Chapitre 3: conception et modélisation

Introduction

La modélisation conceptuelle et organisationnelle constitue une étape importante dans la convergence des notations utilisées dans le domaine de l'analyse de conception objet .Elle a pour objectif d'élaborer des modèles détaillés de l'architecture du système à partir du modèle obtenu lors de l'étape d'analyse des besoins. Elle vise également à réduire la complexité du système.

Dans cette partie, nous présentons la vision conceptuelle des éléments résultant de l'étude fonctionnelle qui précède, plus précisément, on fait une analyse des spécifications des besoins que nous avons déterminé pour concevoir et modéliser le système de gestion.
Nous commençons par justifier le choix du formalisme de modélisation utilisé, à savoir, l'UML, celui-ci va nous aider à faire ressortir le modèle conceptuel détaillé de notre application qui regroupe les différents diagrammes notamment, le diagramme de cas d'utilisation, les diagrammes d'activité, diagramme des classes et les différents diagrammes de séquence.

1. Modélisation fonctionnelle des données

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisations et les structures sous une forme qui facilite la compréhension (scénarios) que nous avons présentés dans les descriptions textuelles et les diagrammes de séquences.
L'étude de conception est la phase la plus importante du cycle de développement d'un système informatique. En effet, elle permet de confronter la spécification et l'analyse avec l'implémentation, elle présente le point de convergence des deux aspects :
« Le quoi faire » (analyse) et le « comment faire » (réalisation).

1. Choix du formalisme de conception

Dans la cadre de notre projet, nous avons opté pour le langage UML comme une approche de conception. Ci-dessous, nous présentons ce langage puis nous justifions notre choix.

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait de lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet

d’exprimer visuellement une solution objet, ce qui facilite la comparaison et l’évaluation des solutions. L’aspect de sa notation, limite l’ambigüité et les incompréhensions.

a) **Identification des acteurs :**

Dans notre cas les acteurs sont :
Les utilisateurs qui peuvent utiliser le système et faire des opérations d’insertion, de suppression, et de modification des formations.
Parmi ces utilisateurs il y a l’administrateur, qui possédera tous les droits dont celui de pouvoir ajouter, modifier et supprimer les données des formations ou bien des autres utilisateurs.

Un acteur peut être un :
Client

b) Identification des cas d’utilisations :
Nous décrivons pour chaque acteur les cas d’utilisation. On distingue les cas d'utilisation suivants :

Acteur	**Rôle**
Client	Manipuler le moteurs Consulter les valeurs des capteurs

Tableau 2: Liste des rôles des acteurs du diagramme des cas d’utili sation

c) Représentation des classes :

La modélisation objet est utilisée dans le langage UML pour définir des objets-métiers et l’architecture de l’application. Ces objets sont créés en tant qu’instance de classe et s’interagissent dynamiquement pour offrir le comportement décrit par les cas d’utilisation. La modélisation objet définit le comportement requis par les différentes classes pour assurer la bonne mise en place des cas d’utilisation et des règles de gestion.

Les objets constituent la base de l’architecture des applications, ils peuvent être réutilisés à travers des domaines d’application ou encore être identifiés et dérivés directement des cas d’utilisation ou des domaines d’application, une classe est composée de :
Attributs : représentant des données dont les valeurs représentent l’état de l’objet.
La méthode : il s’agit des opérations applicables aux objets.

Après avoir réalisé le diagramme des cas d’utilisation, nous pouvons dégager les classes ainsi leurs méthodes et leurs attributs qui sont présentés dans le tableau suivant :

N°	Nom Classe	Liste des Attributs	Méthodes
01	Sensor	idSensor	addSensor(string nom)

		actualHour	deleteSensor(short id)
		actualDay	listSensor()
		valuesOfDay	findSensor(short id)
		minOfWeek	saveValue(short id, double value)
		maxOfWeek	
		avgOfWeek	findValues(short id)
			findMins(short id)
			findMaxs(short id)
			findAvgs(short id)

Tableau 3: Description des entités

Conclusion :

Tout au long de ce chapitre nous avons mené une conception détaillée du système d'information selon une approche objet afin de garantir la fiabilité et l'efficacité de la phase de réalisation de l'application.
Nous avons dressé une liste des acteurs constituants le système en exprimant leurs besoins, puis nous l'avons détaillé en précisant comment les objets et les acteurs doivent collaborer ensemble selon une dimension temporelle.
Finalement, nous avons décrit l'aspect statique avec les diagrammes des classes.
A l'aide de l'étude de notre cas nous avons déterminé l'environnement de développement de notre application qui sera présentée dans le chapitre suivant.

