédier aux limites des solutions précitées, nous proposons une nouvelle application de gestion d'une salle de sport intitulé « Fitness Access Tracker » qui offre les services suivants :

- Gestion des entraîneurs, cours, salles

- Gestion des adhérents et des compagnies
- Planification des séances de cours
- Accès au calendrier des planifications
- Traitement des inscriptions par pack
- La mise en location des terrains
- Génération des factures et le suivi de leurs paiements
- Visualisation des statistiques

De plus, notre application est basée sur l'architecture microservices qui doit être flexible à toutes améliorations et évolutions. Notre solution est modulaire ce qui facilite l'intégration de nouvelles fonctionnalités dans le futur.

VI. Méthodologie adoptée et Planification du travail

Dans cette partie nous allons présenter tout d'abord la méthodologie adoptée durant notre projet.

VI.1. Méthodologie adoptée « Scrum »

Les processus et les techniques de gestion de projet jouent un rôle crucial dans l'exécution, l'accomplissement des objectifs et aident à organiser le projet de manière structurée et rationalisée.

Pour ce faire, nous avons adopté la méthode SCRUM, faisant partie des méthodes agiles comme processus de développement de notre projet.

Scrum vise à subdiviser l'organisation en des petites équipes auto-organisées et transversales afin de réduire les difficultés telles que : le manque de planification, l'évolution constante des besoins, le manque d'investissement des clients

Les principes de base de Scrum sont les suivants :

- Dégager dans un premier lieu le maximum de fonctionnalités à réaliser pour former le backlog du produit.

- Définir les priorités des fonctionnalités et choisir lesquelles seront réalisées dans chaque itération.
- Par la suite focaliser l'équipe de façon itérative sur l'ensemble de fonctionnalités à réaliser, dans des itérations appelées Sprints.

Ces sprints peuvent durer entre deux à quatre semaines. Une réunion entre le client et l'équipe du projet est programmée à la fin de chaque sprint pour présenter l'état d'avancement, recueillir les commentaires et planifier les prochaines étapes

La figure 5 illustre à la fois les différents rôles de Scrum ainsi que le déroulement du processus.

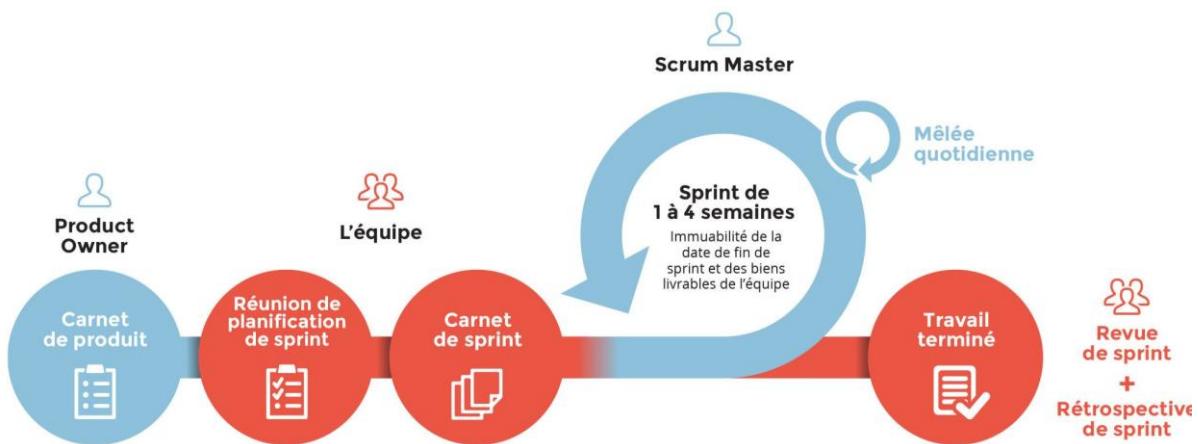


Figure 5. Processus de Scrum

Le choix de Scrum comme une méthodologie de pilotage pour notre projet s'est basé sur les atouts de ce dernier :

- Une souplesse et une flexibilité absolue ce qui est en faveur du chef de projet et toute l'équipe de développement.
- Il impose des réunions quotidiennes qui permettent un suivi continu du développement et l'intervention au bon moment si cela s'avère nécessaire.
- Il encourage la collaboration continue entre les équipes de travail et le partage de connaissances.
- Il utilise une approche itérative et incrémentale pour optimiser la prédictibilité et pour contrôler le risque.
- Sa grande capacité d'adaptation au changement grâce à des itérations courtes, nous permet d'avancer plus rapidement dans la réalisation du projet.

VI.2. Les rôles Scrum

Les acteurs d'un projet scrum [4] sont :

- **Le « Product Owner (PO) » :** est responsable du produit de l'équipe projet client dont :
 - Il porte la vision du produit à réaliser et il s'agit donc généralement d'un expert métier.
 - Il travaille en collaboration directe avec l'équipe de développement. Il est chargé de remplir le « backlog » produit et de déterminer la priorité des user stories à réaliser.
 - Il peut être interne ou externe, même s'il s'agit généralement du client.
- **Le « Scrum Master (SM) » :** C'est lui qui aura pour but de superviser les membres en les aidant, les encourageant, facilitant le partage et la communication des informations essentielles.
- **L'équipe de développement :**
 - Chargée de transformer les besoins exprimés par le « product owner » sous la forme d'user stories en fonctionnalités réelles, opérationnelles et utilisables.
 - Livrer régulièrement une version fonctionnelle du produit.

Notre équipe Scrum est composée de 5 personnes :

- deux stagiaires pour le développement de la plateforme.
- deux encadrants jouant le rôle du Scrum master veillent sur le déroulement du développement comme étant des experts techniques et fonctionnels.
- un client responsable de la spécification des besoins, validation des incrémentations fournis par les développeurs au fur et à mesure du développement et de la planification de l'étape suivante lors des réunions hebdomadaires.

VI.3. Sprint planning

La réunion de planification des sprints est l'évènement le plus important dans SCRUM. Le but de cette réunion est de préparer le planning du travail et d'identifier le Backlog des sprints.

L'un des produits de cette réunion est le choix de la durée des sprints qui diffère selon la complexité du projet et la taille de l'équipe. Pour notre projet nous avons choisi de développer sept sprints d'une durée de deux semaines, comme c'est présenté dans le tableau 2.

Tableau 2. ***Planification des sprints***

Sprint	Description	Date début	Date fin
Sprint 1	Mise en place de l'architecture Microservice	1 Mars	12 Mars
Sprint 2	Authentification (Intégration Keycloak)	15 Mars	26 Mars
Sprint 3	Gestion des planifications	29 Mars	9 Avril
Sprint 4	Développement de la partie liée à la gestion des inscriptions.	12 Avril	23 Avril
Sprint 5	Développement de la partie liée à la mise en locations des terrains.	26 Avril	7 Mai
Sprint 6	Développement de la partie liée à la génération des factures et le suivi des paiements	10 Mai	21 Mai
Sprint 7	Etablir les revenus de la salle de sport Gestion des notifications. Visualisation des statistiques	24 Mai	4 Juin

VI.1. Outil de Planification du travail

Pour bien pratiquer et appliquer la méthodologie Scrum et pour bien gérer les mêlées et les tâches effectuées chaque jour, nous avons utilisé Jira.

Jira Software [5] fait partie d'une gamme de produits conçus pour aider les équipes de tous types à gérer leur travail.

À l'origine, Jira a été pensé comme un outil de suivi des bugs et des tickets. Mais aujourd'hui, c'est devenu un puissant outil de gestion du travail pour toutes sortes de cas d'usage, de la gestion des exigences et des cas de test au développement Agile.



Figure 6. *Logo Jira Software*

VII. Conclusion

Dans ce chapitre, nous avons situé notre projet dans son contexte. En effet, dans un premier lieu, nous avons présenté la société d'accueil. Avant de présenter la solution proposée, nous sommes partis d'une étude critique de quelques systèmes existants sur le marché afin de tenir compte de leurs insuffisances pour optimiser notre réalisation. Finalement, nous avons donné un aperçu sur la méthodologie que nous avons adopté pour la réalisation de ce projet.

Dans le chapitre suivant, nous menons une étude théorique afin d'analyser les besoins et étudier la conception générale de notre application.

CHAPITRE 2 : SPÉCIFICATION DES BESOINS ET ANALYSE

I. Introduction

Dans le présent chapitre, nous commençons par présenter la première phase du cycle de développement du logiciel qui consiste à l'extraction des besoins fonctionnels et non fonctionnels de notre application. Par la suite nous présentons les choix technologiques utilisés durant la réalisation de notre projet. Nous finirons par la présentation de l'architecture globale de la plateforme en menant une étude comparative entre deux architectures différentes.

II. Spécification des besoins

Dans cette section, nous commençons par identifier l'acteur de notre système. Par la suite, nous identifions les besoins fonctionnels et non fonctionnels.

II.1. Définition des acteurs

Un acteur représente une entité externe qui utilise le système. Nous distinguons un seul acteur principal.

L'administrateur : Un administrateur de la salle de sport « Fitness Access Tracker » dont le rôle est de gérer le fonctionnement de la plateforme.

II.2. Identification des besoins fonctionnels

Pour une application, les besoins fonctionnels représentent les contraintes et les conditions qu'elles doivent satisfaire pour avoir un résultat juste et acceptable. En bref, l'application permet à l'administrateur authentifié d'assurer les fonctionnalités suivantes :

- **Paramétrage de la salle de sport**

Il s'agit de la présentation générale de la plateforme (Nom, Logo, Adresse, Numéro de téléphone, Email...).

- **Gestion des cours, des salles, des entraîneurs, des adhérents, des compagnies**
- **Planification des séances de cours**

Choisir le type de planification à suivre. En effet, trois types de planifications se présentent :

- La planification « Une Seule fois »
- La planification de type « Quotidien » en précisant la date début et la date fin
- La planification « Personnalisée » en précisant la date début et la date fin et les jours concernés par cette planification.
- **Intégration du calendrier**
 - L'administrateur peut consulter le calendrier des planifications.
 - Filtrer les planifications par mois, semaine et jour.
 - Filtrer les planifications selon le cours, la salle et l'entraîneur.
- **Inscription à un pack**
 - Un adhérent a la possibilité de s'inscrire à un ou plusieurs packs.
 - Un pack possède plusieurs tarifs selon son type de paiement (mensuel, trimestriel, semestriel, annuel).
 - Consultation de l'historique des abonnements pour chaque adhérent dans la salle de sport « Fitness Access Tracker ».
- **Location des terrains**

Notre application offre la possibilité de louer un terrain de deux manières différentes soit :

- Une seule fois en spécifiant la date et la durée souhaitée
- Location hebdomadaire en spécifiant le jour, le nombre de mois et la durée souhaité
- Un locataire peut être soit une compagnie soit un adhérent (personne physique).

Le service de location est accessible à un adhérent ou bien à une compagnie.

Visualiser les réservations planifiées pour chaque terrain sous forme de calendrier.

- **La génération des factures**
 - Une facture est générée pour un adhérent ou bien pour une compagnie
 - Exporter les informations d'une facture sous format PDF.
- **Paiement des factures**
 - Une facture doit être validée pour se payer
 - Le paiement peut être en espèce ou bien par un ou plusieurs chèques.
 - Possibilité de paiement par facilité
 - Consulter l'historique de paiement pour chaque facture

- **La consultation des notifications de retard**

Recevoir des notifications de retard qui sont lancées automatiquement par le système en cas de dépassement des délais fixés pour le paiement d'une facture.

- **Etablir les Recettes**

Ceci permet d'avoir une vue globale sur les recettes réalisées par la salle de sport pour chaque mois.

- **La consultation des statistiques**

Avoir un tableau de bord et des statistiques permettant de suivre et de visualiser les indicateurs clés de la salle de sport.

II.3. Identification des besoins non fonctionnels

Après avoir identifié les besoins fonctionnels de notre système, des contraintes doivent être prises en compte tout au long de la phase du développement à savoir :

- **Sécurité**

La sécurité est un facteur primordial. Elle englobe des besoins d'établissement de la connexion pour accéder aux services de l'application, des besoins de contrôle du mot de passe et un besoin de déconnexion après une certaine durée d'inactivité.

- **Ergonomie et simplicité**

L'application doit répondre au standard d'ergonomie, et l'interface d'utilisation doit être simple et facile à comprendre et à manipuler.

- **La modularité et l'extensibilité**

Le système doit être modulaire et répond au besoin de l'évolutivité. Ce qui est garanti puisque que nous avons travaillé avec une architecture micro-service. Ainsi nous avons veillé à respecter les bonnes pratiques de développement, ce qui génère un code compréhensible, maintenable, fermé à la modification et ouvert à l'extensibilité.

- **Compatibilité et portabilité**

Une application web quel que soit son domaine, son éditeur et son langage de programmation ne peut être fiable qu'avec une compatibilité avec tous les navigateurs web et tous les moyens que ce soit PC, IPAD ou Mobiles.

III. Besoins techniques

Les besoins techniques sont ceux liés à l'environnement de travail. Pour la réalisation de notre application, nous avons eu recours à plusieurs moyens matériels et logiciels.

III.1. Environnement matériel

Pour mener à bien la réalisation, nous avons utilisé comme environnement matériel, un poste de travail ayant les caractéristiques indiquées dans le tableau 3.

Tableau 3. Environnement matériel utilisé

Caractéristiques	Ordinateur
Marque	DELL
Processeur	Intel Core i7-4510U
Mémoire vive	8 GO
Disque dur	1 TO
Système d'exploitation	Windows® 10 64bit

III.2. Environnement logiciel

Nous détaillons dans cette partie les logiciels et les langages utilisés pour le développement de notre application.

III.2.1. Environnement de développement

- **Visual Studio Code**

Visual Studio Code [6] (VS Code) est un IDE (environnement de développement) open source, multiplateforme développé par Microsoft. Il intègre plusieurs fonctionnalités facilitant le développement comme la coloration syntaxique, le système d'auto complétion IntelliSense,

la gestion du code source avec une intégration simple et puissante de Git, le support de terminal intégré qui permet de sélectionner et d'utiliser le Shell de la plateforme de développement.



Figure 7. Logo Visual Studio Code

- **Eclipse**

Eclipse [7] est un environnement de développement intégré libre extensible, universel et polyvalent, permettant potentiellement de créer des projets de développement mettant en œuvre n'importe quel langage de programmation. Eclipse IDE est principalement écrit en Java (à l'aide de la bibliothèque graphique SWT, d'IBM), et ce langage, grâce à des bibliothèques spécifiques, est également utilisé pour écrire des extensions.



Figure 8. Logo Eclipse IDE

- **Postman**

Postman est un client API [8] populaire qui permet aux développeurs de créer, partager, tester et documenter facilement les API. Pour ce faire, il permet aux utilisateurs de créer et d'enregistrer des requêtes HTTP/s simples et complexes, ainsi que de lire leurs réponses.



Figure 9. Logo Postman

III.2.2. Technologies utilisées

- **HTML 5**

HTML5 pour « HyperText Markup Language 5 », est une version du célèbre format HTML utilisé pour concevoir les sites Internet. C'est un langage de balisage permettant la mise en forme d'une page Web.

HTML5 [9] est une version successeur de la version HTML4.0 Lancée en octobre 2014, cette version a ajouté certaines nouveautés par rapport à la version précédente notamment la possibilité de délimiter le contenu de la page Web en plusieurs sections (Header, Footer, Nav, Main et Aside).



Figure 10. Logo HTML5

- **CSS 3**

CSS (Cascading Style Sheets /feuilles de styles en cascade) [10], servent à mettre en forme des documents web, type page HTML ou XML. Par l'intermédiaire de propriétés d'apparence (couleurs, bordures, polices, etc.) et de placement (largeur, hauteur, côté à côté, dessus-dessous, etc.), le rendu d'une page web peut être intégralement modifié sans aucun code supplémentaire. Les feuilles de styles ont d'ailleurs pour objectif principal de dissocier le contenu de la page de son apparence visuelle.



Figure 11. Logo CSS3

- **TypeScript**

TypeScript [11] est un langage de programmation libre et open source développé par Microsoft .Le code TypeScript est transcompilé en JavaScript, pouvant ainsi être interprété par n'importe quel navigateur web ou moteur JavaScript.



Figure 12. Logo TypeScript

- **REST**

REST (Representational State Transfer) est un protocole [12] qui constitue un style architectural et un mode de communication fréquemment utilisé dans le développement de services Web. C'est un style d'architecture sans état qui est basé sur le protocole http.

Dans son style, les interactions entre clients et services sont améliorées par le recours à un nombre limité d'opérations (verbes) ayant chacune une signification spéciale (GET, POST, PUT et DELETE), pour éviter toute ambiguïté.



Figure 13. Logo REST

- **Système des gestions des bases des données**

MySQL [13] est un système de gestion de bases de données relationnelles (SGBDR). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisées au monde, autant par le grand public (applications web principalement) que par des professionnels.



Figure 14. Logo MySQL

III.2.3. Framework utilisés

- **SpringBoot**

Spring Boot [14] est un framework de développement Java qui facilite le développement d'applications fondées sur Spring en offrant des outils permettant d'obtenir une application packagée en jar, totalement autonome.

C'est un Framework qui a été conçu essentiellement pour développer des architectures à base de microservices. Ceci nous intéresse particulièrement, puisque notre plateforme est découpée en des Microservice.



Figure 15. Logo SpringBoot

- **Hibernate**

C'est un framework [15] open source de type ORM (Object /Relational/ Mapping) qui est concerné par la persistance des données, il permet d'avoir une représentation de la base de données relationnelle en objets Java et vice versa. Il est également une implémentation de la spécification Java Persistence API (JPA).



Figure 16. Logo Hibernate

- **Angular11**

Angular11 [16] est un Framework permettant de créer des applications client en HTML et TypeScript. Les applications web développées par ce Framework sont de type SPA (Simple web application) contenant une seule page web récupérée du serveur dont le contenu change selon le contexte.

- Choix du Framework Angular 11

Pour le développement de la partie frontale de notre application web nous avons choisi d'utiliser le Framework Angular 11:

- Angular permet le développement rapide d'applications responsives et de haut niveau de performance, sécurité et efficacité
- Le code généré sera très bien organisé
- Forte maintenabilité : il est très facile de faire des changements au cours du temps sur le projet grâce à la bonne structuration définie par l'architecture du Framework
- Le contenu se charge dynamiquement et d'une façon très rapide : les allers-retours entre le serveur et le navigateur sont considérablement réduits
- Angular est un Framework multiplateforme
- Simple à utiliser.



Figure 17. Logo Angular

• Angular Material

Angular Material est une bibliothèque de composants d'interface utilisateur [17] pour les développeurs Angular. Les composants d'interface utilisateur réutilisables d'Angular Material aident à créer des pages Web et des applications Web attrayantes, cohérentes et fonctionnelles tout en adhérant aux principes de conception Web modernes telles que la portabilité du navigateur, l'indépendance des appareils.



Figure 18. Logo Angular Material

III.2.4. Langage et outils de conception

- **UML**

UML (Unified Modelling Language) [18] est un moyen pour exprimer des modèles objets en faisant abstraction de leur implémentation, c'est-à-dire que le modèle fourni par UML est valable pour n'importe quel langage de programmation. Il fournit une panoplie d'outils permettant de représenter l'ensemble des éléments du monde objet ainsi que les liens qui les relient.



Figure 19. Logo UML

- **Draw.io**

Draw.io est une application gratuite en ligne [19] qui permet de dessiner des diagrammes ou des organigrammes très simplement. Cet outil permet de concevoir toutes sortes de diagrammes, de dessins vectoriels, de les enregistrer au format XML puis de les exporter dans différents formats disponibles.

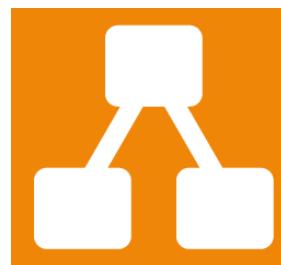


Figure 20. Logo Draw.io

III.2.5. Outils de gestion de projets et de communication

- **Bitbucket**

Pour la gestion du code du projet, nous utilisons Bitbucket, une plateforme de gestion de code projet qui repose sur le système de branches introduit par Git. Il englobe également un espace de gestion de projet, un gestionnaire de planification, un environnement de collaboration

autour du code source, ainsi des fonctions de test et de CI/CD, pour intégration et livraison continues.



Figure 21. Logo bitbucket

- **Slack**

Slack [21] est une plateforme collaborative qui centralise les communications et les outils dans une interface agréable et facile d'utilisation, pour que les membres de l'équipe restent productifs malgré la distance qui les sépare.



Figure 22. Logo slack

IV. Architecture logique de l'application

Dans notre projet nous avons adopté une architecture microservices. Ce choix a été justifié par une étude comparative entre l'architecture monolithique et l'architecture microservice.

IV.1. Etude comparative des patrons architecturaux

Pour bien illustrer les caractéristiques des différentes architectures, nous avons choisi de dresser un tableau comparatif [22] entre deux architectures différentes.