Chapitre 4: Réalisation du projet

Introduction

Dans le chapitre précédant, la modélisation fonctionnelle, statique et dynamique établie par le biais du formalisme de modélisation UML, nous a permis de réaliser les différents diagrammes d'analyse permettant la mise en œuvre de l'application, par ailleurs, ce chapitre illustre la justification du choix de la technologie utilisée, par la suite, il décrit les différents outils de développement choisis pour l'élaboration pratique du projet et enfin, on expose le travail réalisé à travers les captures écrans des interfaces de l'application réalisée.

1. Choix de la technologie :

1. Présentation de la technologie java EE:

La technologie Java est à la base de la plupart des applications en réseau, elle est exploitée dans le monde entier pour développer et fournir des applications mobiles et imbriquées, des jeux, du contenu Web et des logiciels d'entreprise. Utilisée par plus de 9 millions de développeurs dans le monde, la technologie Java permet de développer, de déployer et d'utiliser efficacement des applications et des services fascinants.

Des ordinateurs portables aux centres de données, des consoles de jeux aux superordinateurs scientifiques, des téléphones portables à Internet, la technologie Java est présente sur tous les fronts !

Des chiffres sur l'utilisation de JAVA :

97 % des bureaux d'entreprise exécutent Java ;
89 % des bureaux (ou ordinateurs) des Etats-Unis exécutent Java ;
9 millions de développeurs Java dans le monde ;
Choix n° 1 des développeurs ;
Plate-forme de développement n° 1 ;
3 milliards de téléphones mobiles exécutent Java ;
100 % des lecteurs Blu-ray livrés avec Java ;
5 milliards de cartes Java utilisées ;
125 millions de périphériques TV exécutent Java ;
Les 5 fabricants d'équipement d'origine principaux fournissent Java ME.

La technologie Java a été testée, ajustée, étendue et mise à l'épreuve par une communauté dédiée de développeurs, d'architectes et de passionnés de Java. Elle a été conçue pour permettre le développement d'applications portables hautes performances sur une large gamme de plates-formes informatiques. Grâce à la mise à disposition d'applications dans des environnements hétérogènes, les entreprises peuvent proposer davantage de services et dynamiser la productivité, la communication et la collaboration de l'utilisateur final, tout en réduisant considérablement le coût de propriété des applications d'entreprise et grand public. Java est aujourd'hui devenue un outil indispensable qui permet aux développeurs :

d'écrire des logiciels sur une plate-forme et de les exécuter sur pratiquement toutes les autres plates-formes,

de créer des programmes qui peuvent être exécutés dans un navigateur Web et accéder aux services Web disponibles,
de développer des applications côté serveur pour des forums, des magasins et des sondages en ligne, pour le traitement de formulaires HTML, etc.,
de combiner des applications ou des services basés sur le langage Java pour créer des applications ou des services très personnalisés,
d'écrire des applications puissantes et efficaces pour les téléphones portables, les processeurs à distance, les microcontrôleurs, les modules sans fil, les capteurs, les passerelles, les produits de consommation et tous les autres types de dispositif électronique.

Java EE est un environnement Java indépendant de toute plate-forme qui sert à développer, fabriquer et déployer des applications d'entreprise Web en ligne. Java EE comprend de nombreux composants de Java Standard Edition (Java SE). La plate-forme Java EE est constituée d'un jeu de services, d'API et de protocoles qui permettent de développer des applications Web multiniveaux.

Les développeurs pour entreprises ont besoin de Java EE car l'écriture d'applications professionnelles distribuées n'est pas une tâche aisée. Ils ont besoin d'une solution hautement performante qui leur permette de se concentrer exclusivement sur l'écriture de la logique d'entreprise tout en disposant d'une gamme complète de services professionnels tels que des objets distribués transactionnels, des intergiciels orientés vers les messages et des services de répertoire et de nom.

1. Présentation du framework Spring :

Le développement d’une application web directement avec Java EE en respectant les bonnes pratiques de la programmation s’avère une tâche fastidieuse est presque impossible à réaliser vue la contrainte du temps réserver pour le projet.

C’est pourquoi l’utilisation d’un framework va nous faire gagner du temps et nous permettre de développer une application en respectant les normes.

D’abord c’est quoi un framework ?

Un framework est un cadre facilitant le développement, il s’agit généralement d’un ensemble d’API masquant la complexité d’autres API sous-jacents .Ils existent beaucoup de framwok basés sur le langage java sur le marché tels que Chrysalis, Echo 2, Echo, Expresso Framework, Jaffa, Jucas, Maverik, Millstone, Spring, MVC, Struts , Tapestry, Turbine, VRaptor, Wicket, ...

Mais Spring reste le plus utilisé par plusieurs sociétés à travers le monde

C’est quoi Spring ?