Tableau 4. Tableau comparatif des architectures

Architecture Monolithique	Architecture Micro services
<ul style="list-style-type: none">Le développement est assez simple.Le test est très simple. Il suffit de lancer l'application et de lancer des tests de bout en bout.Tous les composants sont interconnectés et interdépendants.Architecture monolithique n'est pas flexible. Nous ne pouvons pas utiliser différentes technologies. La pile technologique est décidée au début et suivie tout au long.Ce n'est pas fiable. Si une fonctionnalité tombe en panne, toute l'application peut tomber en panne.Les applications monolithiques sont difficiles à faire évoluer une fois qu'elles sont plus grandes.	<ul style="list-style-type: none">Une modification apportée à un élément a moins de risque d'entraîner des changements non prévus au sein d'autres éléments, puisque les modules sont relativement indépendants.Les services sont indépendants et faiblement couplés.Les services n'ont pas besoin de partager la même pile technologique, les mêmes bibliothèques ou les mêmes frameworks. En effet, les choix technologiques peuvent être adaptés aux besoins spécifiques de chaque micro-service.Les services sont responsables de la persistance de leurs propres données ou de leur état externe.Les services peuvent être déployés indépendamment. Une équipe peut mettre à jour un service existant sans reconstruire ni redéployer l'intégralité du système.

✓ Synthèse

Suite à l'étude des critères des deux architectures, nous avons choisi d'adopter l'architecture micro-services pour la mise en place de notre application.

Notre choix est justifié par plusieurs raisons:

- L'extensibilité : Tolérer l'injection d'autres composants selon la demande de nos clients.
- L'utilisation des microservices par des grandes entreprises comme Amazon, Netflix, eBay donne assez de confiance pour que ce style d'architecture soit prêt à être évalué et utilisé par les développeurs d'applications professionnelles.
- La réduction du délai de mise en marché ou « time to market »
- La factorisation et la réutilisation des microservices.

IV.2. Architecture interne d'un microservice

Afin de garantir et respecter les différents critères et caractéristiques d'un micro-service, il fallait penser à son architecture interne. Cette architecture se compose principalement de deux parties :

- **Architecture front-end**

Pour la partie Frontend nous avons utilisé Angular qui propose sa propre architecture basée sur l'utilisation des composants et des services.

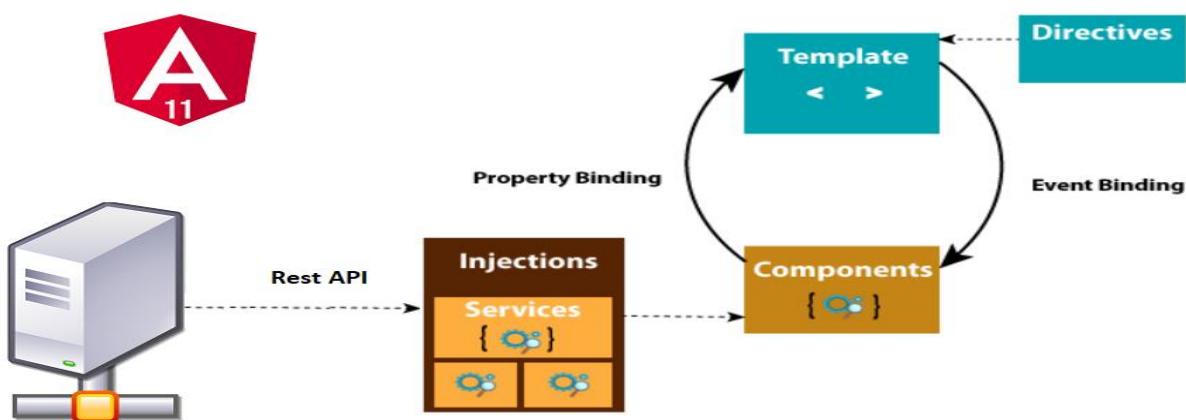


Figure 23. Architecture frontend de l'application

Angular utilise les services pour communiquer avec les API REST en utilisant le protocole HTTP pour récupérer des données de la partie backend. Afin d'insérer dynamiquement les données récupérées dans les vues des composants, Angular définit les techniques de Data Binding qui sont illustrées par la figure 23.

- **Architecture back-end**

Lors de la mise en place de l'architecture de la partie backend nous l'avons partitionnée en quatre sous dossiers ayant chacun une seule responsabilité.

- Persistance

Ce module représente la couche d'accès à la base de données, il contient les entités et les entrepôts de notre application.

- Service

Il contient tous les services métier de l'application. Il est responsable de chaque traitement métier.

- Application

Ce module contient les services web, il est responsable de la communication entre les clients et le serveur web par exposition des API REST à utiliser par le client.

- Common

Il contient les classes, les constantes et les fonctions partagées entre les autres modules (persistance, métier et application).

Dans la Figure 24, nous présentons la disposition des couches aux seins du micro-service tout en indiquant les relations de dépendances entre elles.

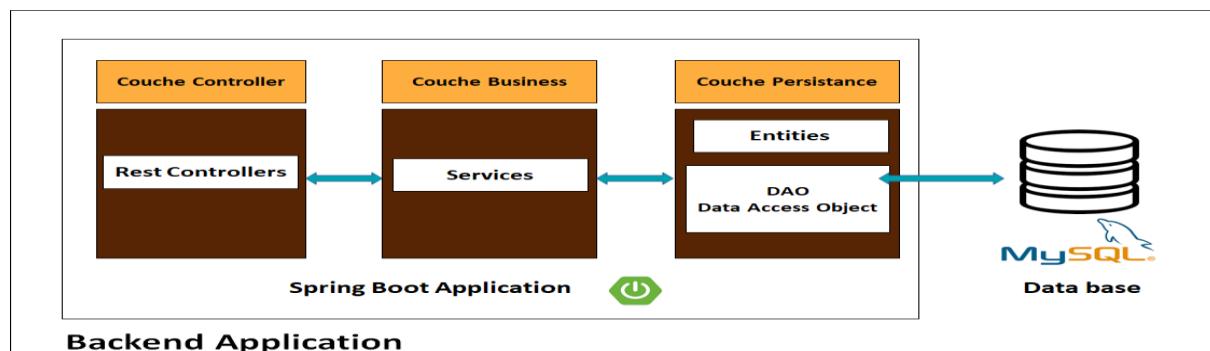


Figure 24. Architecture de la partie backend

V. Conclusion

La phase de spécification des besoins et d'analyse, sert à clarifier et à simplifier les besoins des clients. Dans ce chapitre, nous avons abordé en premier lieu les besoins fonctionnels et non fonctionnels de notre système. Par la suite, nous avons exposé les différentes technologies, Frameworks et logiciels à utiliser durant le développement. Finalement, nous avons essayé de présenter l'architecture adoptée pour notre projet.

En vue de suivre un avancement logique dans ce rapport, une étude du premier et deuxième sprint fera l'objet du prochain chapitre.

CHAPITRE 3 : MISE EN PLACE DE L'ARCHITECTURE MICRO-SERVICE ET INTEGRATION KEYCLOAK

I. Introduction

Après avoir spécifié et analysé les besoins fonctionnels et non fonctionnels, nous entamons la conception et la réalisation des différents modules de l'application.

Dans le présent chapitre, nous réalisons la conception et le développement relatifs au premier et deuxième sprint du backlog produit qui se composent :

- Sprint 1 : Mise en place de l'architecture Microservice
- Sprint 2 : Intégration du Keycloak.

II. Sprint1 : Mise en place de l'architecture Microservice

Dans cette partie nous détaillons le backlog du premier sprint ainsi que les détails de l'intégration de l'architecture microservice de notre projet.

II.1. Backlog du sprint1

Le tableau 5 représente le backlog du premier sprint.

Tableau 5. *Backlog du sprint 1*

Id	Description de la tâche
A.1	Documentation sur l'architecture microservices
A.2	Développement de Eureka, Zuul et Config Server
A.3	Création d'un microservice de test

II.2. Intégration des microservices

Après avoir effectué nos choix technologiques durant le chapitre précédent pour mettre en place notre système, nous pouvons donner maintenant une image globale sur l'architecture cible et son fonctionnement.

La figure 25 présente l'architecture qui résume l'étude technique du fonctionnement de notre application.

L'architecture de la solution est divisée en 3 grandes parties, une première partie Backend contenant les composants de base de l'architecture microservices tels que le service d'enregistrement Eureka, API Gateway Zuul, le serveur de configuration ainsi qu'un service d'authentification Keycloak qui communique avec une deuxième partie Frontend à travers Zuul API Gateway et une troisième partie constituée de deux bases de données, soit une pour chaque microservice.

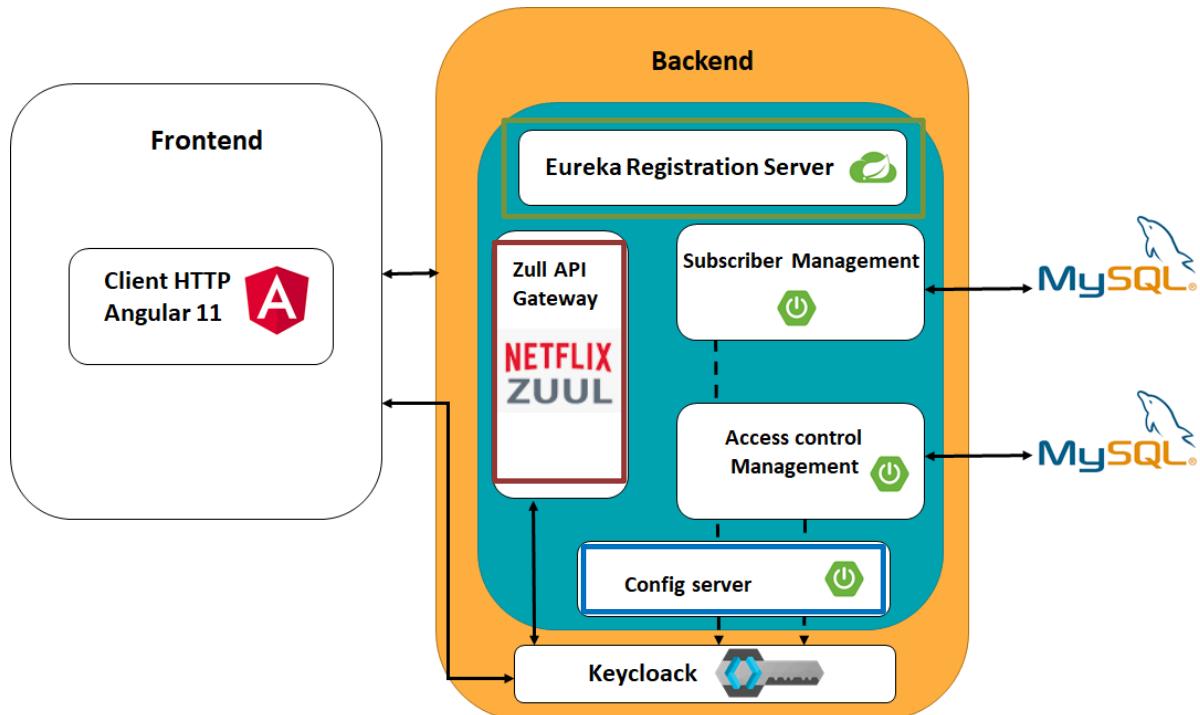


Figure 25. Architecture globale de l'application

Dans cette rubrique, nous allons détailler les composants de base d'une architecture microservices.

- **Service de découverte**

Dans une architecture microservices les services doivent se trouver et communiquer les uns avec les autres. C'est le service de découverte qui gère la façon d'enregistrement et de localisation des microservices déployés. Dans notre architecture nous avons opté pour le service de découverte Eureka de la stack Netflix.

La figure 26 représente le dashboard de l'annuaire de service Eureka, dans lequel nous pouvons visualiser toutes les instances de nos microservices.

The screenshot shows the Spring Eureka dashboard. At the top, it displays 'spring Eureka' and 'HOME LAST 1000 SINCE STARTUP'. Below this, the 'System Status' section shows environment 'test' and data center 'default'. It also lists current time (2021-06-12T11:32:15 +0100), uptime (00:01), lease expiration enabled (false), renew threshold (3), and renew count (0). A red warning message at the bottom of this section states: 'EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.' The 'DS Replicas' section lists three services: GATEWAY-SERVICE, SUBSCRIBERMANAGEMENT-SERVICE, and ACCESSCONTROL-SERVICE, each with one instance in one availability zone and status 'UP'. The 'General Info' section is partially visible.

Figure 26. Dashboard Eureka

- **L'API Gateway Zuul**

Une architecture en microservices implique l'utilisation de plusieurs services qui peuvent changer ou modifier d'emplacement. Cette charge ne doit pas apparaître aux clients.

Pour assurer la transparence entre le partitionnement en microservices et la navigation de client nous avons utilisé le patron API Gateway.

Zuul offre une seule url au client lui permettant de naviguer entre les différents microservices. Afin de router les requêtes aux microservices concernés, Zuul consulte l'annuaire des services Eureka pour connaître les adresses des instances disponibles.

La figure 27 illustre le fonctionnement du proxy.

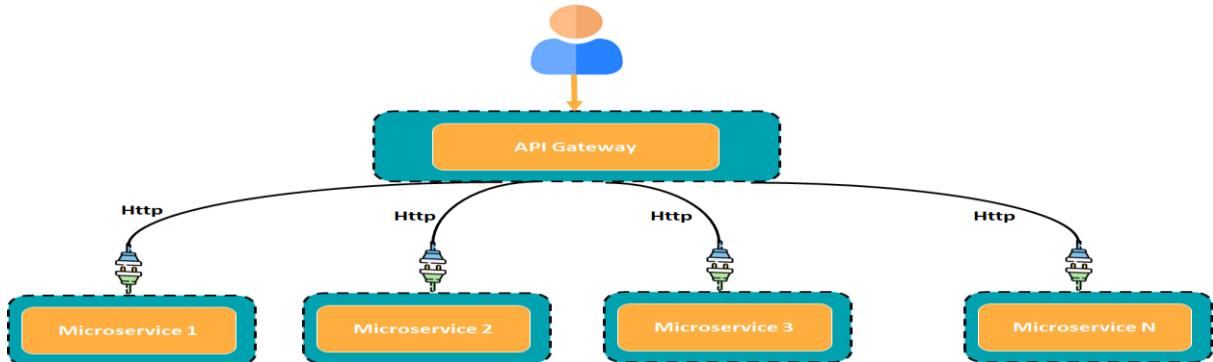


Figure 27. API Gateway pattern

- **Serveur de configuration**

Un serveur de configuration permet de centraliser les configurations du système distribué, dans le but de rendre plus aisée la maintenance des microservices.

- **Communication entre les micro-services**

Les microservices sont autonomes mais ils ont besoin d'échanger des données pour réaliser quelques fonctionnalités. Dans une architecture en microservices, chaque microservice gère ses propres données et ne les partage pas directement avec d'autres services. C'est pourquoi nous avons réalisé des intercommunications synchronisées via des protocoles ouverts. Pour réaliser ces intercommunications, nous avons utilisé RestTemplate.

III. Sprint2 : Intégration de Keycloak

Après la mise en place de l'architecture microservices, nous allons passer maintenant au deuxième sprint qui vise à assurer la sécurité et le paramétrage de l'application.

III.1. Concepts théoriques

Nous allons énoncer quelques concepts théoriques en commençant par l'introduction de la notion SSO.

- **Notion de Single Sign On : l'authentification unique**

Single Sign On [23] désigne un mécanisme d'authentification unique dans lequel un utilisateur accède à plusieurs services par une seule authentification ce qui est montré par la figure 28.

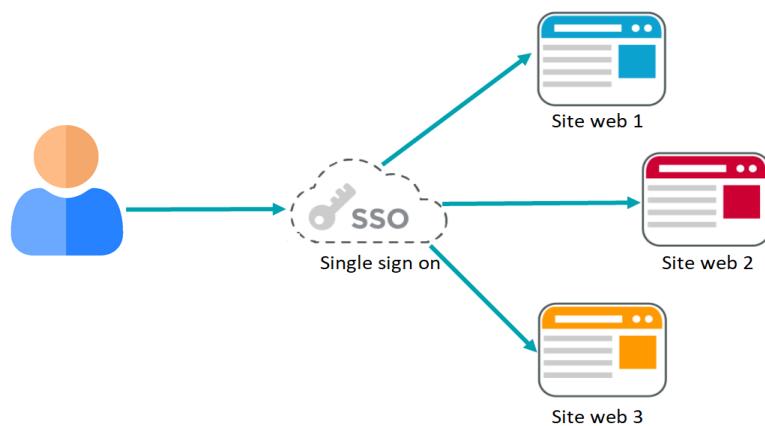


Figure 28. Fonctionnement SSO

Une fois connecté au service de SSO, le service indique à chaque application que l'utilisateur s'avère connecté. Nous pouvons citer comme exemple bien connu de SSO « se connecter avec Facebook » ou « se connecter avec Gmail ».

Une fois connecté à son compte Facebook (son compte Gmail) sur un appareil, l'utilisateur n'a plus besoin de se connecter aux différentes applications une par une.

Parmi les autres solutions Single Sign On les plus populaires, nous pouvons citer les protocoles OAuth et SAML.

Ceci présente plusieurs avantages :

- Une plus grande commodité et un gain de temps aux utilisateurs.
- Réduction des coûts de support : ayant moins de mots de passe à retenir, les utilisateurs requièrent moins d'assistance pour créer, supprimer ou mettre à jour leurs facteurs d'authentification.
- Renforcement de la sécurité : la robustesse des mots de passe individuels peut être améliorée pour mieux résister aux attaques.
- **OAuth2**

OAuth2 [24], est un protocole de délégation d'autorisation. Le but du protocole est de permettre d'autoriser un site web, un logiciel ou une application (dit « consommateur ») à utiliser l'API sécurisée d'un autre site web (dit « fournisseur ») pour le compte d'un utilisateur.

La figure 29 illustre le principe de fonctionnement du protocole OAuth2.

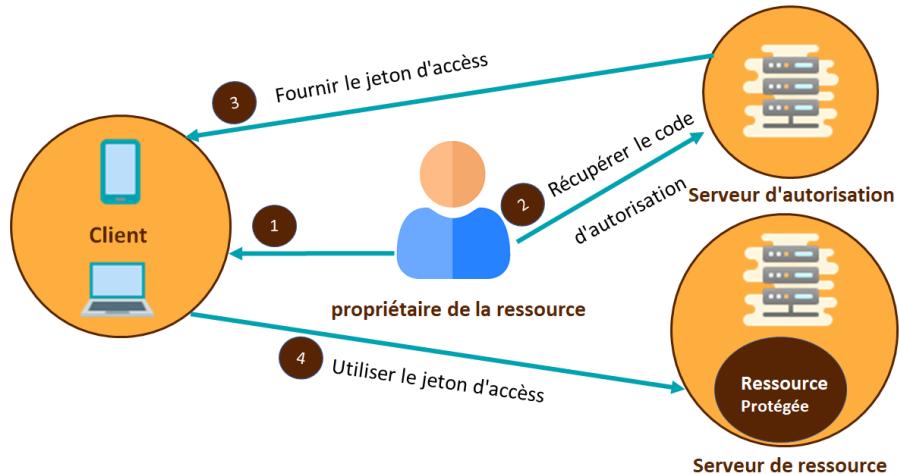


Figure 29. Fonctionnement OAuth2

- **OpenIdConnect**

OpenId Connect(OIDC) est un standard d'identification mis en place par la fondation OpenID, s'appuie sur le protocole OAuth 2.0 en y ajoutant une couche d'authentification.

Ce protocole permet de vérifier l'identité d'un utilisateur auprès d'un serveur d'autorisation afin d'obtenir des informations de profil de base sur l'utilisateur final.

La figure 30 détaille le principe de fonctionnement du protocole OpenId Connect.

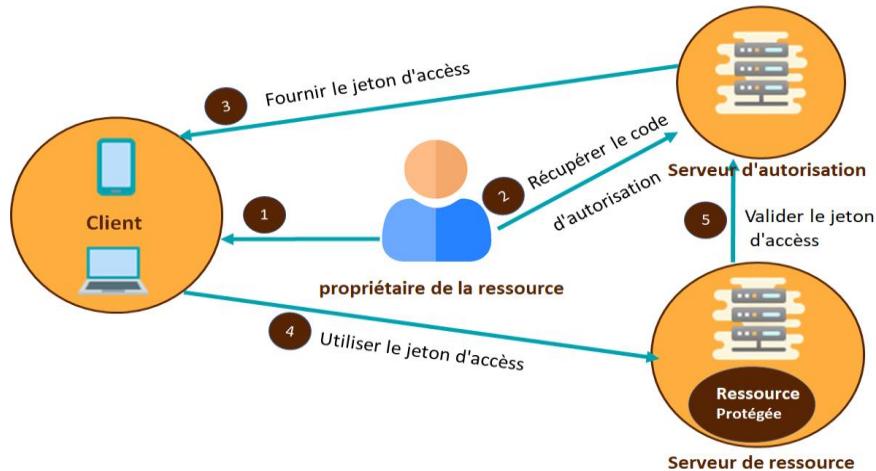


Figure 30. Fonctionnement OpenIdConnect

- **Keycloak**

Keycloak [25]est une solution open source de gestion des identités et des accès destinés aux applications et services modernes. Cette solution d'IAM a été lancée le 10 septembre 2014, sous licence Apache License 2.0, et est portée par la communauté JBoss sous la responsabilité de Red Hat.