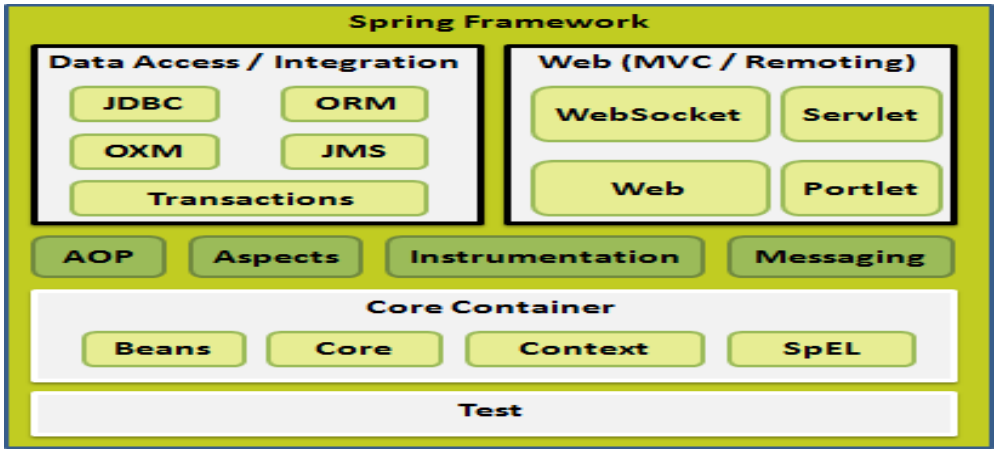


Figure 7 : Modules du framework Spring

Spring est considéré comme un conteneur dit « léger ». La raison de ce nommage est expliquée par Erik Gollot dans l'introduction du document «Introduction au framework Spring».

Spring est effectivement un conteneur dit « léger », c'est-à-dire une infrastructure similaire à un serveur d'applications J2EE. Il prend donc en charge la création d'objets et la mise en relation d'objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces objets. Le gros avantage par rapport aux serveurs d'application est qu'avec Spring, les classes n'ont pas besoin d'implémenter une quelconque interface pour être prises en charge par le framework (au contraire des serveurs d'applications J2EE et des EJBs). C'est en ce sens que Spring est qualifié de conteneur « léger ».

Spring s'appuie principalement sur l'intégration de trois concepts clés :

1. **L'inversion de contrôle est assurée de deux façons différentes: la recherche de dépendances et l'injection de dépendances ;**
2. **La programmation orientée aspect ;**
3. **Une couche d'abstraction.**

Pour l'inversion de contrôle, la recherche de dépendance consiste pour un objet à interroger le conteneur, afin de trouver ses dépendances avec les autres objets. C'est un cas de fonctionnement similaire aux EJBs.

Alors que l'injection de dépendances peut être effectuée de trois manières possibles:

L'injection de dépendance via le constructeur.
L'injection de dépendance via les modificateurs (setters).
L'injection de dépendance via une interface.

Les deux premières sont les plus utilisées par Spring.

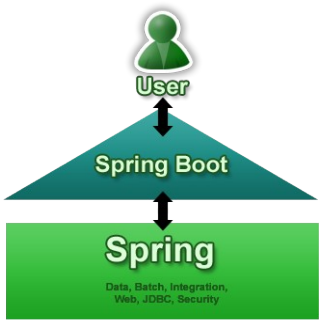
En ce qui concerne la couche d'abstraction, elle permet d'intégrer d'autres frameworks et bibliothèques avec une plus grande facilité. Cela se fait par l'apport ou non de couches

d’abstraction spécifiques à des frameworks particuliers. Il est ainsi possible d’intégrer un module d’envoi de mails plus facilement.

Ce framework, grâce à sa couche d’abstraction, ne concurrence pas d’autres frameworks dans une couche spécifique d’un modèle architectural Modèle-Vue-Contrôleur mais s’avère un framework multi-couches pouvant s’insérer au niveau de toutes les couches; modèle, vue et contrôleur. Ainsi il permet d’intégrer **Hibernate** ou **iBATIS** pour la couche de persistance ou encore Struts et JavaServer Faces pour la couche présentation.

Mais malheureusement Spring est connu pour sa configuration qui peut s’avérer complexe et fastidieuse. Il n’était pas rare de passer plusieurs jours sur la configuration d’un projet Spring notamment pour des personnes novices dans l’utilisation du framework. Fort de ce constat, les équipes de Spring décidèrent de travailler sur un projet permettant de faciliter le développement d’application Spring pour les développeurs. C’est ainsi qu’est né **Spring Boot**.

Figure 8: Schéma du Spring Boot



Spring Boot est un projet ou un micro framework qui a notamment pour but de faciliter la configuration d’un projet Spring et de réduire le temps alloué au démarrage d’un projet.
Tout est déjà configuré pour le développeur Spring, il ne lui reste qu'à se concentrer sur le code métier.