Il permet aux utilisateurs de s'authentifier via Keycloak plutôt que d'utiliser des applications individuelles. Cela signifie que leurs applications n'ont pas à gérer les formulaires de connexion, à authentifier les utilisateurs ou à stocker les utilisateurs.

Ainsi, La déconnexion unique permet aux utilisateurs de se déconnecter une seule fois pour être déconnectés de toutes les applications qui utilisent Keycloak.

- **Choix Keycloak**

Keycloak implémente en standard les protocoles (OpenID Connect, OAuth 2.0, et SAML). Il facilite la mise en place de fonctionnalités comme :

- Authentification et déconnexion uniques (Single-Sign On)
- Connection via les réseaux sociaux
- Les mots de passe à usage unique (OTP)
- Gestion des comptes utilisateurs via la console Web et l'API REST,
- Autorisation précise pour différents services.

III.2. Backlog du sprint 2

Le tableau 6 représente le backlog du deuxième sprint contenant les tâches à accomplir.

Tableau 6. Backlog du sprint 2

Id	Nom Story	User Story
B.1	En tant qu'administrateur, je veux m'authentifier l'application.	En tant qu'administrateur, je veux m'authentifier via un login et un mot de passe.
		En tant qu'administrateur, je veux m'authentifier via Facebook.
		En tant qu'administrateur, je veux m'authentifier via Google.
B.2	En tant qu'administrateur, je veux paramétrier l'application.	En tant qu'administrateur, je veux faire les paramètres généraux de l'application.
		En tant qu'administrateur, je veux faire le paramétrage de la facture (tva, timbre fiscal).

Pour avoir une vision globale du comportement fonctionnel des user stories de ce sprint, nous avons choisi le diagramme de cas d'utilisation.

III.3. Diagramme de cas d'utilisation

Les diagrammes de cas d'utilisation [26] permettent une meilleure représentation des interactions entre les acteurs du système et le système lui-même. Il regroupe l'ensemble des fonctionnalités que doit fournir le système.

Le diagramme illustré par la figure 31 résume les fonctionnalités que l'administrateur peut avoir après ce sprint.

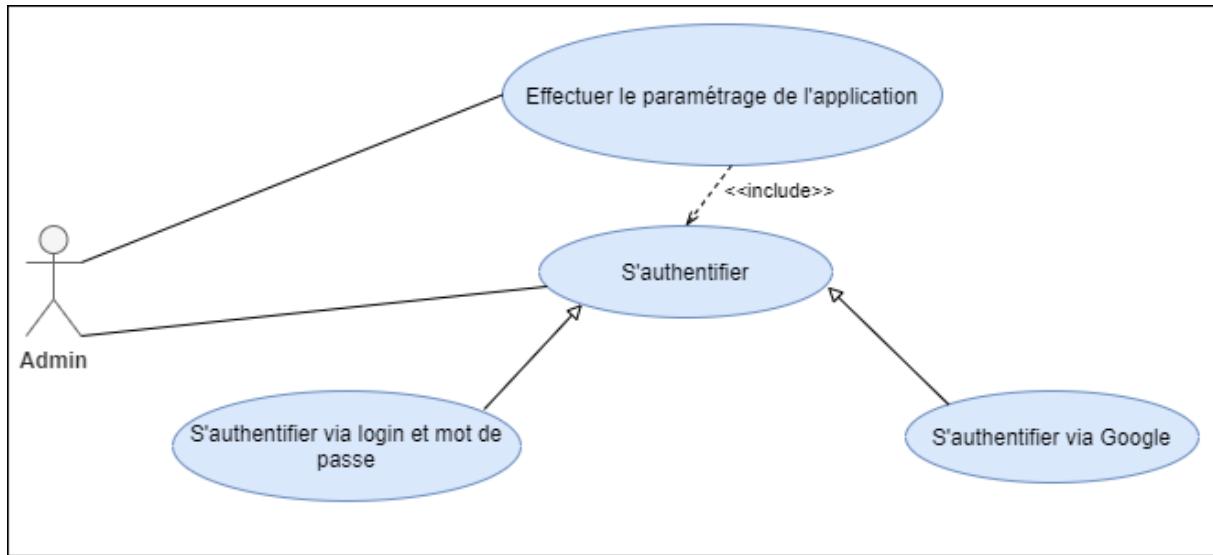


Figure 31. Diagramme de cas d'utilisation du sprint 2

La description textuelle du cas d'utilisation « S'authentifier via login et mot de passe » est détaillée au niveau du tableau 7.

Tableau 7. Description textuelle du cas d'utilisation « S'authentifier via login et mot de passe »

Sommaire	
Titre	Authentification via login et mot de passe
Acteur	Administrateur
Résumé	L'administrateur effectue l'authentification via login et mot de passe.
Description de l'enchaînements	
Pré-condition	L'administrateur doit avoir un compte dans Keycloak.
Post-condition	L'administrateur peut accéder à l'application.

Scénario nominal	<ol style="list-style-type: none"> 1- L'administrateur affiche l'interface « authentification ». 2- L'administrateur remplit les champs nécessaires pour s'authentifier (nom d'utilisateur et le mot de passe). 3- L'administrateur clique sur le bouton « S'authentifier ». 4- Le système vérifie la cohérence des données saisis 5- Le système redirige l'administrateur vers la page d'accueil.
Scénario alternatif	<p>A1 : Champs vides</p> <ol style="list-style-type: none"> 1- Le système affiche un message d'erreur « email requis » ou « mot de passe requis ». 2- Le scénario reprend par 2 <p>A2 : email ou mot de passe erroné :</p> <ol style="list-style-type: none"> 1- Le système affiche un message d'erreur 2- Le scénario nominal reprend au point 2.

III.4. Diagramme de séquence

Les diagrammes de séquences [27] illustrent l'aspect dynamique d'un système. En effet les diagrammes de séquences montrent comment communiquent les objets entre eux pour réaliser une certaine fonctionnalité en mettant l'accent sur la chronologie des messages envoyés.

L'étape de l'authentification, assurée par l'administrateur s'avère une phase primordiale pour accéder à l'application. Pour ce faire, l'administrateur est appelé à saisir son login et son mot de passe. Par la suite, le système vérifie la validité des données saisies. Si les données sont valides, le système redirige l'administrateur au tableau de bord sinon le système affiche un message d'erreur.

Le diagramme de séquence du cas d'utilisation « S'authentifier via login et mot de passe » est représenté par la figure 32.

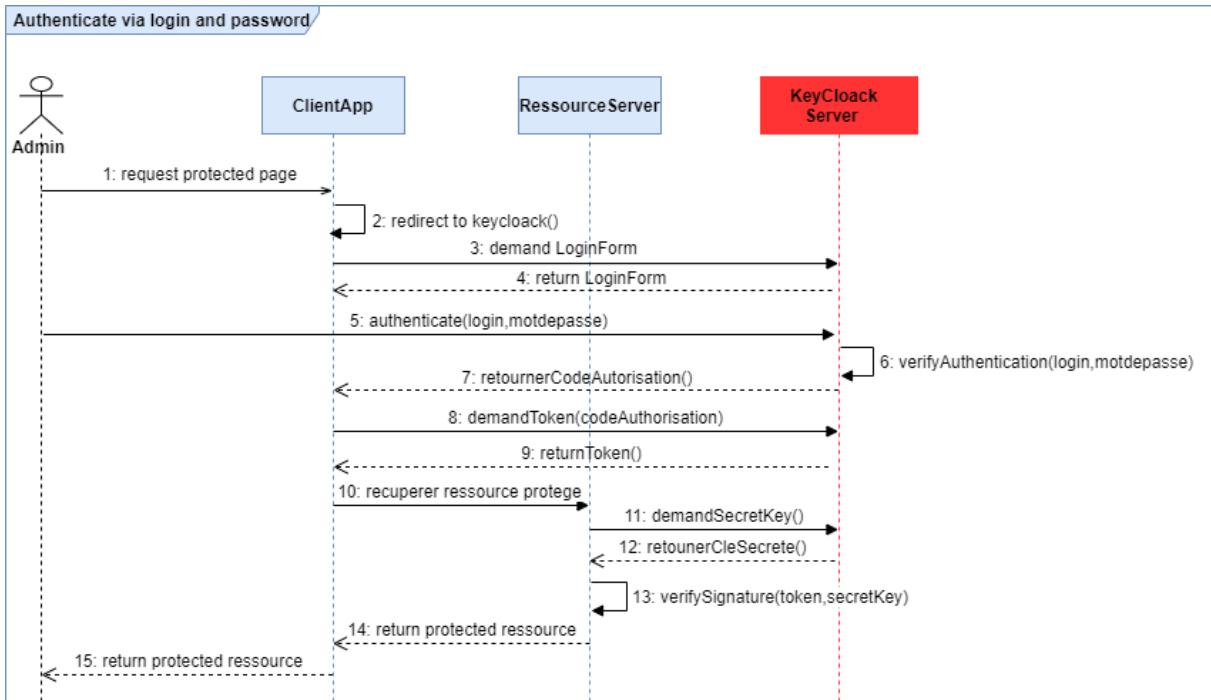


Figure 32. Diagramme de séquence du cas d'utilisation « S'authentifier avec login et mot de passe »

III.5. Réalisation

Dans cette section, nous allons présenter les étapes de configuration de keycloak ainsi que le paramétrage de l'application.

III.5.1. Configuration de Keycloak

Lors du démarrage du serveur Keycloak, par défaut un domaine principal appelé « Master » sera initialisé ceci est illustré par la figure 33.



Figure 33. Realm « master »

- **Création compte admin**

Pour pouvoir configurer Keycloak, il faut tout d'abord créer un compte admin via la console d'administration comme indique dans la figure 34.

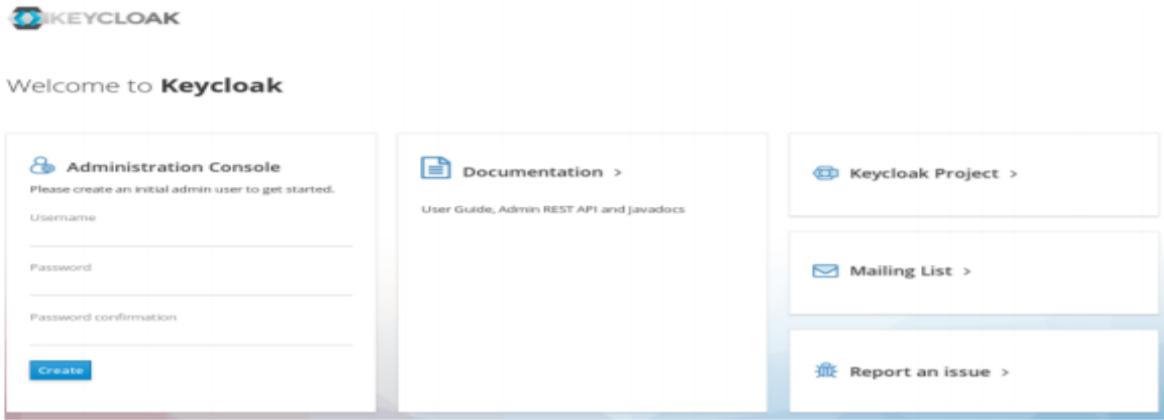


Figure 34. Interface de création d'un compte admin

- **Création d'un domaine/realm Keycloak**

Realm est un concept dans Keycloak qui fait référence à un objet gérant un ensemble d'utilisateurs ainsi que leurs informations d'identification, leurs rôles et leurs groupes. En effet un utilisateur dans Keycloak n'appartient qu'à un seul domaine.

La figure 35 représentée illustre la création d'un realm « FitnessAccessTracker ».

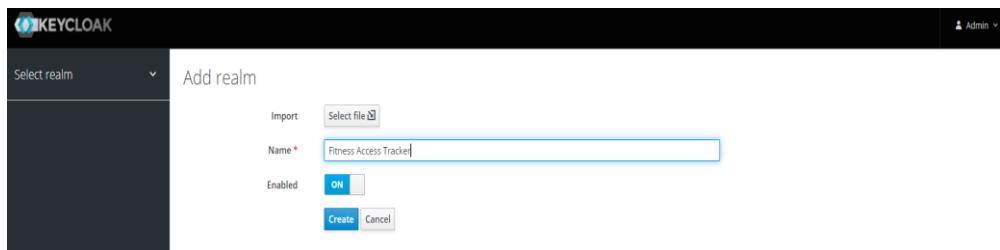


Figure 35. Crédit d'un realm

- **Ajout d'un client**

Dans la figure 36 nous présentons l'ajout d'un client « fitness-app » qui représente notre application.

Add Client

Import

Client ID *

Client Protocol

Root URL

Figure 36. Ajout client « fitness-app »

La figure 37 présente l'interface d'authentification par défaut fournit par keycloak.

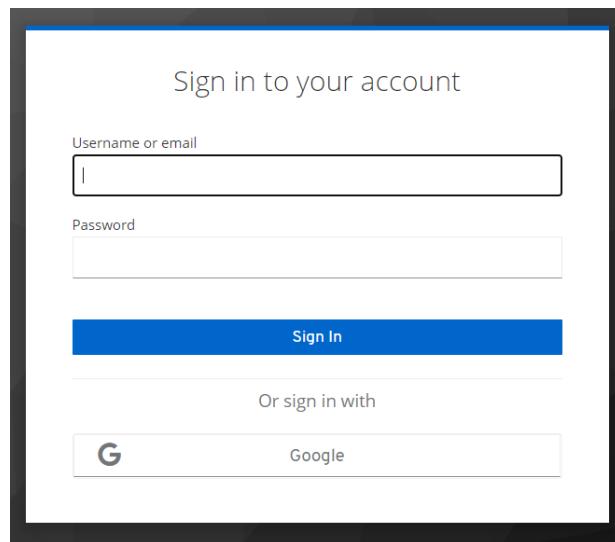


Figure 37. Interface d'authentification par défaut

Nous allons montrer par les figures 38 et 39 les étapes réalisées afin de changer le style de la page de connexion.

stage_pfe > gestionadherents-back-front > keycloak-13.0.0 > themes

Nom	Modifié le	Type	Taille
base	06/05/2021 11:15	Dossier de fichiers	
dksoft	08/06/2021 15:33	Dossier de fichiers	
keycloak	06/05/2021 11:15	Dossier de fichiers	
keycloak.v2	06/05/2021 11:15	Dossier de fichiers	
README	06/05/2021 11:15	Document texte	1 Ko

Figure 38. Contenu du dossier themes

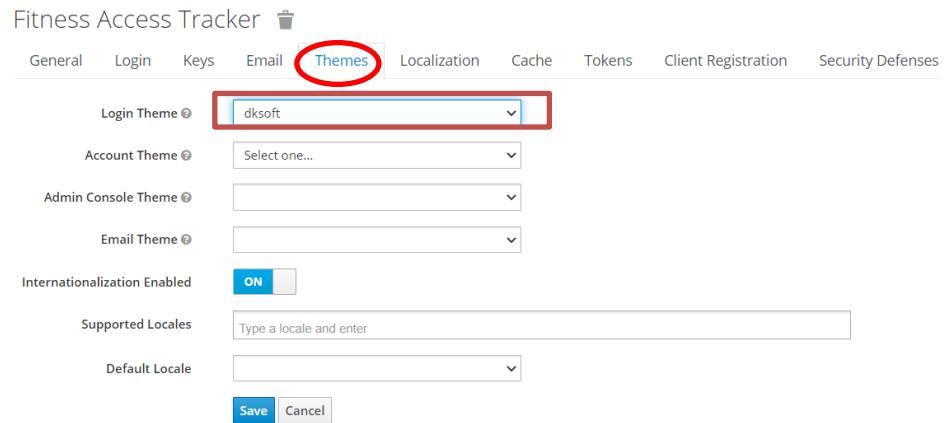


Figure 39. Interface Themes dans keycloak

Le résultat est illustré par la figure 40. Pour se connecter il faut saisir « Nom d'utilisateur ou courriel » et le « mot de passe » et cliquer sur le bouton « Se connecter » ou bien choisir de se connecter via les réseaux sociaux « Google » ou « Facebook ».

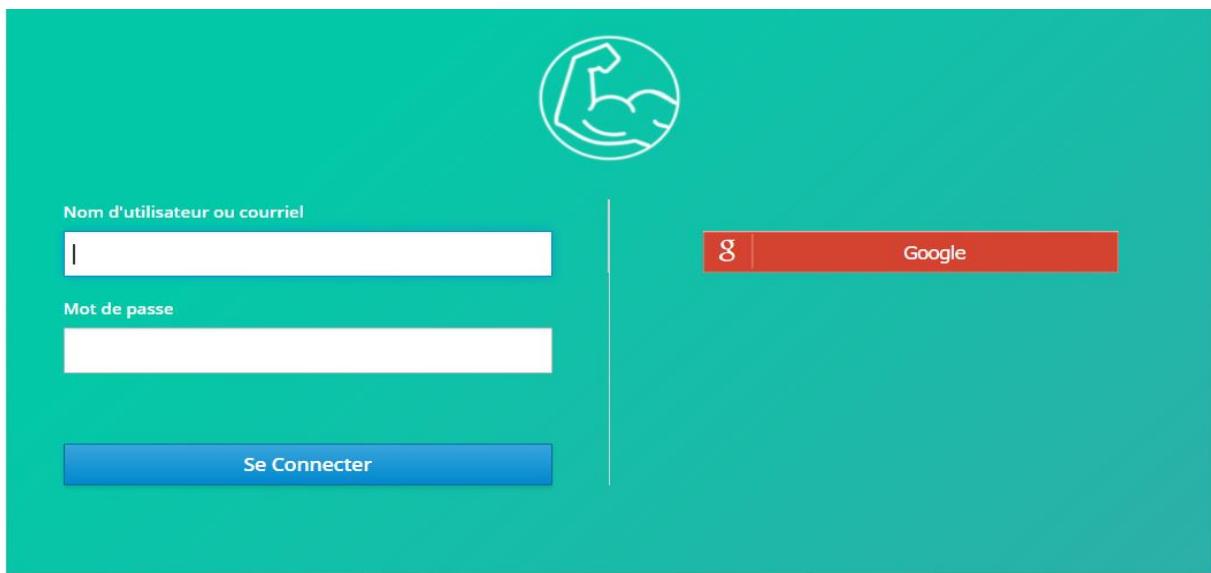


Figure 40. Interface d'authentification

III.5.2. Paramétrage de l'application

A travers l'interface représenté par la figure 41 l'administrateur peut modifier les paramètres généraux ainsi que les paramètres liés à la facture de l'application.

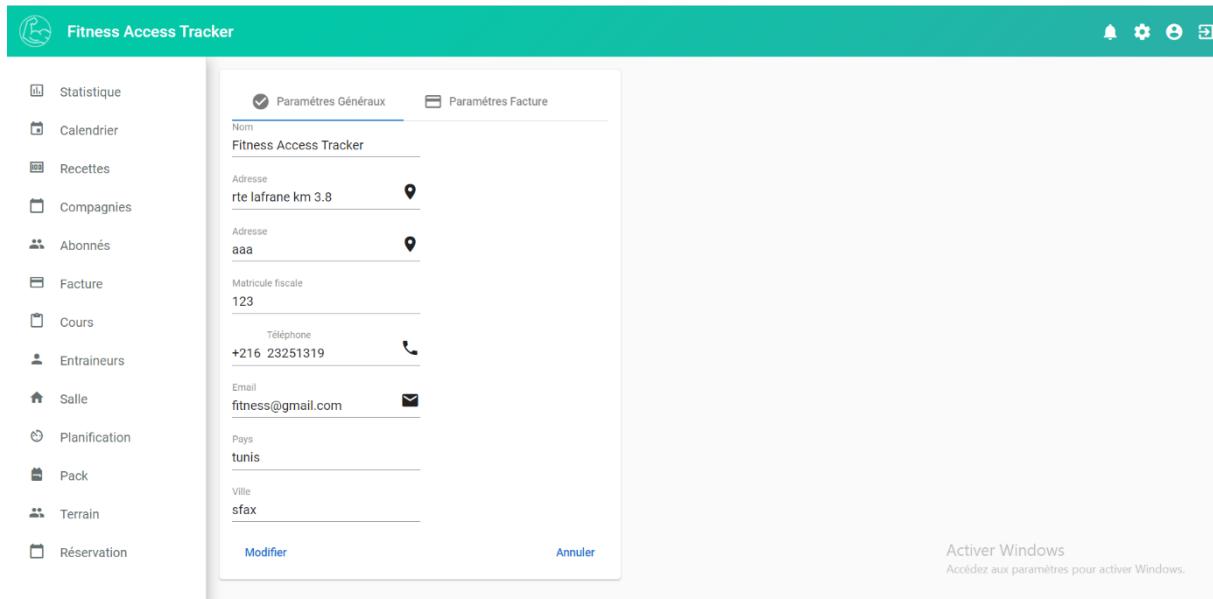


Figure 41. Interface de paramétrage de l'application

IV. Conclusion

Dans ce chapitre nous avons mis en place l'architecture microservices ainsi que l'intégration de keycloak pour assurer la sécurité de notre application. Dans le chapitre suivant nous allons aborder le troisième et le quatrième sprint.