1. Architectures logiciels et outils de développement:

1. Architecture logiciel de l’application :

Notre application est basée sur un modèle en couche comme le montre la figure suivante:

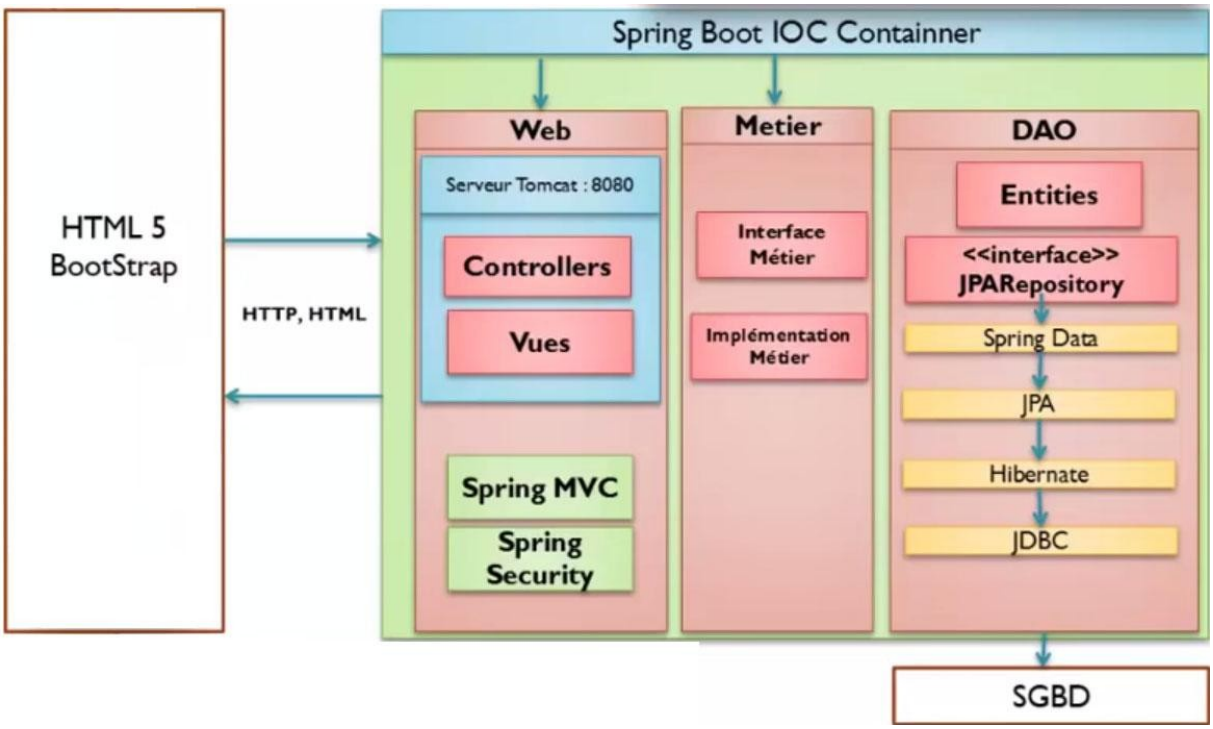


Figure 9: Architecture logiciel de l'application

En effet, c'est l'architecture à trois niveaux ou architecture à trois couches qu'est l'application du modèle plus général multi-tiers.

Son nom provient de l'anglais 'tier' signifiant étage ou niveau. Il s'agit d'un modèle logique d'architecture applicative qui vise à modéliser une application comme un empilement de trois couches logicielles (étages, niveaux, tiers ou strates) dont le rôle est clairement défini:

La présentation des données (**couche WEB**) : correspondant à l'affichage, la restitution sur le poste de travail, le dialogue avec l'utilisateur;
Le traitement métier des données (**couche Métier**) : correspondant à la mise en œuvre de l'ensemble des règles de gestion et de la logique applicative ;
L'accès aux données persistantes (**couche DAO –Data Access Objet)** : correspondant aux données qui sont destinées à être conservées sur la durée, voire de manière définitive.

2. Présentation des modules Spring utilisés:

a) Spring Data JPA:

Spring Data facilite l'écriture des couches d'accès aux données et tente d'offrir une abstraction commune pour l'accès aux données quelques soient les sources de données sous-jacentes, tout en prenant en compte les spécificités de celles-ci
Spring Data JPA offre une couche d'abstraction supplémentaire par rapport à JPA et se charge de l'implémentation des fonctionnalités les plus courantes des DAO.
Dans notre projet, on se concentre sur l'essentiel : l'écriture des requêtes, comme c'est indiqué dans la figure ci-dessous:
Pour l'intégrer, il suffit d'ajouter une dépendance dans Maven :
<dependency>

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Spring Data JPA utilise Hibernate qui est un framework ORM (Object-Relational-Mapping) pour assurer le requetage en base des données. Le concept d'ORM (Object-Relational-Mapping) est un concept de persistance permettant de transformer de manière transparente un objet en données au sein d'un SGBDR et vice versa.

a) **Conteneur Spring IOC :**

Le conteneur IOC est au cœur du framework Spring, il est responsable d'instancier, configurer et assembler les objets, gérer leur cycle de vie complet de la création jusqu'à la destruction. Il utilise l'injection de dépendance (DI) pour gérer les éléments qui composent une application.