CHAPITRE 4 : TRAITEMENT DES PLANIFICATIONS ET DES INSCRIPTIONS

I. Introduction

Ce chapitre porte sur la présentation du troisième et quatrième sprint. Le troisième sprint traite la gestion des planifications. Le quatrième sprint porte sur la gestion des inscriptions.

II. Sprint3

Dans cette partie nous détaillons en premier lieu le backlog produit du troisième sprint. Après, nous présenterons la partie conception et la réalisation de notre travail illustré par des captures d'écran.

II.1. Backlog du sprint 3

Le tableau 8 représente l'ensemble des récits utilisateurs réalisées durant ce sprint.

Tableau 8. Backlog du sprint 3

Id	Nom Story	User Story
C.1	En tant qu'administrateur, je veux gérer les salles.	En tant qu'administrateur, je veux ajouter une salle.
		En tant qu'administrateur, je veux modifier les informations d'une salle
		En tant qu'administrateur, je veux supprimer une salle
		En tant qu'administrateur, je veux consulter la liste des salles.
C.2	En tant qu'administrateur, je veux créer les cours.	En tant qu'administrateur, je veux consulter la liste des cours
		En tant qu'administrateur, je veux ajouter un cours.
		En tant qu'administrateur, je veux rechercher un cours
		En tant qu'administrateur, je veux modifier un cours.
		En tant qu'administrateur, je veux supprimer un cours
C.3	En tant qu'administrateur, je veux gérer les entraîneurs.	En tant qu'administrateur, je veux consulter la liste des entraîneurs.
		En tant qu'administrateur, je veux filtrer la liste des entraîneurs.

		En tant qu'administrateur, je veux ajouter un entraîneur
		En tant qu'administrateur, je veux modifier les informations d'un entraîneur.
		En tant qu'administrateur, je veux supprimer un entraîneur.
		En tant qu'administrateur, je veux voir les détails d'un entraîneur.
C.4	En tant qu'administrateur, je veux planifier une séance de cours.	En tant qu'administrateur, je veux planifier une séance de cours.
		En tant qu'administrateur, je veux modifier les informations d'une planification
		En tant qu'administrateur, je veux filtrer le calendrier des planifications selon le cours.
		En tant qu'administrateur, je veux filtrer le calendrier des planifications selon l'entraîneur
		En tant qu'administrateur, je veux filtrer le calendrier des planifications selon la salle
		En tant qu'administrateur, je veux supprimer une planification.

II.2. Diagramme de cas d'utilisation

La figure 42 résume les fonctionnalités que l'administrateur peut l'avoir après ce sprint. Le diagramme de cas d'utilisation montre qu'une planification possède trois formes.

Une planification peut être effectuée une seule fois (planification pour une journée), quotidiennement (chaque jour entre la date début et la date fin) ou bien d'une manière personnalisée c'est-à-dire en choisissant les jours concernés par cette planification.

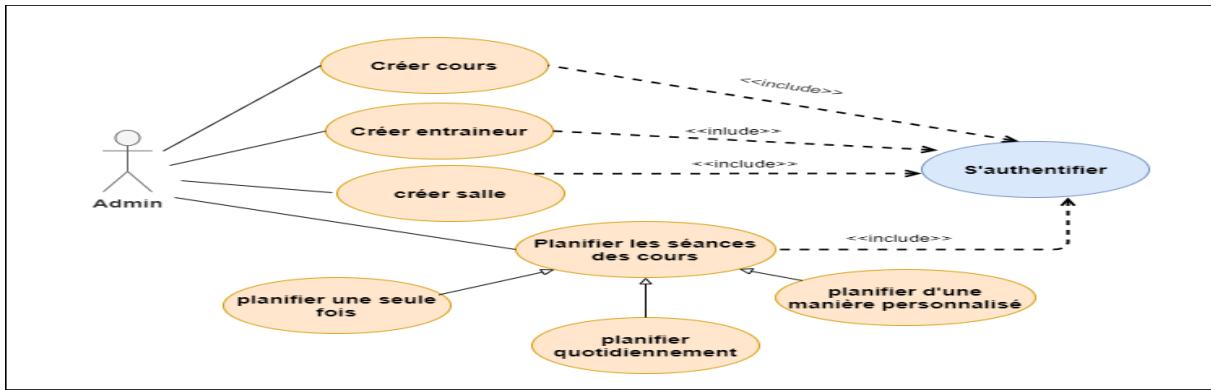


Figure 42. Diagramme de cas d'utilisation détaillé du sprint 3

Le tableau 9 résume la description textuelle de cas d'utilisation « Créer cours ».

Tableau 9. Description textuelle du cas d'utilisation « Créer cours »

Sommaire	
Titre	Créer Cours
Acteur	Administrateur
Résumé	L'administrateur ajoute un nouveau cours.
Description de l'enchaînements	
Pré-condition	L'administrateur doit être authentifié. L'administrateur doit accéder à l'interface des cours
Post-condition	Afficher la liste des cours
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton ajouter nouveau cours. 2. Le système affiche le formulaire de l'ajout, l'administrateur saisie le nom du nouveau cours, choisit la couleur et puis clique sur le bouton « Ajouter ». 3. Le système vérifie l'existence d'un cours avec le même nom saisi. 4. Le système enregistre les données et affiche un message de succès.

Scénario alternatif	<p>A1 : L'administrateur ne remplit pas le champ « nom » du formulaire:</p> <ol style="list-style-type: none"> 1. Le système affiche « nom obligatoire » <p>A2 : Le nom du cours existe déjà :</p> <ol style="list-style-type: none"> 1. Le système affiche une alerte « Cours existe déjà ! ». 2. Le scénario nominal reprend au point 2
----------------------------	--

Le tableau 10 résume la description textuelle de cas d'utilisation « planifier quotidiennement ».

Tableau 10. *Description textuelle du cas d'utilisation « planifier quotidiennement »*

Sommaire	
Titre	Planifier quotidiennement
Acteur	Administrateur
Résumé	L'administrateur ajoute une nouvelle planification quotidienne
Description de l'enchaînements	
Pré-condition	<p>L'administrateur doit être authentifié.</p> <p>L'administrateur doit accéder à l'interface des planifications</p>
Post-condition	Afficher la liste des planifications
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton ajouter. 2. Le système affiche le formulaire de l'ajout, l'administrateur choisit la salle, le cours associé à cette planification, l'entraîneur adéquat, ainsi il choisit le type de planification « quotidienne » ainsi que la date début et fin de cette planification. 3. Le système vérifie l'existence d'une planification dans la salle choisie. 4. Le système enregistre les données et affiche un message de succès.

Scénario alternatif	<p>A1 : L'administrateur ne remplit pas les champs obligatoires du formulaire:</p> <ol style="list-style-type: none"> 1. Le système affiche « champs obligatoire » <p>A2 : Une planification dans la salle choisie existe déjà :</p> <ol style="list-style-type: none"> 1. Le système affiche une alerte. 2. Le scénario nominal reprend au point 2
----------------------------	--

II.3. Diagramme de séquences

Le diagramme de séquence illustré par la figure 43 correspond au cas d'utilisation « Crée cours »

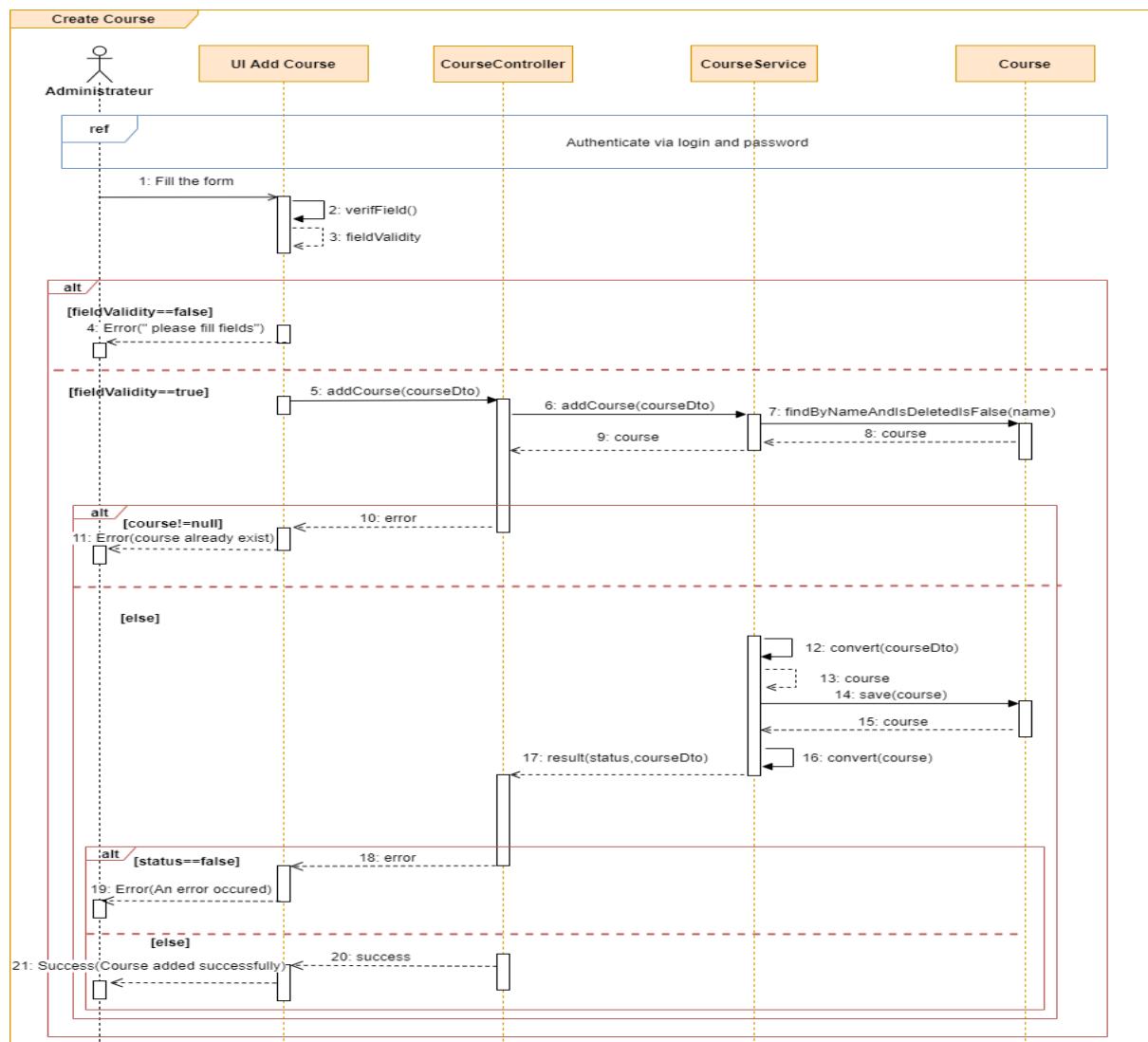


Figure 43. Diagramme de séquence du cas d'utilisation « créer cours »

Le diagramme de séquence illustré par la figure 44 correspond au cas d'utilisation « planifier quotidiennement »

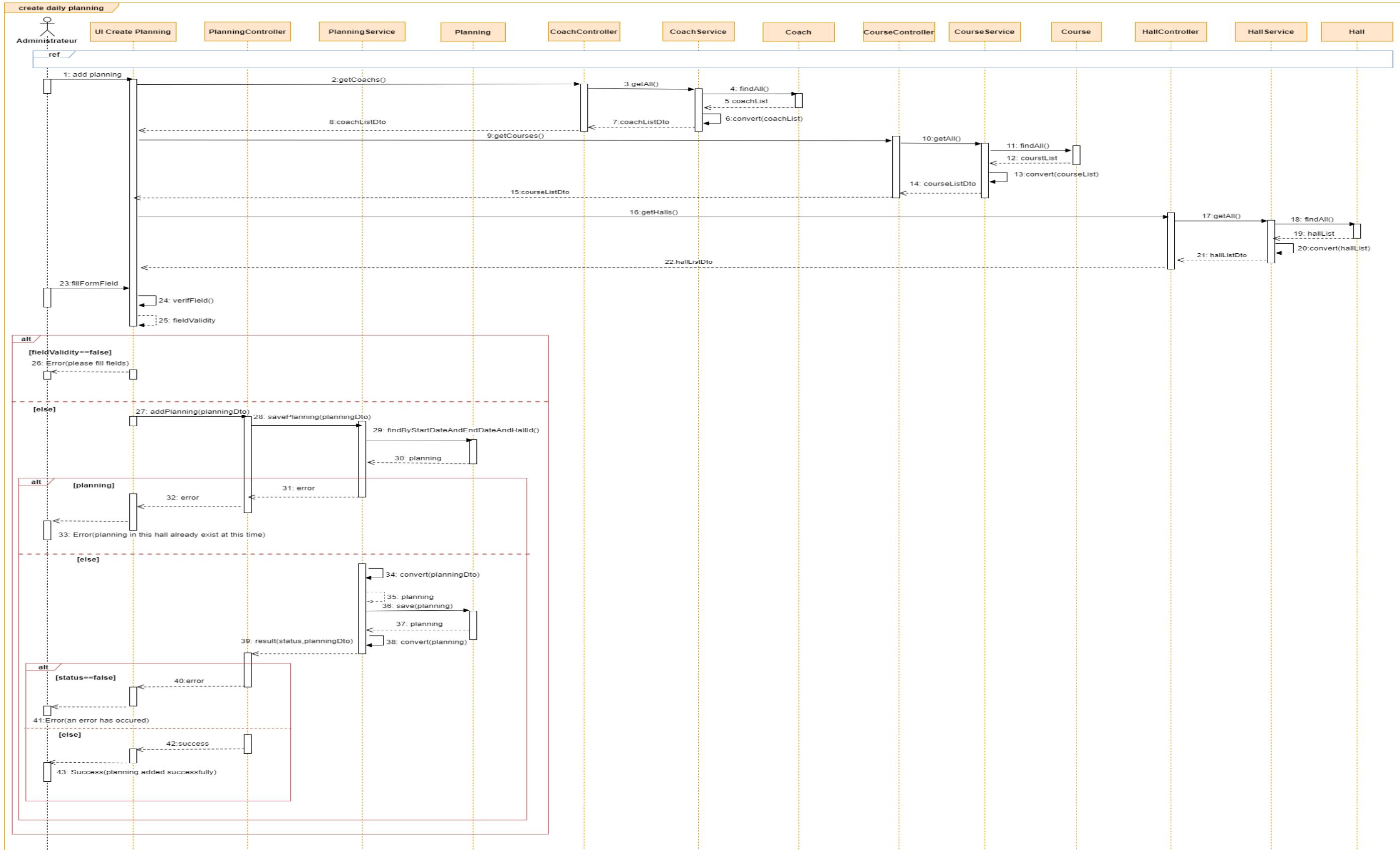


Figure 44. Diagramme de séquence du cas d'utilisation « planifier quotidiennement »

II.4. Diagramme de classes

Le diagramme de classes [25] est un diagramme d'un aspect statique dans le standard UML permettant de modéliser les entités du système ainsi que les relations établies entre elles.

Nous avons utilisé des couleurs différentes pour présenter le diagramme de classe. Chaque couleur représente les classes relatives à un sprint bien déterminé.

Le diagramme de classes du troisième sprint est représenté dans la figure 45.

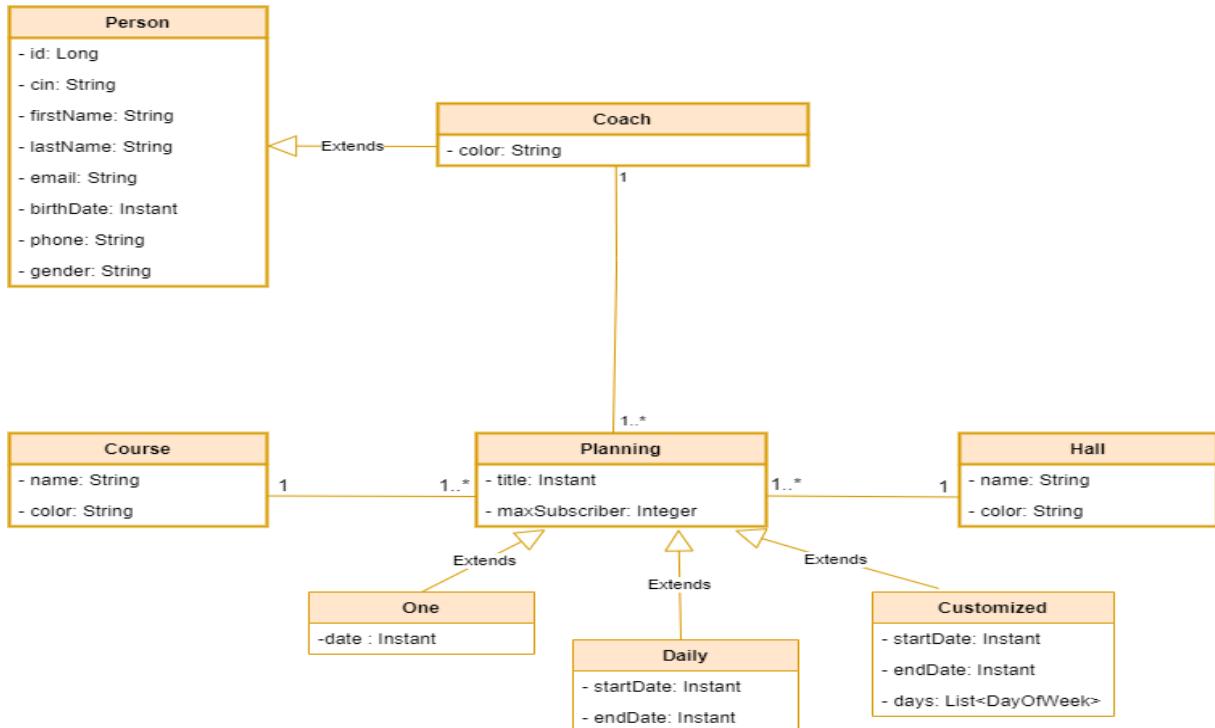


Figure 45. Diagramme de classes du sprint 3

II.5. Réalisation

Dans cette partie, nous présentons les interfaces du troisième sprint afin de mieux comprendre la réalisation des différentes tâches de ce sprint.

Pour l'ajout d'un nouveau cours, l'administrateur doit entrer le nom du cours par conséquent le système va vérifier si ce cours existe déjà si c'était le cas un message d'erreur sera affiché comme indiqué par la figure 46 sinon le cours va être enregistré avec succès.

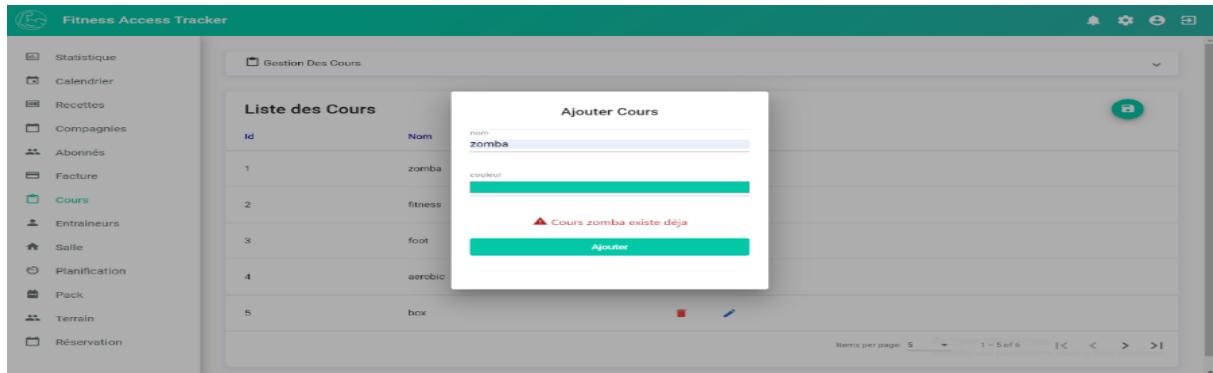


Figure 46. Interface d'ajout d'un nouveau cours

La figure 47 représente l'interface de gestion des planifications. C'est à travers cette interface que l'administrateur peut ajouter, modifier supprimer, rechercher et voir les informations d'une planification spécifique.

En cliquant sur l'icône œil, une fenêtre contenant les informations d'une planification sera affichée comme montre la figure 49.

Gestion des planifications					
Liste des Créneaux					
ID	Titre	Type de planification	Date Début	Date Fin	actions
1	creneau1	Une seule fois	16 mai 2021	16 mai 2021	
2	creneau2	Quotidien	3 avr. 2021	28 mai 2021	
3	creneau3	Personnalisé	16 mai 2021	20 mai 2021	

Figure 47. Interface des planifications

	creneau1	Une seule fois : Samedi
Entraîneur	Nombre maximal adhérents	Cours Salle
yanguirada	11	foot Salle6
Date Début	Date Fin	Heure Début Heure Fin
16 mai 2021	16 mai 2021	05:48 06:48

Figure 48. Interface de visualisation de informations d'une planification

Pour l'ajout d'une nouvelle planification, l'administrateur doit choisir l'entraîneur, le cours et la salle concernée et continue le remplissage du formulaire en choisissant le type de planification la date de début et la date fin du planning.

Par conséquent le système va vérifier si la salle est disponible ou non pendant le créneau entré s'il est indisponible un message d'erreur sera affiché comme indiqué par la figure 50 sinon une nouvelle planification sera enregistrée.

Figure 49. Interface d'ajout d'une nouvelle planification

Le planning c'est un calendrier qui contient toutes les planifications des cours. Par défaut les événements sont regroupés par cours comme illustré par la figure 50 l'administrateur peut filtrer ainsi les planifications selon la salle ou l'entraîneur.

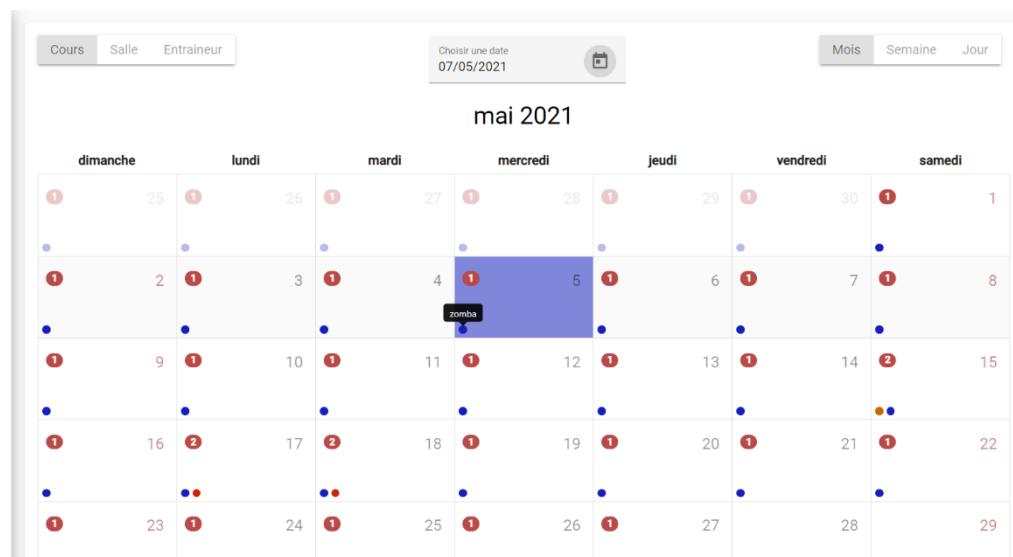


Figure 50. Calendrier des planifications avec filtre cours

III. Sprint 4

Après avoir développé la partie responsable à la gestion des planifications, nous passons maintenant au quatrième sprint en présentant le backlog produit, le diagramme de cas d'utilisation, la conception et la réalisation.

III.1. Backlog du sprint 4

Le tableau 11 représente l'ensemble des récits utilisateurs réalisés durant ce sprint.

Tableau 11. Backlog du sprint 4

Id	Nom Story	User Story
D.1	En tant qu'administrateur, je veux gérer les Compagnies.	En tant qu'administrateur, je veux consulter la liste des compagnies.
		En tant qu'administrateur, je veux ajouter une compagnie.
		En tant qu'administrateur, je veux modifier les informations d'une compagnie.
		En tant qu'administrateur, je veux rechercher une compagnie
		En tant qu'administrateur, je veux supprimer une compagnie.
D.2	En tant qu'administrateur, je veux gérer les adhérents.	En tant qu'administrateur, je veux consulter la liste des adhérents.
		En tant qu'administrateur, je veux ajouter un adhérent.
		En tant qu'administrateur, je veux modifier les informations d'un adhérent.
		En tant qu'administrateur, je veux voir les détails d'un adhérent.
		En tant qu'administrateur, je veux supprimer un adhérent.

D.3	En tant qu'administrateur, je veux gérer les packs.	En tant qu'administrateur, je veux ajouter un pack en spécifiant les différents types d'inscription selon la période(mensuel,trimestriel,semestriel,annuel).
		En tant qu'administrateur, je veux modifier les informations d'un pack.
		En tant qu'administrateur, je veux consulter les détails d'un pack.
		En tant qu'administrateur, je veux supprimer un pack.
D.4	En tant qu'administrateur, je veux affecter une inscription à un abonné.	En tant qu'administrateur, je veux consulter les détails d'une inscription.
		En tant qu'administrateur, je veux affecter une inscription à un abonné en choisissant un pack et le type d'inscription adéquat (mensuel,trimestriel...).
		En tant qu'administrateur, je veux modifier les informations d'une inscription.
		En tant qu'administrateur, je veux supprimer une inscription.

III.2. Diagramme de cas d'utilisation

La figure 51 illustre les diagrammes de cas d'utilisation détaillés du quatrième sprint.

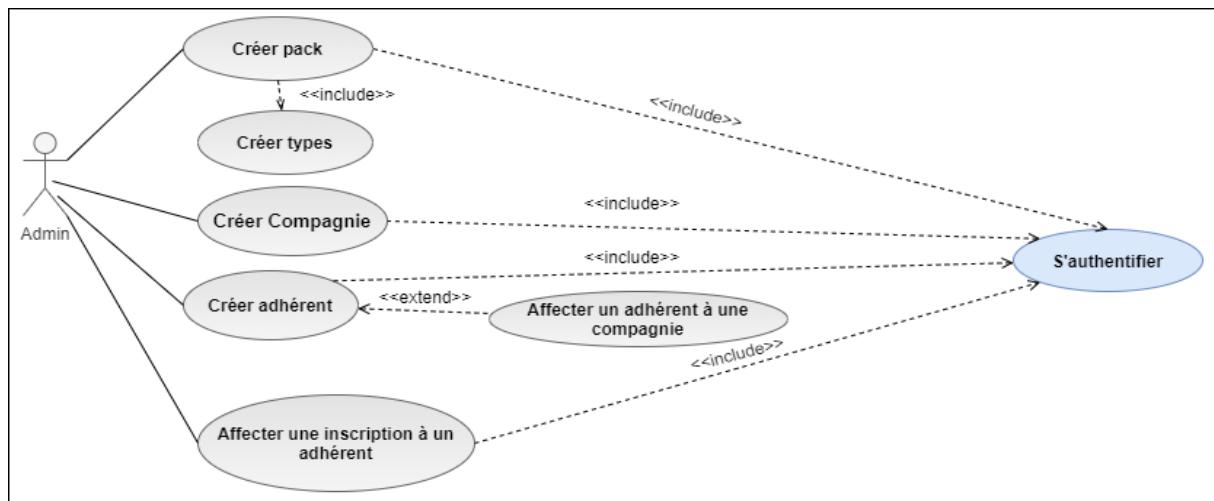


Figure 51. Diagramme de cas d'utilisation détaillé du sprint 4

Une description détaillée du cas d'utilisation « Créer pack » est donnée par le tableau 12.

Tableau 12. *Description textuelle du cas d'utilisation « Créer pack »*

Sommaire	
Titre	Créer pack
Acteur	Administrateur
Résumé	L'administrateur ajoute un nouveau pack avec les différents types d'inscriptions.
Description de l'enchaînements	
Pré-condition	L'administrateur doit être authentifié.
Post-condition	Afficher la liste des packs.
Scénario nominal	<ol style="list-style-type: none"> 1- L'administrateur clique sur le bouton ajouter pack. 2- Le système affiche le formulaire d'ajout. 3- L'administrateur sélectionner une ou plusieurs planifications, saisit un ou plusieurs tarifs (intitulé, période, prix). 4- Le système enregistre les données et affiche un message de succès.
Scénario alternatif	A1 : L'administrateur ne remplit pas les champs obligatoires du formulaire: 1- Le système affiche « champs obligatoire »

Une description détaillée du cas d'utilisation « Affecter une inscription à un adhérent » est donnée par le tableau 13.

Tableau 13. Description textuelle du cas d'utilisation « affecter une inscription à un adhérent »

Sommaire	
Titre	Affecter une inscription à un utilisateur
Acteur	Administrateur
Résumé	L'administrateur ajoute une nouvelle inscription.
Description de l'enchaînements	
Pré-condition	<p>L'administrateur doit être authentifié.</p> <p>L'utilisateur est inscrit au système.</p>
Post-condition	Afficher la liste des abonnements
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton ajouter nouveau abonnement. 2. Le système affiche le formulaire d'ajout. 3. L'administrateur choisit le pack, le type d'inscription (mensuel, trimestriel, semestriel...) et indique la date de début de l'inscription. 4. L'administrateur clique sur le bouton enregistrer. 5. Le système vérifie les données saisies 6. Le système enregistre les données et affiche la liste des inscriptions.
Scénario alternatif	<p>A1 : L'administrateur ne remplit pas les champs obligatoires du formulaire:</p> <ol style="list-style-type: none"> 1. Le système affiche « champs obligatoire » 2. Le scénario nominal reprend au point 3 <p>A2 : L'adhérent est déjà abonné au pack choisi :</p> <ol style="list-style-type: none"> 1. Le système affiche un message en indiquant que cet utilisateur est déjà abonné à ce pack. 2. Le scénario nominal reprend au point 3

III.3. Diagramme de séquences

Dans cette partie nous allons présenter les diagrammes de séquence système de quelques cas d'utilisation classés plus importants. La figure 52 montre le diagramme de séquence du cas d'utilisation « Créer pack ».

Dans un premier temps, l'administrateur est appelé à sélectionner les planifications associées à ce pack par la suite il va entrer un ou plusieurs types d'inscriptions contenant les informations suivantes (nom, période, prix).

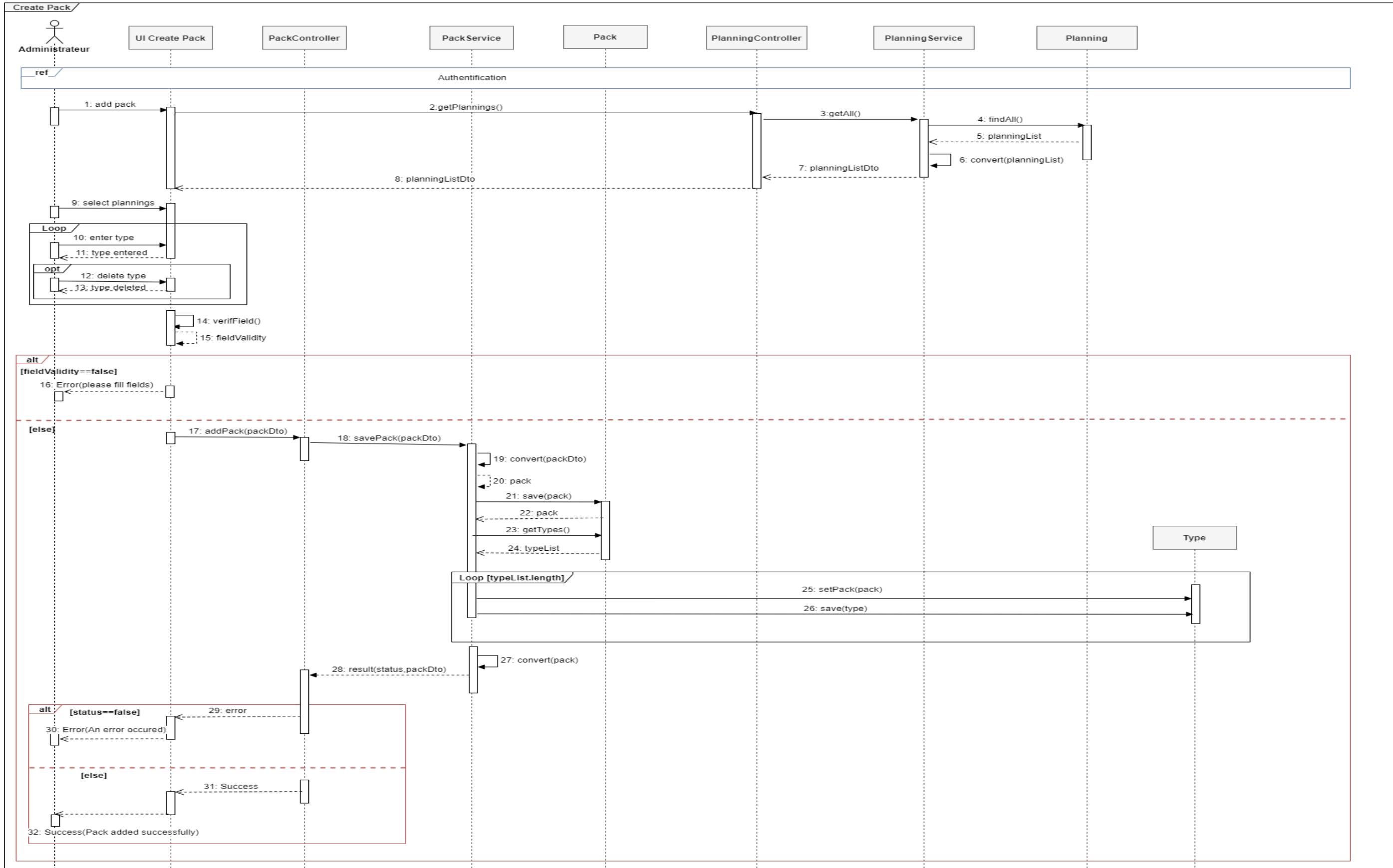


Figure 52. Diagramme de séquence « Crée pack »

La figure 53 résume l'interaction entre les classes qui interviennent lors de l'affectation d'une inscription à un adhérent. En effet, pour un adhérent sélectionné, nous pouvons lui affecter une inscription en choisissant le pack adéquat ainsi le type d'inscription (mensuel, trimestriel...).

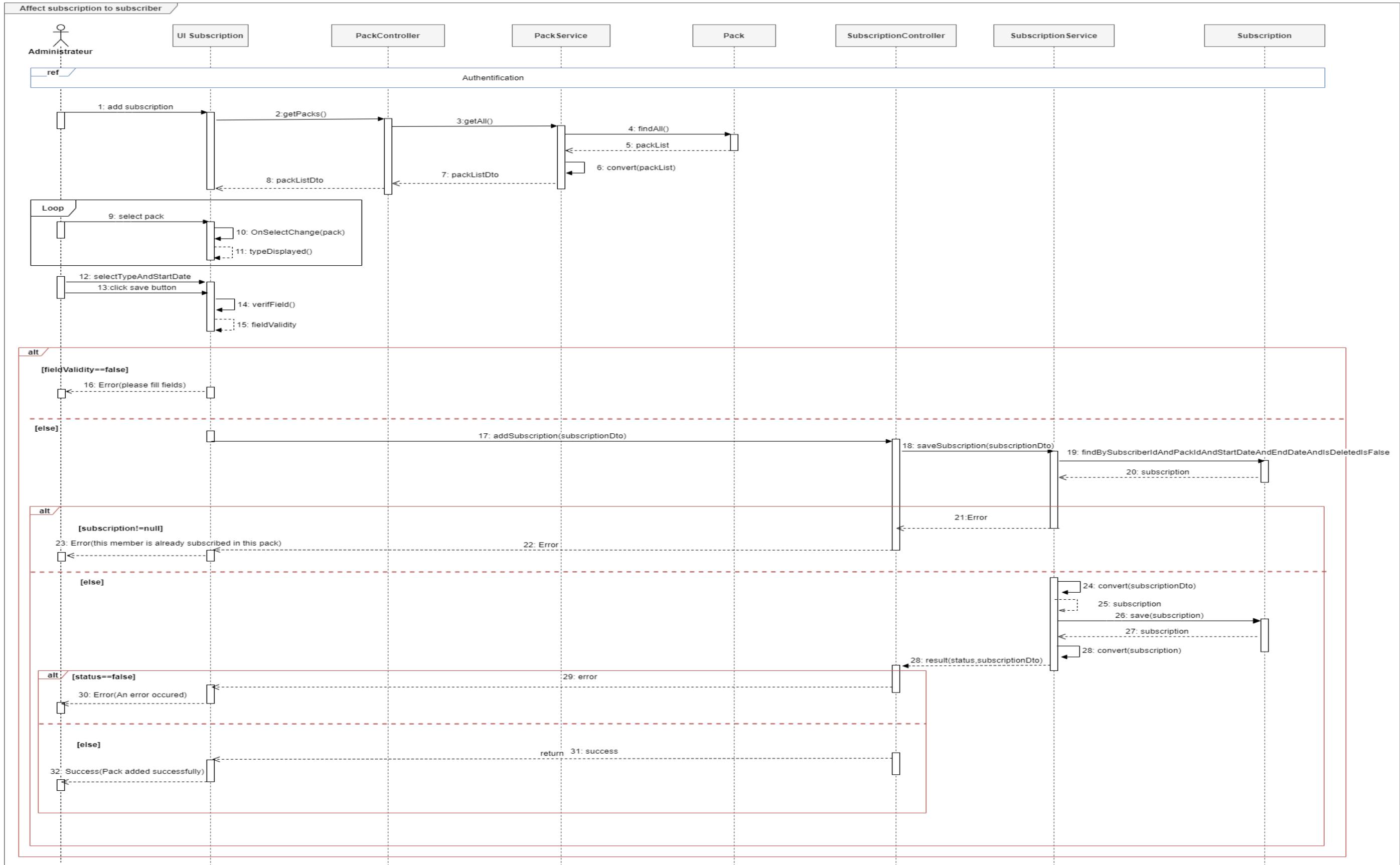


Figure 53. Diagramme de séquence « affecter une inscription à un adhérent »

III.4. Diagramme de classes

La Figure 54 représente le diagramme des classes quatrième sprint. Les classes qui viennent de s'ajouter sont en couleur gris.

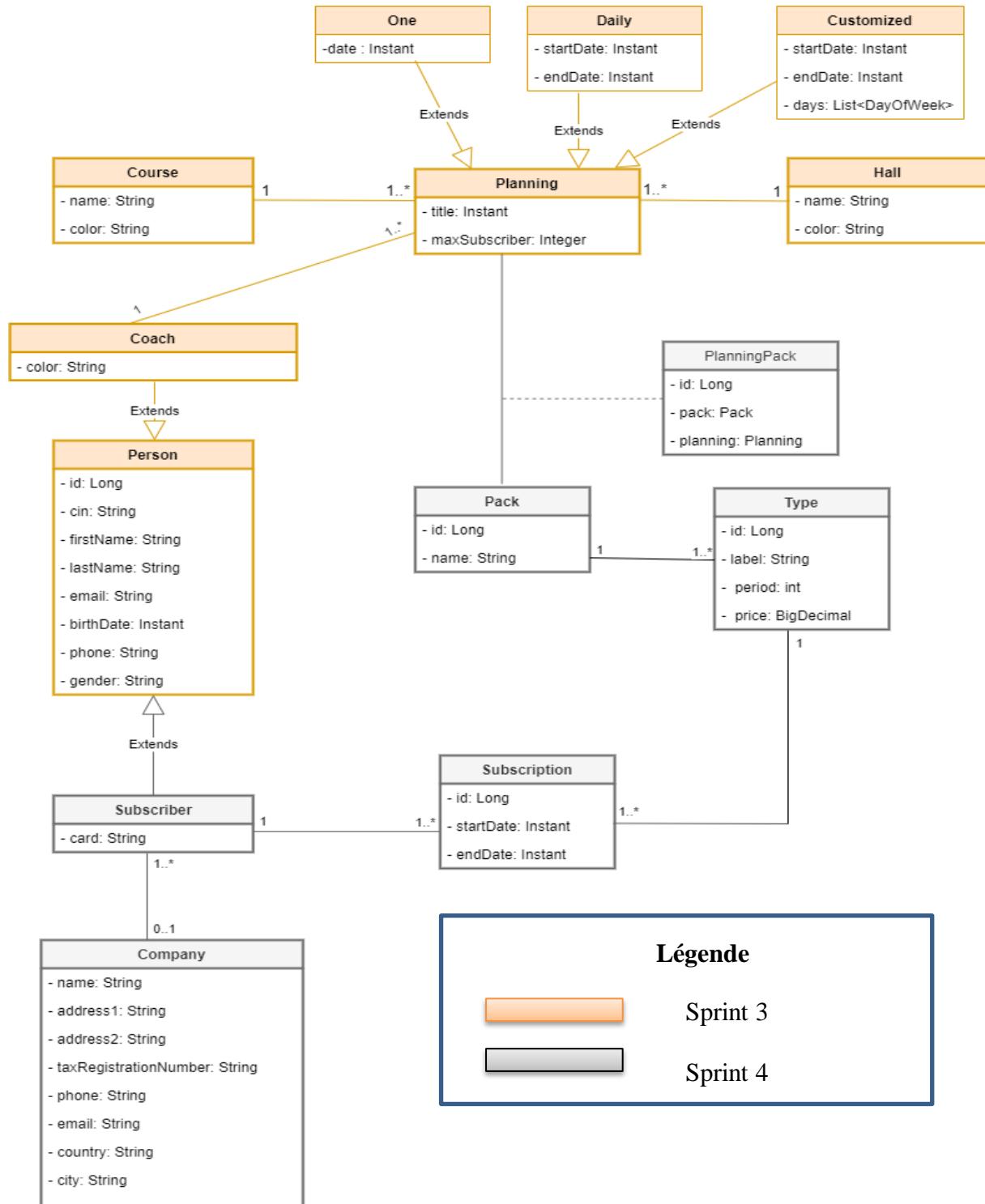


Figure 54. Diagramme de classe du sprint 4

III.5. Réalisation

Dans cette partie, nous présentons les interfaces du quatrième sprint afin de mieux comprendre la réalisation des différentes tâches.