Le conteneur reçoit les instructions sur les objets à instancier, configuré, assemblé par la lecture des métadonnées de configuration fournis. Les métadonnées de configuration peuvent être représentées soit par XML, annotations Java, ou un code Java. Le conteneur Spring IOC rend l'utilisation des classes Java POJO et les métadonnées de configuration pour produire un système ou une application entièrement configurée et exécutable.

Il existe deux types de conteneurs IOC :

BeanFactory: Interface qui existe dans le package org.springframework.beans.factory.BeanFactory
ApplicationContext : Interface qui existe dans le package org.springframework.context.ApplicationContext

BeanFactory et ApplicationContext sont des interfaces qui agissent comme des conteneurs IoC.

L'interface ApplicationContext est construite au-dessus de l'interface BeanFactory. Il ajoute quelques fonctionnalités supplémentaires que BeanFactory tels que l'intégration simple avec l'AOP de Spring, la couche application contexte spécifique (par exemple WebApplicationContext) pour les applications Web. Il est donc préférable d'utiliser ApplicationContext que BeanFactory.

3. Outils de développement utilisés:

a) **Eclipse IDE for java EE**

Eclipse est un environnement de développement intégré (Integrated Development Environment) dont le but est de fournir une plate-forme modulaire pour permettre de réaliser des développements informatiques.

b) **Maven :**

Maven est un outil d'automatisation de construction utilisé principalement pour des projets Java. Maven aborde deux aspects grands aspects: il décrit comment le logiciel est construit, et deuxièmement, il décrit ses dépendances.

c) **MySQL :**

MySQL est un serveur de bases de données relationnelles Open Source, il stocke les données dans des tables séparées plutôt que de tout rassembler dans une seule table. Cela améliore la rapidité et la souplesse de l'ensemble.

Les tables sont reliées par des relations définies, qui rendent possible la combinaison de données entre plusieurs tables durant une requête.
Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données.

Conclusion

Dans ce chapitre, au départ, nous avons mis en avant les outils de développement utilisés pour la programmation de l’application par la suite nous avons présenté les modules et architectures du framework Spring et nous avons clôturé par la présentation des écrans de l’application réalisée.

Chapitre: Annexe

1. Modélisation fonctionnelle des données

Un modèle d'analyse livre une spécification complète des besoins issus des cas d'utilisations et les structures sous une forme qui facilite la compréhension (scénarios) que nous avons présentés dans les descriptions textuelles et les diagrammes de séquences.
L'étude de conception est la phase la plus importante du cycle de développement d'un système informatique. En effet, elle permet de confronter la spécification et l'analyse avec l'implémentation, elle présente le point de convergence des deux aspects :
« Le quoi faire » (analyse) et le « comment faire » (réalisation).

1. Choix du formalisme de conception

Dans la cadre de notre projet, nous avons opté pour le langage UML comme une approche de conception. Ci-dessous, nous présentons ce langage puis nous justifions notre choix.

UML (Unified Modeling Language) est un langage formel et normalisé en termes de modélisation objet. Son indépendance par rapport aux langages de programmation, aux domaines de l'application et aux processus, son caractère polyvalent et sa souplesse ont fait de lui un langage universel. En plus UML est essentiellement un support de communication, qui facilite la représentation et la compréhension de solution objet. Sa notation graphique permet d'exprimer visuellement une solution objet, ce qui facilite la comparaison et l'évaluation des solutions. L'aspect de sa notation, limite l'ambigüité et les incompréhensions.

UML fournit un moyen astucieux permettant de représenter diverses projections d'une même représentation grâce aux vues.

Une vue est constituée d'un ou plusieurs diagrammes. On distingue trois types de vues :

- **Vue fonctionnelle :** donnant une vue globale de notre projet.
  - **Diagrammes de cas d'utilisation :** identifient les utilisateurs du système (acteurs) et leurs interactions avec le système.
- **La vue statique**: permettant de représenter le système physiquement :
  - **Diagrammes de classes :** représentent des collections d'éléments de modélisation statique (classes, paquetages...), qui montrent la structure d'un modèle.
  - **Diagrammes d'objets :** ces diagrammes montrent des objets (instances classes dans un état particulier) et des liens (relations sémantiques) entre objets.
  - **Diagrammes de composants :** permettent de décrire l'architecture physique statique d'une application en termes de modules : fichiers sources, librairie exécutables, etc.
- **La vue dynamique :** montrant le fonctionnement du système :
  - **Diagrammes de collaboration :** montrent des interactions entre objet (instances de classes et acteurs).
  - **Diagrammes de séquence :** permettent de représenter des collaborations en objets selon un point de vue temporel, on y met l'accent sur la chronologie (envois de messages).
  - **Diagrammes d'états-transitions** : permettent de décrire les changements d'états d'un objet ou d'un composant, en réponse aux interactions avec d'autres objets/composants ou avec des acteurs.
  - **Diagrammes d'activités** :(une variante des diagrammes d'états-transitions) servent à représenter graphiquement le comportement d'une méthode ou déroulement d'un cas d'utilisation.