- Gestion des packs

La figure 55 présente l'interface de gestion des packs. A travers cette interface l'administrateur peut consulter les packs existants dans notre système. Il peut également cliquer sur l'icône crayon pour modifier un pack. Comme il peut cliquer sur l'icône poubelle pour supprimer un pack et par ce fait, un message de confirmation apparaîtra.

Liste des Pack		
Id	Nom	actions
1	pack1	⊕ ⚡ ✎
2	pack2	⊕ ⚡ ✎
<small>Items per page: 5 1 – 2 of 2 < < > > </small>		

Figure 55. Interface de visualisation des packs

Comme illustré par la figure 56 un pack possède plusieurs types qui se différencient selon la durée de l'inscription et le prix.

Détails Pack		/pack1 /détail
Abonnement 1 Mois Pack: foot zomba	Prix: 40 TND	
Abonnement 3 Mois Pack: foot zomba	Prix: 120 TND	
Abonnement 6 Mois Pack: foot zomba	Prix: 170 TND	
Abonnement 12 Mois Pack: foot zomba	Prix: 250 TND	
		← Retourner à la liste

Figure 56. Interface de visualisation de détails d'un pack

- Gestion des adhérents

La figure 57 présente l'interface de gestion des adhérents. C'est à travers cette interface que nous pouvons ajouter, modifier ou supprimer un adhérent.

En cliquant sur l'icône œil dans l'interface représentée par la figure 58, une fenêtre contenant les informations de l'adhérent sera affichée comme montre la figure 59. Quant à l'icône calendrier celle permet de visualiser l'historique des inscriptions d'un adhérent qui est représenté par la figure 60.

Figure 57. Interface de gestion des adhérents

Figure 58. Interface de visualisation de informations d'un adhérent

ID	Pack	Date Début	Date Fin	actions
1	pack1	18-05-2021	18-06-2021	
4	pack2	26-05-2021	26-06-2021	

Figure 59. Interface de visualisation de l'historique d'abonnement d'un adhérent

Lors de l'ajout d'un nouvel abonnement, le système va vérifier si cet adhérent a déjà un abonnement en cours si c'était le cas un message d'erreur sera affiché, ceci est illustré par la figure 60 sinon un nouvel abonnement sera enregistré.

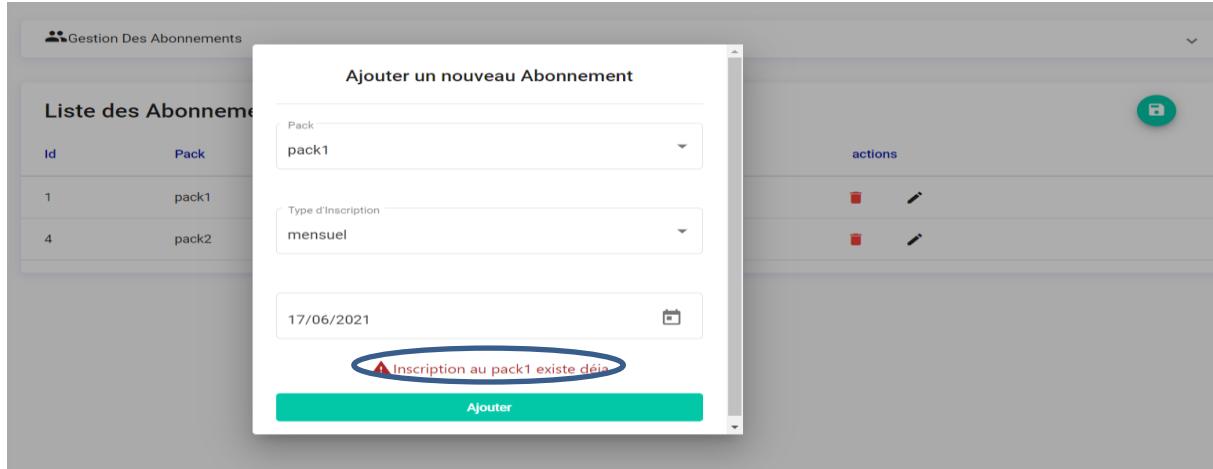


Figure 60. Interface d'ajout d'un nouvel abonnement

IV. Conclusion

Ce chapitre a présenté le backlog du troisième et quatrième sprint, leur étude analytique et conceptuelle. Nous avons détaillé ces parties à travers quelques diagrammes et nous avons terminé par la partie réalisation où nous avons démontré quelques interfaces du livrable.

**CHAPITRE 5 : TRAITEMENT DE LAMISE EN LOCATION DES
TERRAINS ET DES FACTURES**

I. Introduction

Après avoir achevé le développement du troisième et quatrième sprint, nous détaillerons dans ce chapitre le cycle de vie du cinquième sixième et septième sprint. Nous présenterons, pour chaque sprint le backlog, par la suite nous détaillerons le diagramme de cas d'utilisation, la conception et la réalisation.

II. Sprint5

Cette section présente le cinquième sprint. Pour ce dernier, nous nous sommes intéressés par développement de la partie liée à la mise en locations des terrains.

II.1. Backlog Sprint

Le tableau 14 représente l'ensemble des récits utilisateurs réalisées durant ce sprint.

Tableau 14. *Backlog du sprint 5*

Id	Nom Story	User Story
E.1	En tant qu'administrateur, je veux gérer les terrains.	En tant qu'administrateur ,je veux ajouter un terrain en spécifiant les différents types de locations.
		En tant qu'administrateur ,je veux modifier les informations d'un terrain
		En tant qu'administrateur ,je veux supprimer un terrain.
		En tant qu'administrateur, je veux consulter la liste des terrains
		En tant qu'administrateur, je veux consulter les détails d'un terrain

		En tant qu'administrateur, je veux consulter le calendrier de réservation d'un terrain.
E.2	En tant qu'administrateur je veux gérer les locations des terrains.	<p>En tant qu'administrateur ,je veux ajouter une location.</p> <p>En tant qu'administrateur ,je veux modifier une location.</p> <p>En tant qu'administrateur ,je veux supprimer une location.</p> <p>En tant qu'administrateur ,je veux filtrer les locations.</p>

II.2. Diagramme de cas d'utilisation

La figure 61 illustre le diagramme de cas d'utilisation du sprint 5.

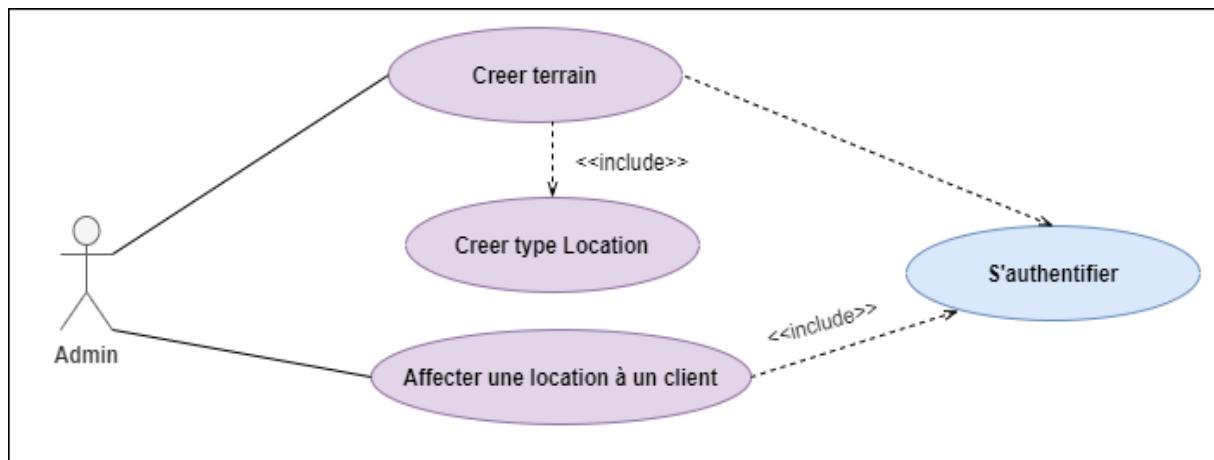


Figure 61. Diagramme de cas d'utilisation détaillé du sprint 5

Une description détaillée du cas d'utilisation « Créeer terrain » est donnée par le tableau 15.

Tableau 15. Description textuelle du cas d'utilisation « Créeer terrain »

Sommaire	
Titre	Créer terrain
Acteur	Administrateur
Résumé	L'administrateur ajoute un nouveau terrain
Description de l'enchaînements	
Pré-condition	L'administrateur doit être authentifié.
Post-condition	Afficher la liste des terrains
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton ajouter nouveau terrain. 2. Le système affiche le formulaire d'ajout. 3. L'administrateur entre un ou plusieurs types de location. 4. L'administrateur clique sur le bouton enregistrer. 5. Le système vérifie les données saisies 6. Le système enregistre les données et affiche la liste des terrains.
Scénario alternatif	<p>A1 : L'administrateur ne remplit pas les champs obligatoires du formulaire:</p> <ol style="list-style-type: none"> 1. Le système affiche « champs obligatoire » 2. Le scénario nominal reprend au point 2 <p>A2 : le choix du type de location « Hebdomadaire »</p> <ol style="list-style-type: none"> 1. L'administrateur doit entrer la période (en nombre de mois) 2. Le scénario nominal reprend au point 3 <p>A3 : Le terrain portant le même nom existe déjà :</p> <ol style="list-style-type: none"> 3. Le système affiche une alerte « Terrain existe déjà ! ». 4. Le scénario nominal reprend au point 2

Une description détaillée du cas d'utilisation « affecter une location à un client » est donnée par le tableau 16.

Tableau 16. *Description textuelle du cas d'utilisation « Affecter une location à un client »*

Sommaire	
Titre	Créer terrain
Acteur	Administrateur
Résumé	L'administrateur ajoute une nouvelle location
Description de l'enchaînements	
Pré-condition	L'administrateur doit être authentifié.
Post-condition	Afficher la liste des locations
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton ajouter. 2. Le système affiche le formulaire d'ajout. 3. L'administrateur choisit le client, le terrain à louer, le type de location ainsi que la date début de location. 4. L'administrateur clique sur le bouton enregistrer. 5. Le système vérifie les données saisies 6. Le système enregistre les données et affiche la liste des terrains.
Scénario alternatif	<p>A1 : L'administrateur ne remplit pas les champs obligatoires du formulaire:</p> <ol style="list-style-type: none"> 1. Le système affiche « champs obligatoire » 2. Le scénario nominal reprend au point 2 <p>A2 : le choix du type de location « Hebdomadaire »</p> <ol style="list-style-type: none"> 1. L'administrateur doit choisir le jour 2. Le scénario nominal reprend au point 4 <p>A3 : Le terrain est déjà loué pour la date entrée :</p> <ol style="list-style-type: none"> 1. Le système affiche une alerte « Terrain réservé ! ». 2. Le scénario nominal reprend au point 2

II.3. Diagramme de séquences

La figure 62 résume l'interaction entre les classes qui interviennent lors de la création d'un terrain.

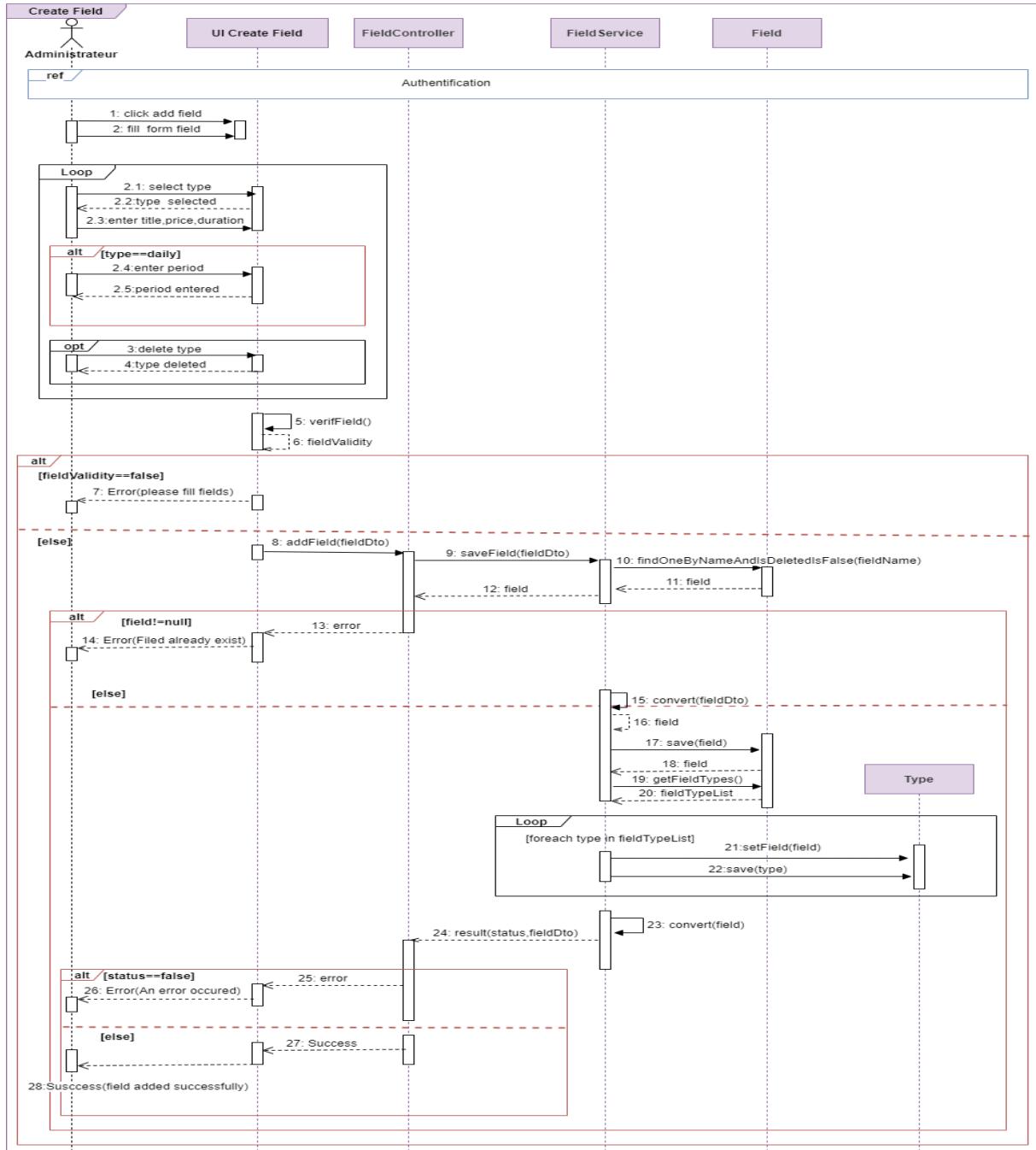


Figure 62. Diagramme de séquence du cas d'utilisation « créer terrain »

La figure 63 résume l'interaction entre les classes qui interviennent lors de l'affectation d'une location à un client.

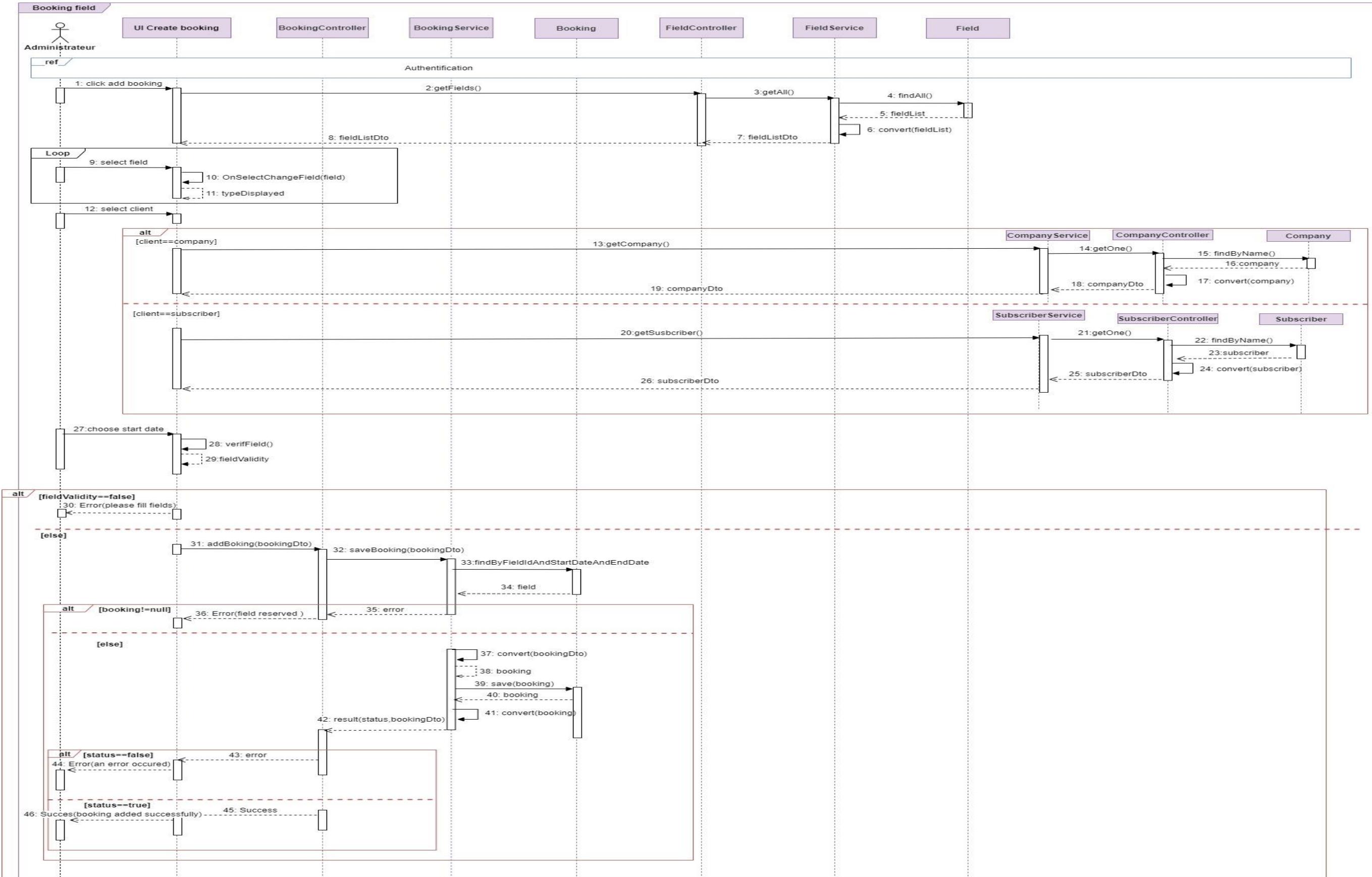


Figure 63. Diagramme de séquence « affecter une location à un client »

II.4. Diagramme de classes

Le diagramme de classes du sprint 5 est représenté dans la figure 64.