La conception de notre projet a été élaborée en suivant la démarche suivante :

- L'élaboration des diagrammes de cas d'utilisation. Cette étape a été réalisée suite à la spécification fonctionnelle de l'application.
- Dresser les diagrammes de séquences pour mettre en évidence les interactions entre les différents objets du système.
- Recensement des classes candidates et élaboration du diagramme des classes.
- Elaborer le diagramme d'activité pour montrer les différentes activités pour les méthodes des objets.

1. Diagramme de cas d'utilisation :

Les cas d'utilisation décrivent un ensemble d'actions réalisées par le système, en réponse à une action d'un acteur.

a) **Identification des acteurs :**

Dans notre cas les acteurs sont :
Les utilisateurs qui peuvent utiliser le système et faire des opérations d'insertion, de suppression, et de modification des formations.
Parmi ces utilisateurs il y a l'administrateur, qui possédera tous les droits dont celui de pouvoir ajouter, modifier et supprimer les données des formations ou bien des autres utilisateurs.

Un acteur peut être un :
- ✓ Administrateur
- ✓ Personnel

Un Personnel peut être un :
- Secrétaire

a) **Identification des cas d'utilisations :**

Nous décrivons pour chaque acteur les cas d'utilisation. On distingue les cas d'utilisation suivants :

Acteur	**Rôle**
Administrateur	• S'authentifier • Manipuler le système • Rechercher dans l'archive • Gérer les formations (ajouter, supprimer, modifier) • Gérer les personnels du centre (ajouter, supprimer, modifier) • Gérer les profs et les étudiants (ajouter, supprimer, modifier) • Gérer les services des salles (ajouter, supprimer, modifier) • Gérer des charges (ajouter, supprimer, modifier) • Consulter les charges
Secrétaire	• S'authentifier • Rechercher dans l'archive • Ajouter/Modifier nouveau étudiant/ prof • Ajouter/Modifier les Formations • Consulter les charges • Gérer les services des salles.

Tableau 1: Liste des rôles des acteurs du diagramme des cas d'utili sation

Dans la figure ci-dessous, on montre le diagramme des cas d'utilisation :