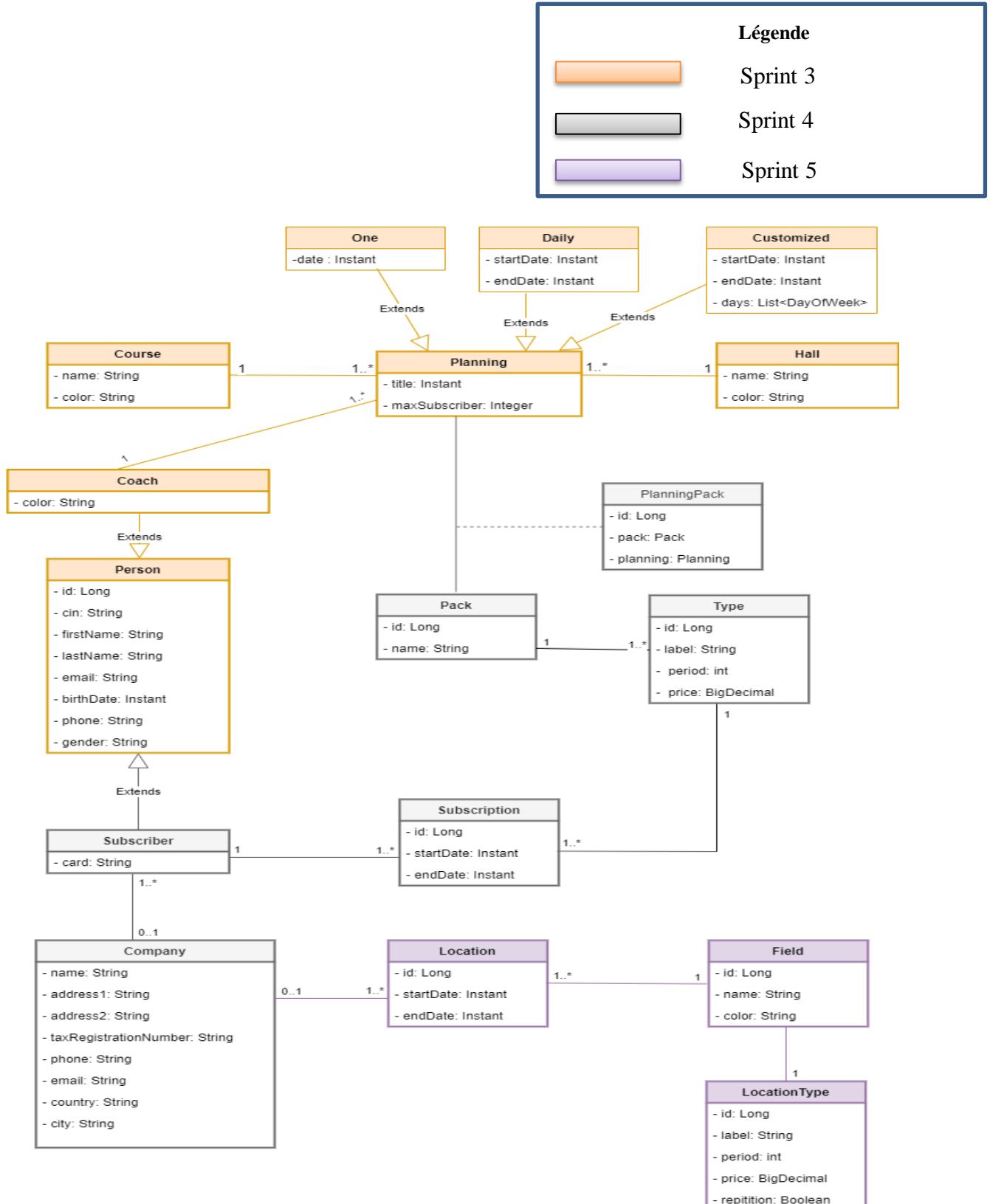


Figure 64. Diagramme de classe du sprint 5

Le diagramme de classes de sprint 5 est composé de 3 classes. La classe « Field » regroupe les données du terrain. En effet, une location est concernée par un terrain qui possède plusieurs types caractérisés par la période et le prix.

II.5. Réalisation

Dans cette section, nous allons présenter quelques exemples d'interfaces graphiques réalisées au cours de ce cinquième sprint.

La figure 65 montre l'interface de gestion des terrains.

Liste des Terrains		
Id	Nom	actions
1	foot	
2	tennis	
3	basket	

Items per page: 5 | < < > >|

Figure 65. Interface de gestion des terrains

L'administrateur peut ajouter un nouveau terrain comme illustré par la figure 66. Dans un premier temps l'administrateur doit entrer le nom du terrain ensuite, il doit ajouter un ou plusieurs types de locations en spécifiant la durée, le prix de la location et si le type de location est hebdomadaire il doit entrer la période en nombre de mois.

Ajouter Terrain				
Nom				
tennis				
Type de location				
Intitulé par jour	Liste des types une seule fois	Durée(en minutes) 120	Prix 100	
Intitulé par mois	Liste des types Hebdomadaire	Durée(en minutes) 60	Période(en mois) 1	
Ajouter				

Figure 66. Interface d'ajout d'un nouveau terrain

En cliquant sur l'icône calendrier, un calendrier sera affiché contenant les réservations associées au terrain par semaine comme indiqué par la figure 67.

Le calendrier affiche une grille horizontale par jour et une grille verticale par heure. Les heures sont marquées de 12 AM à 10 AM. Les jours sont dimanche (mai 30), lundi (mai 31), mardi (juin 1), mercredi (juin 2), jeudi (juin 3), vendredi (juin 4) et samedi (juin 5). Une réservation est indiquée pour le mardi matin de 9 AM à 10 AM, avec une étiquette 'réserve'. Des boutons pour 'Activer Windows' et 'Accédez aux paramètres pour activer Windows.' sont visibles dans le coin inférieur droit.

	dimanche mai 30	lundi mai 31	mardi juin 1	mercredi juin 2	jeudi juin 3	vendredi juin 4	samedi juin 5
12 AM							
1 AM							
2 AM							
3 AM							
4 AM							
5 AM							
6 AM							
7 AM							
8 AM							
9 AM							
10 AM			réserve				

Figure 67. Calendrier des réservations d'un terrain

Après avoir mis un terrain en location, l'administrateur peut créer une réservation en spécifiant le terrain adéquat, le type de location, le client, ainsi que la date de location comme le montre la figure 68.

L'interface de gestion des réservations de terrain permet de créer une nouvelle réservation. Le formulaire 'Ajouter Réservation' demande les informations suivantes : Terrain (basket), Type de location (par jour), Client (compagnie), Compagnie (DK Soft) et Date Début (25/06/2021). La liste des réservations existantes est visible au bas de l'écran, avec des colonnes pour l'Id et le Terrain. Des boutons pour ajouter, modifier et supprimer sont également disponibles.

Figure 68. Interface de gestion des réservations de terrain

III. Sprint6

Le sixième sprint a pour de but d'assurer la génération des factures. De plus nous avons réalisé le paiement des factures et nous assurons la consultation des historiques de paiements pour chaque facture.

III.1. Backlog du sprint 6

Le tableau 17 présente l'ensemble des histoires utilisateurs réalisées durant le sprint 6.

Tableau 17. Backlog du sprint 6

Id	Nom Story	User Story
F.1	En tant qu' administrateur, je veux établir une facture.	En tant qu'administrateur, je veux générer une facture pour une compagnie
		En tant qu'administrateur, je veux générer une facture pour un adhérent
		En tant qu'administrateur, je veux modifier une facture dont l'état est brouillon
		En tant qu'administrateur, je veux filtrer les factures selon le client.
		En tant qu'administrateur, je veux filtrer les factures selon l'état(Brouillon, Non payé, Partiellement payé, Payé)
		En tant qu'administrateur, je veux supprimer une facture brouillon
		En tant qu'administrateur, je veux imprimer une facture
		En tant qu'administrateur, je veux valider une facture.
	En tant qu'administrateur, je veux encaisser une facture.	En tant qu'administrateur je veux payer une facture en espèce.
		En tant qu'administrateur, je veux payer une facture par chèque.

F.2		En tant qu'administrateur, je veux payer une facture par facilité.
		En tant qu'administrateur, je veux consulter l'historique de paiement pour chaque facture.

III.2. Diagramme de cas d'utilisation

La figure 69 illustre le diagramme de cas d'utilisation du sprint 6

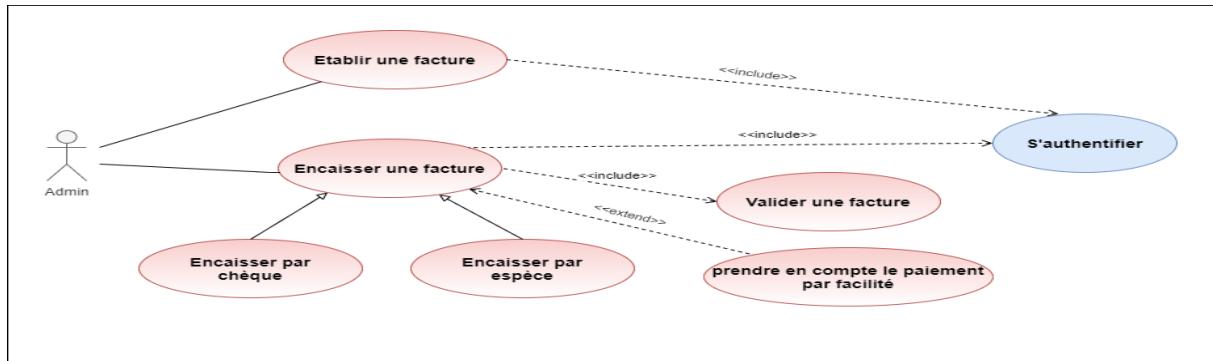


Figure 69. Diagramme de cas d'utilisation détaillé du sprint 6

Une description détaillée du cas d'utilisation « Etablir une facture » est donnée par le tableau 18.

Tableau 18 : Description textuelle du cas d'utilisation « Etablir une facture »

Sommaire	
Titre	Etablir une facture
Acteur	Administrateur
Résumé	L'administrateur encaisse une facture.
Description de l'enchaînements	
Pré-condition	L'administrateur doit être authentifié.
Post-condition	Afficher la facture créée.

Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur l'icône d'ajout. 2. Le système affiche le formulaire. 3. L'administrateur choisit le client « compagnie ». 4. L'administrateur entre le nom de la compagnie. 5. Le système affiche la liste des locations de cette compagnie ainsi la liste des inscriptions pour chaque adhérent associé à cette compagnie. 7. L'administrateur choisit la liste des inscriptions et des locations à facturer. 8. L'administrateur clique sur le bouton « Créer Facture » 9. Le système enregistre la facture, affiche un message « Facture créée avec succès » et affiche la facture récemment créée.
Scénario alternatif	<p>A1 le client choisi est un adhérent</p> <ol style="list-style-type: none"> 1. L'administrateur entre le nom de l'adhérent 2. Le système affiche la liste des inscriptions et des locations associées à cet adhérent. <p>Le scénario nominal reprend au point 6</p>

Une description détaillée du cas d'utilisation « Encaisser par espèce » est donnée par le tableau 19.

Tableau 19. *Description textuelle du cas d'utilisation « Encaisser par espèce »*

Sommaire	
Titre	Encaisser par espèce
Acteur	Administrateur
Résumé	L'administrateur encaisse une facture.
Description de l'enchaînements	
Pré-condition	<p>L'administrateur doit être authentifié.</p> <p>L'utilisateur est inscrit au système.</p>

Post-condition	Afficher la liste des factures
Scénario nominal	<ol style="list-style-type: none"> 1. L'administrateur clique sur le bouton payer. 2. Le système affiche le formulaire. 3. L'administrateur choisit le type de paiement en espèce. 4. L'administrateur choisit le mode de paiement par facilité 5. L'administrateur entre la date d'encaissement et le montant. 6. L'administrateur clique sur le bouton payer. 7. Le système vérifie les données saisies 8. Le système enregistre les données et affiche la liste des factures.
Scénario alternatif	<p>A1 Le mode de paiement n'est pas par facilité</p> <ol style="list-style-type: none"> 1. Le système affiche un message en indiquant que le montant entré est inférieur au montant à payer 2. Le scénario nominal reprend au point 4 <p>A2 Le montant entré est supérieur au montant de facture</p> <ol style="list-style-type: none"> 1. Le système affiche un message en indiquant que le montant entré est supérieur au montant à payer 2. Le scénario nominal reprend au point 5

III.3. Diagramme de classes

La figure 70 représente le diagramme de classe résultant. Le diagramme de classes de sprint six est composé de cinq classes. En effet, chaque facture doit être associé obligatoirement à une compagnie ou bien à un adhérent. Chaque facture peut posséder plusieurs paiements.

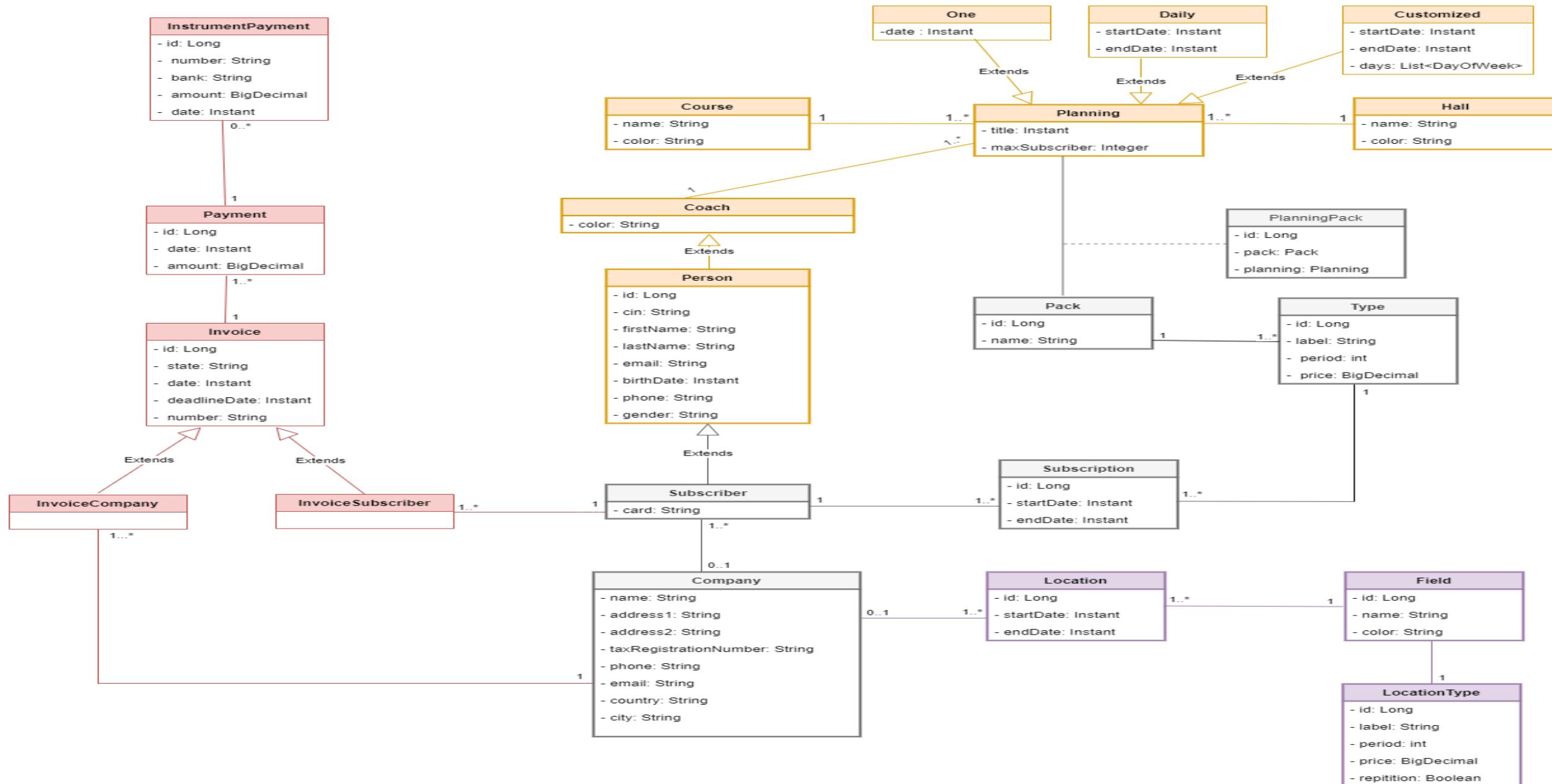
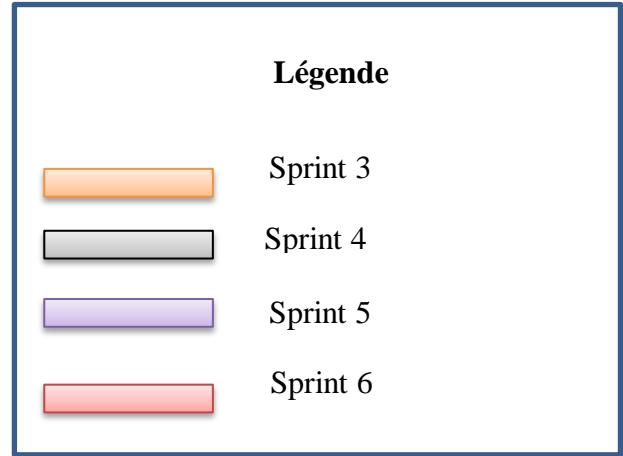


Figure 70. Diagramme de classes résultant

III.4. Diagramme de séquence

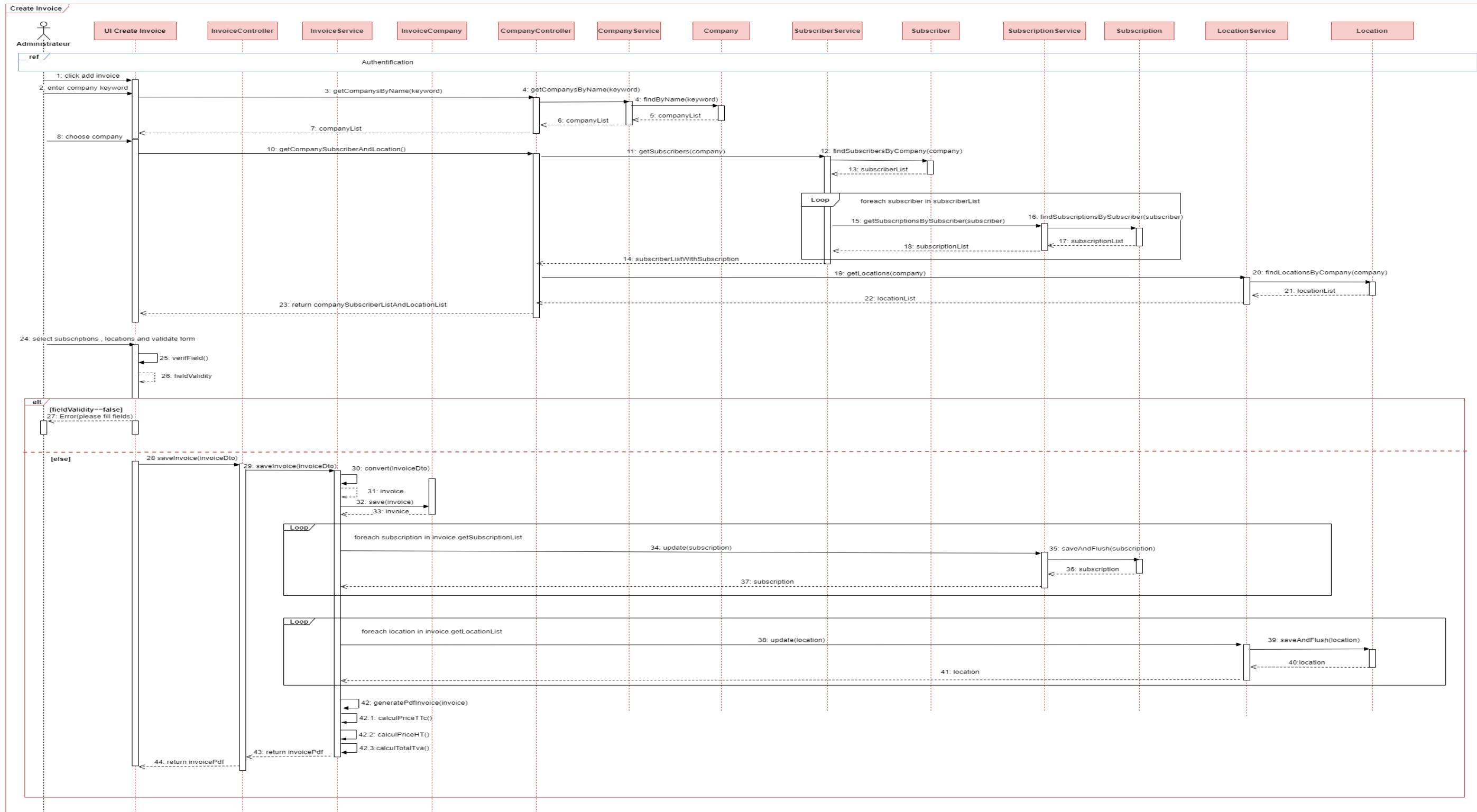


Figure 71. Diagramme de séquence du cas d'utilisation « Etablir une facture »

III.5. Réalisation

Par défaut la liste des factures est affichée comme illustré par la figure 72. Dès sa création une facture possède l'état « Brouillon », cette facture peut être supprimée ou modifiée.