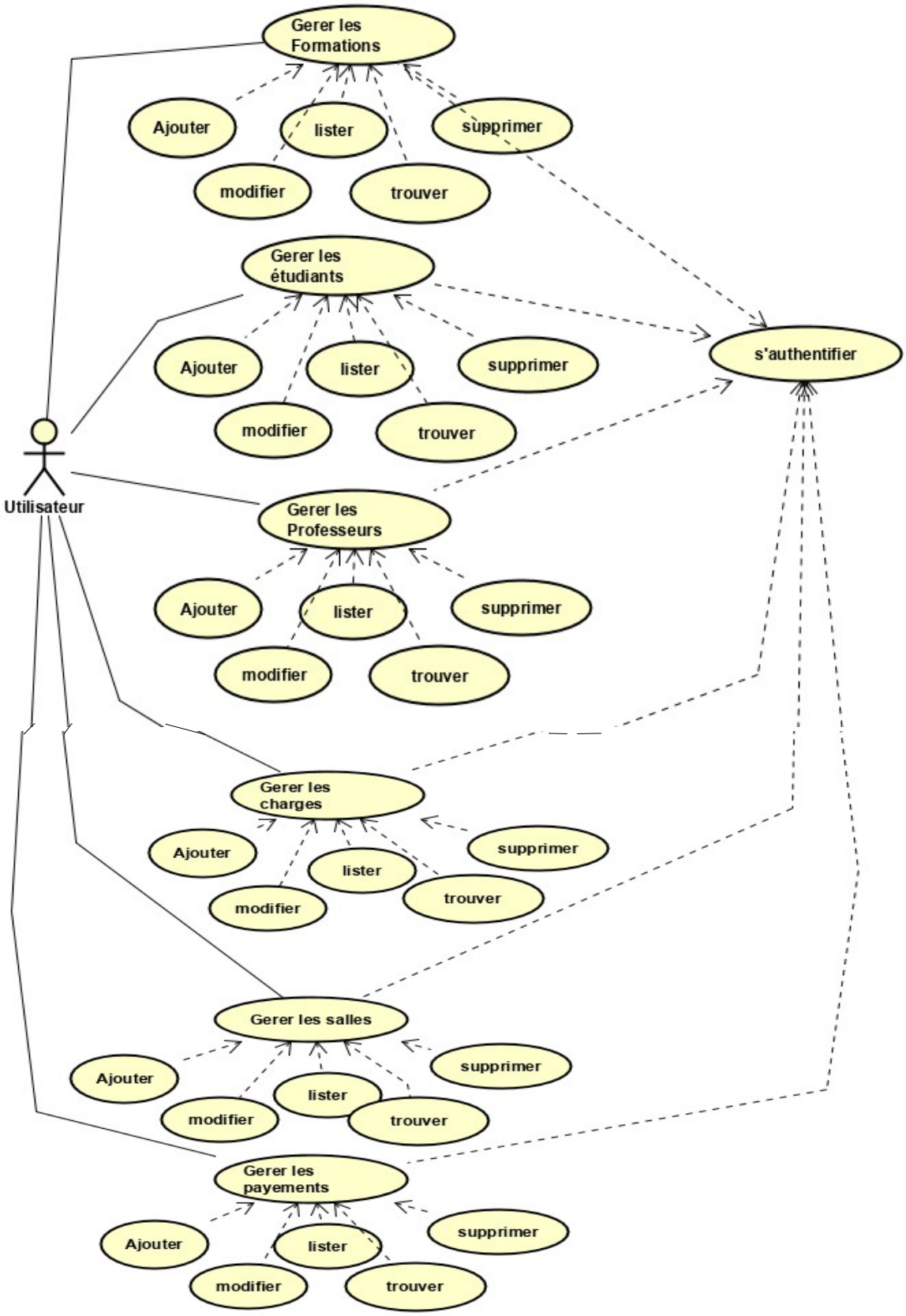


Figure 1: Diagramme des cas d'utilisation

1. Modélisation statique des données :

La modélisation des données permet de dégager l'ensemble des données manipulées en vue d'élaborer le diagramme de classes. En effet, ce dernier donne une vue statique du système. Il décrit les types et les objets du système.

Il s'agit donc d'une représentation des données du champ de l'étude ainsi que le lien sémantique entre ces données, facilement compréhensible, permettant de décrire le système d'information à l'aide des concepts proposés par le modèle UML.

1. Représentation des classes :

La modalisation objet est utilisée dans le langage UML pour définir des objets-métiers et l'architecture de l'application. Ces objets sont créés en tant qu'instance de classe et s'interagissent dynamiquement pour offrir le comportement décrit par les cas d'utilisation. La modélisation objet définit le comportement requis par les différentes classes pour assurer la bonne mise en place des cas d'utilisation et des règles de gestion.

Les objets constituent la base de l'architecture des applications, ils peuvent être réutilisés à travers des domaines d'application ou encore être identifiés et dérivés directement des cas d'utilisation ou des domaines d'application, une classe est composée de :
Attributs : représentant des données dont les valeurs représentent l'état de l'objet.
La méthode : il s'agit des opérations applicables aux objets.

Après avoir réalisé le diagramme des cas d'utilisation, nous pouvons dégager les classes ainsi leurs méthodes et leurs attributs qui sont présentés dans le tableau suivant :

N°	Nom Classe	Liste des Attributs	Méthodes
01	User	idUser Login Password Droit	connecter() ajouterUser() modifierUser() supprimerUser()
02	Personne	idPerson Nom Prenom adresse Tel	
03	Etudiant	Photo dateNaissance dateInscription montantPayer Payer	ajouterEtud() modifierEtud() listerEtud() trouverEtud() supprimerEtud()
04	Prof	Pourcentage montantRemun Remun	ajouterProf() modifierProf() listerProf() trouverProf() supprimerProf()
05	Payement	idPayement datePayement moisPayer Etudiant	ajouterPayement() listerPayement() supprimerPayement()
06	Formation	idFormation nomFromation Type	ajouterFormation() modifierFormation() listerFormation()

		dateDebut	trouverFormation() supprimerFormation()
		Duree	
		idProf	
		idSalle	
07	Salle	idSalle	ajouterSalle() listerSalle()
		Numero	
		Capacité	
08	Charge	idCharge	ajouterCharge() modifierCharge() listerCharge() trouverCharge() supprimerCharge()
		nomCharge	
		typeCharge	
		dateCharge	
		montant	

Tableau 2: Description des entités

2. Représentation des associations entre les classes :

Les associations sont des relations entre classes, elles représentent un lien durable ou ponctuel entre deux objets, une appartenance, ou une collaboration, elles sont représentées par une ligne entre les classes.
Le modèle de données d'UML comprend trois associations génériques principales :
Généralisation, association et dépendance, à partir de ces trois associations de base, nous représentons ainsi les différents types d'association qui décrivent les dépendances entre les classes déjà citées.
Les cardinalités : précisent combien d'objets de classe considérée peuvent être liées à un objet de l'autre classe. Le tableau suivant illustre une représentation de ces cardinalités :

Cardinalités	**Désignation**
1	Un et un seul
0.1	Zero ou un
N	Entier naturel
m.n	De m à n
0.*	De 0 à plusieurs
1.*	De 1 à plusieurs

Tableau 3: Désignation des cardinalités du diagramme des classes

La figure ci-dessous récapitule les tableaux précédents dans un diagramme de classes qui contient toutes les informations telles que les classes, les méthodes, les associations et les propriétés.