En cliquant sur l'icône « valider » la facture passe à l'état non payée et un code sera affecté.

The screenshot shows the 'Fitness Access Tracker' application interface. On the left is a sidebar with various menu items: Statistique, Calendrier, Recettes, Compagnies, Abonnés, Facture (selected), Cours, Entraineurs, Salle, Planification, Pack, Terrain, and Réservation. The main area is titled 'Gestion Des Factures' and contains a sub-section 'Liste des Factures'. A table lists invoices with columns: Id, Client, Code, Etat, Date Sortie, and actions. The first invoice (Id 18) has its 'Code' field circled in blue. The fourth invoice (Id 50) is highlighted with a red border and has its 'Etat' field circled in pink. The bottom right of the table area shows 'Activer Windows' and 'Items per page: 5'. The top right of the main window has icons for notifications, settings, and others.

ID	Client	Code	Etat	Date Sortie	actions
18	yanguil nour	#202100025	Non payé	23/05/2021	[edit] [download] [print] [trash]
44	yanguil nour	#202100026	Payé	29/05/2021	[edit] [download] [print] [trash]
45	mbarki samar	#202100027	Payé	29/05/2021	[edit] [download] [print] [trash]
50	DK Soft	-----	Brouillon	31/05/2021	[edit] [download] [print] [trash] [red] [blue] [checkmark]
51	mbarki samar	#202100028	Partiellement payé	09/06/2021	[edit] [download] [print] [trash]

Figure 72. Liste des factures

L'administrateur peut filtrer les factures par groupement de compagnie ou selon l'état de la facture ou bien en combinant les deux filtres.

Il suffit de choisir le nom de la compagnie et l'état comme illustré par la figure 73.

This screenshot shows the same application interface as Figure 72, but with filters applied. In the top-left search bar, 'DK Soft' is entered under 'Compagnie' and 'Brouillon' is selected under 'Liste état Facture'. Below the search bar, the 'Chercher' (Search) button is highlighted in green. The rest of the interface is identical to Figure 72, showing the list of invoices with the same structure and highlighting.

ID	Client	Code	Etat	Date Sortie	actions
50	DK Soft	-----	Brouillon	31/05/2021	[edit] [download] [print] [trash] [red] [blue] [checkmark]

Figure 73. Liste des factures avec filtre « état facture » & « compagnie »

L'administrateur peut générer une facture en cliquant sur l'icône d'ajout. Il doit choisir tout d'abord le client qui peut être une compagnie ou bien une personne physique comme illustré par la figure 74.

Figure 74. Interface de création d'une facture

L'étape suivante consiste à choisir les inscriptions et les locations à facturer ceci est illustré par la figure 75.

	Terrain:foot	Label: par jour	Durée :60 minutes	Prix: 40,000 TND
<input checked="" type="checkbox"/>				
<input type="checkbox"/>				

Figure 75. Interface de création d'une facture

En cliquant sur « Crée Facture » une facture sera générée comme le montre la figure 76. Cette facture contient les coordonnés du client, les informations concernant la salle du sport, les informations relatives aux inscriptions et aux terrains loués et le montant total de la facture.

 Fitness Access Tracker																													
		Date: 2021-06-23																											
		Etat: Brouillon																											
Informations Société																													
Fitness Access Tracker rte lafrane km 3.8 123 Adresse:rte lafrane km 3.8 Matricule Fiscal:123 Email:fitness@gmail.com Tel: (+216)23251319																													
Informations Client																													
DK Soft Adresse:rte lafrane km 3.5 Matricule Fiscal:123 Email: dkSoft@gmail.com Tel: (+216)23251319																													
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Description</th> <th>Qté</th> <th>Prix unitaire</th> <th>Prix HT</th> <th>TVA</th> <th>Prix TTC</th> </tr> </thead> <tbody> <tr> <td>terrain foot par jour</td> <td>1</td> <td>36.80</td> <td>36.80 TND</td> <td>8.00 %</td> <td>40 TND</td> </tr> <tr> <td>pack2 mensuel</td> <td>1</td> <td>92.00</td> <td>92.00 TND</td> <td>8.00 %</td> <td>100 TND</td> </tr> <tr> <td>pack2 trimestriel</td> <td>1</td> <td>248.40</td> <td>248.40 TND</td> <td>8.00 %</td> <td>270 TND</td> </tr> </tbody> </table>						Description	Qté	Prix unitaire	Prix HT	TVA	Prix TTC	terrain foot par jour	1	36.80	36.80 TND	8.00 %	40 TND	pack2 mensuel	1	92.00	92.00 TND	8.00 %	100 TND	pack2 trimestriel	1	248.40	248.40 TND	8.00 %	270 TND
Description	Qté	Prix unitaire	Prix HT	TVA	Prix TTC																								
terrain foot par jour	1	36.80	36.80 TND	8.00 %	40 TND																								
pack2 mensuel	1	92.00	92.00 TND	8.00 %	100 TND																								
pack2 trimestriel	1	248.40	248.40 TND	8.00 %	270 TND																								
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td>Total H.T.</td> <td>377.20 TND</td> </tr> <tr> <td>Total T.V.A</td> <td>30.1760 TND</td> </tr> <tr> <td>Timbre fiscal</td> <td>0.600 TND</td> </tr> <tr> <td>Total T.T.C</td> <td>410.600 TND</td> </tr> </table>						Total H.T.	377.20 TND	Total T.V.A	30.1760 TND	Timbre fiscal	0.600 TND	Total T.T.C	410.600 TND																
Total H.T.	377.20 TND																												
Total T.V.A	30.1760 TND																												
Timbre fiscal	0.600 TND																												
Total T.T.C	410.600 TND																												
Montant en toute lettre : quatre cent dix Dinars et six cent Millimes																													

Figure 76. Interface de visualisation d'une facture

En cliquant sur l'icône « payer » l'interface illustrée par la figure 77 est affichée, l'administrateur choisit le type de paiement (en espèce / par chèque) ainsi que le mode de paiement.

The screenshot shows a payment interface with the following details:

- Reste à payer:** 410,600 TND
- Type de paiement:** En espèce Par chèque
- Mode de paiement:** Facilité
- Date d'encaissement:** 23/06/2021

Figure 77. Interface de paiement d'une facture

L'historique d'encaissement permet à l'administrateur de consulter tous les montants encaissés avec les détails de type de paiement (en espèce/par chèques) ainsi que le montant assigné. La figure 78 montre l'interface « historique de paiement » d'une facture.

1 Historique De Paiement			
Date Facturation	Code	Débit	Crédit
29/05/2021	#202100027	140,600 TND	
29/05/2021	Paiement espèce		20,000 TND
29/05/2021	Paiement Chèque		80,000 TND
29/05/2021	Paiement espèce		40,600 TND

Figure 78. Interface historique de paiement

IV. Sprint 7

Cette partie est consacrée pour la présentation du septième sprint. En effet, nous allons présenter tout d'abord le backlog produit, le diagramme des cas d'utilisation et nous clôturons par la présentation de quelques exemples d'interfaces graphiques.

IV.1. Backlog du sprint 7

Le tableau 20 présente l'ensemble des histoires utilisateurs réalisées durant le sprint 7.

Tableau 20. Backlog du sprint 7

Id	Nom Story	User Story
G.1	Consulter les notifications lancées par le système	En tant qu'administrateur, je veux être notifié par le système en cas de l'échéance d'une facture.
G.2	En tant qu'administrateur, je veux consulter les revenus mensuels.	En tant qu'administrateur, je veux filtrer les revenus par date.
G.3	En tant qu'administrateur, je veux consulter les statistiques.	En tant qu'administrateur, je veux savoir l'occupation des terrains par heure.
		En tant qu'administrateur, je veux filtrer le nombre de packs achetés par genre et par âge.

IV.2. Diagramme de cas d'utilisation

La figure 79 résume les fonctionnalités que l'administrateur peut avoir après ce sprint.

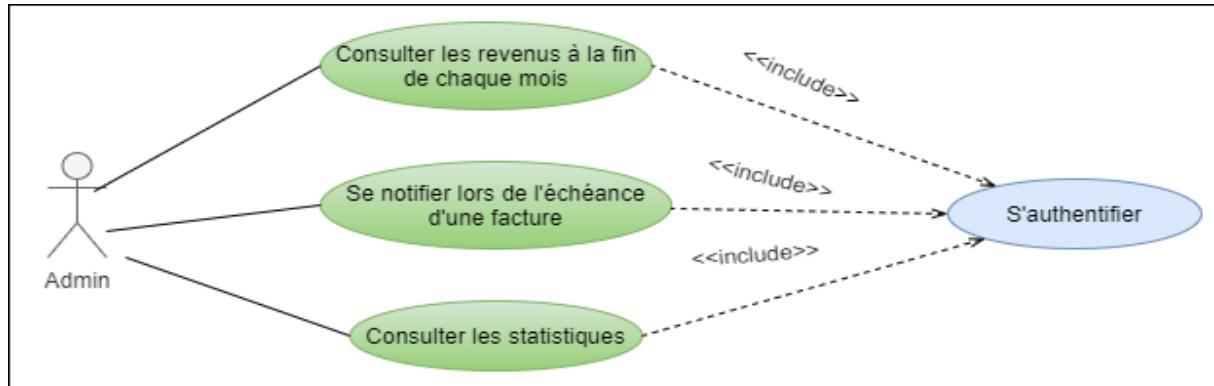


Figure 79. Diagramme de cas d'utilisation sprint6

IV.3. Conception UML

Le diagramme de classes de septième sprint est identique au diagramme de classes du sprint précédent.

IV.4. Réalisation

La figure 80 présente l'interface de visualisation de la liste des recettes, par défaut cette liste affiche les revenus de l'année en cours par mois et par client. Grace au filtre « date début » et « date fin » nous pouvons visualiser les revenus dans une période bien déterminée.

mai 2021		551,800 TND
juin 2021		151,200 TND
mbarki samar	09/06/2021	50,000 TND
mbarki samar	11/06/2021	20,000 TND
telnet	17/06/2021	60,600 TND

Total: 703,000 TND

Figure 80. Interface de visualisation des revenus

Les notifications représentées par la figure 81 sont envoyées pour renseigner l'administrateur en cas où une facture a dépassé les délais fixés pour le paiement.

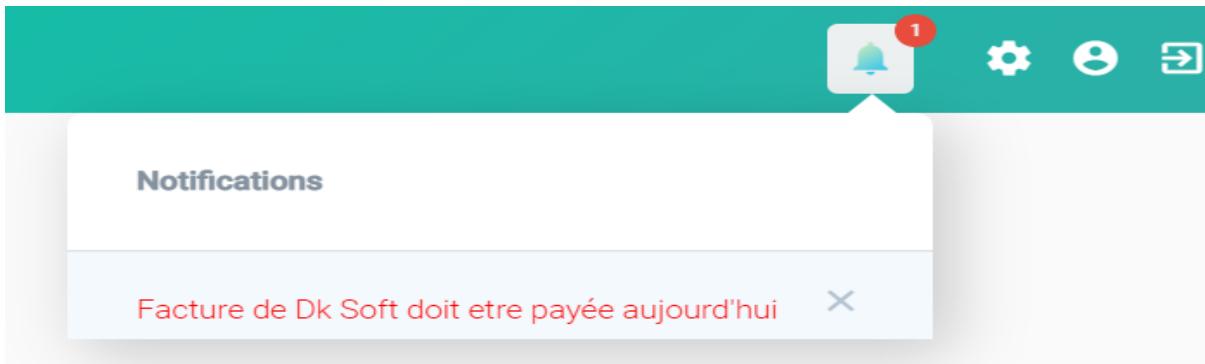


Figure 81. Interface de consultation des notifications

L'administrateur peut consulter les statistiques pour avoir une idée sur l'occupation des terrains par heure en utilisant les filtres date début et date fin. Ainsi il peut avoir une idée sur le nombre de packs achetés par genre et par âge.

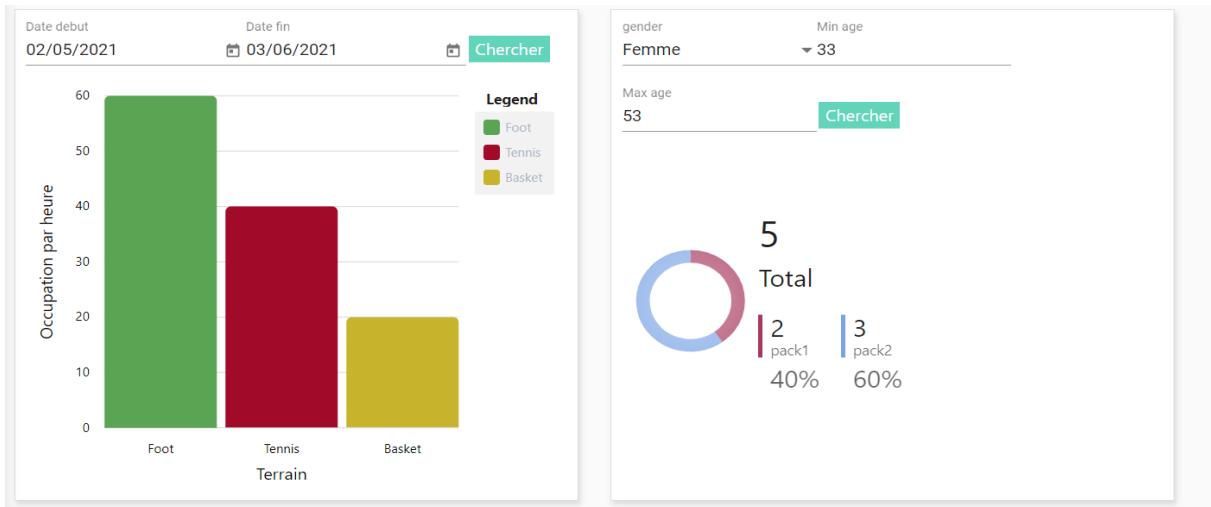


Figure 82. Interface de consultation des statistiques

V. Conclusion

Au cours de ce chapitre nous avons présenté les sprints cinq six et sept. Par la suite, nous avons présenté les diagrammes de conception avec les différentes tâches réalisées.

Dans ce qui suit, nous présenterons une conclusion générale qui récapitule la démarche que nous avons suivie tout au long de notre stage.

Conclusion générale et perspectives

Tout au long de ce rapport, nous avons présenté les différents aspects d'élaboration de notre projet de sa phase d'analyse jusqu'à la phase d'implémentation.

Notre solution consiste à développer une application web à base de l'architecture microservices spécialisée dans la gestion d'une de sport. En effet, notre application offre plusieurs fonctionnalités pour faciliter le processus de gestion d'une salle de sport à savoir la gestion des adhérents et des inscriptions, mise en location des terrains, la génération des factures et le suivi des paiements.

Ce travail a été une occasion pour mettre en pratique les connaissances que j'ai acquises pendant ma formation à l'École Nationale d'Ingénieurs de Sfax et de les enrichir. En effet, il m'a procuré une opportunité pour, d'une part aborder un domaine métier et d'autre part confirmer une fois de plus mes compétences dans le développement Java/JEE et Web.

Ce stage a touché aussi le côté humain. Il était très bénéfique car nous avons bien intégré dans l'environnement d'entreprise ce qui a fortifié nos compétences de communication, de collaboration et de conduite de projets.

Il est à signaler que la solution que nous avons proposée, reste une première version de « Fitness Access Tracker ». Dans le futur, nous planifions d'étendre notre application par l'ajout d'un microservice pour la gestion des équipements de la salle de sport. Aussi, nous visons à développer un microservice qui présente un système intelligent qui aide les adhérents à prendre des décisions pour le choix des cours adéquat en fonction de leurs besoins physiologiques.

BIBLIOGRAPHIE

- [1] DK Soft, <https://www.dksoft.tn/> [02/03/2021]
- [2] LogiSam, <https://logisam.com/logisport-logiciel-de-gestion-pour-etablissements-sportifs-salles-ou-club-de-sport/> [13/03/2021]
- [3] Gym Management, https://apps.odoo.com/apps/modules/12.0/abs_gym_management/
- [4] Jules Renard « Les acteurs d'un projet Scrum » [13/03/2021]
- [5] Jira, <https://www.atlassian.com/fr/software/jira/guides/use-cases/what-is-jira-used-for> [13/03/2021]
- [6] Visual Studio Code, <http://www.ird.fr/us191/spip.php?article74> [13/03/2021]
- [7] Eclipse, <https://www.techno-science.net/definition/517.html>
- [8] Postman, <https://blog.webnet.fr/presentation-de-postman-outil-multifonction-pourapiweb/> [13/03/2021]
- [9] HTML5, <https://www.journaldunet.fr/web-tech/dictionnaire-duwebmastering/1203257-html5-hypertext-markup-langage5-definition-traduction/> [13/03/2021]
- [10] CSS, <https://www.w3schools.com/css/> [13/03/2021]
- [11] TypeScript, <https://www.typescriptlang.org/> [13/03/2021]
- [12] REST, <https://www.lemagit.fr/definition/REST> [13/03/2021]
- [13] MySQL, <https://www.keepapi.ovh/formation/developpeur/mysql/> [13/03/2021]
- [14] Spring Boot, <https://spring.io/projects/spring-boot> [13/03/2021]
- [15] Hibernate, <https://www.jmdoudoux.fr/java/dej/chap-hibernate.htm> [13/03/2021]
- [16] Angular, <https://angular.io/> [13/03/2021]
- [17] Angular Materials, [13/03/2021]
- [18] UML, <https://web.maths.unsw.edu.au/~lafaye/CCM/uml/umlcara.htm> [13/03/2021]
- [19] Draw.io, <https://fondationlitterairefleurdelys.com/2014/01/30/draw-io-un-outil-pour-dessiner-des-diagrammes-en-ligne/> [13/03/2021]

- [20] Bitbucket, <https://bitbucket.org/product> [13/03/2021]
- [21] Slack, <https://slack.com/> [13/03/2021]
- [22] Microservice, <https://microservices.io/> [13/03/2021]
- [23] SSO, <https://www.onelogin.com/fr/learn/how-single-sign-on-works> [13/03/2021]
- [24] OAuth2, <https://accetal.fr/oauth2-pour-securiser-les-api/> [13/03/2021]
- [25] Keycloak, <https://blog.desdelinux.net/fr/keycloak%2C-une-solution-open-source-de-gestion-des-identités-et-des-accès/> [13/03/2021]
- [26] Diagramme de cas d'utilisation UML, <http://www.uml-sysml.org/diagrammes-uml-et-sysml/diagramme-uml/use-case-diagramme>, consulté en mai 2020.
- [27] Diagramme de class UML, <http://www.uml-sysml.org/diagrammes-uml-et-sysml/diagramme-uml/use-case-diagramme>, consulté en mai 2020.

Conception et développement d'une application de gestion d'une salle de sport

Nour Elimen Yangui

الخلاصة: يندرج هذا العمل المنجز في شركة « Dk Soft » في سياق مشروع ختم الدروس للحصول على الشهادة الوطنية في الهندسية الإعلامية. الهدف من هذا المشروع هو تصميم وتنفيذ تطبيق واب لإدارة قاعة رياضية. هذه التطبيقة تمكن مستخدميها من إدارة المنخرطين وعملية التسجيل، عملية تنظيم حجز الملاعب، إعداد الفواتير وعملية الاستخلاص. هذا العمل يعتمد على بنية الخدمات المصغرة وتسجيل الدخول الفردي عن طريق « keycloak ».

Résumé : Ce travail s'inscrit dans le contexte du projet de fin d'études pour l'obtention du diplôme national d'ingénieur en génie informatique réalisé au sein de la société « Dk Soft ». Le but de ce travail est la conception et le développement d'une application de gestion d'une salle de sport. La solution proposée offre les fonctionnalités suivantes à savoir la gestion des adhérents et des inscriptions, mise en location des terrains, génération des factures et le suivi des paiements. Cette application est basée sur l'architecture microservices et sécurisée à travers l'outil de gestion des identités et des accès Keycloak.

Abstract : This work, is a part of graduation project for the national degree in computer engineering and it is carried out within the company « Dk Soft ». It aims to design and implement a web application for the management of a gym. The proposed solution offers the following functionalities, namely the management of members and the registrations, the rental of gym fields, as well as the generating invoices and the control of payments. This application is based on the microservices architecture and secured through Keycloak which is an identity and an access management tool.

المفاتيح: تطبيق الويب، سيرينق، أنقالر، سكروم، ميكرو سيرفيس

Mots clés: Application Web, Spring, Angular, Scrum, Microservices, Keycloak

Key-words: Web Application, Spring, Angular, Scrum, Microservices, Keycloak