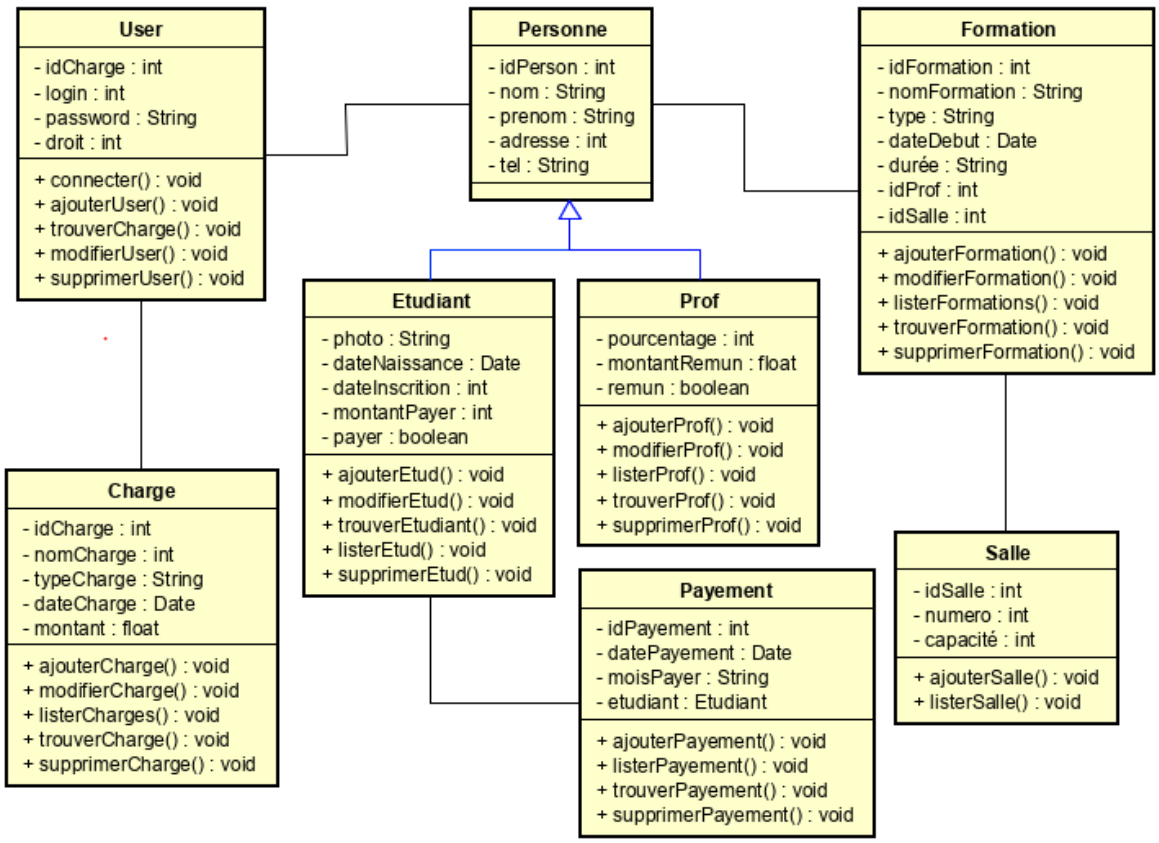


Figure 2: Diagramme des classes

2. Modélisation dynamique des données

2.1 Diagramme d'activité :

Les diagrammes d'activités permettent de mettre l'accent sur les traitements des données, ils sont donc particulièrement adaptés à la modélisation du cheminement de flots de contrôle et de flots de données. Ils permettent ainsi de représenter graphiquement le comportement d'une méthode ou le déroulement d'un cas d'utilisation.
Les diagrammes d'activités sont relativement proches des diagrammes d'états-transitions dans leur présentation, mais leur interprétation est sensiblement différente.
Une activité représente une exécution d'un mécanisme, un déroulement d'étapes séquentielles. Le passage d'une activité vers une autre est matérialisé par une transition.
Les transitions sont déclenchées par la fin d'une activité et provoquent le début immédiat d'une autre, dans la figure suivante, nous présentons un exemple du diagramme d'activité authentification, les autres diagrammes sont présentés en annexes du rapport.

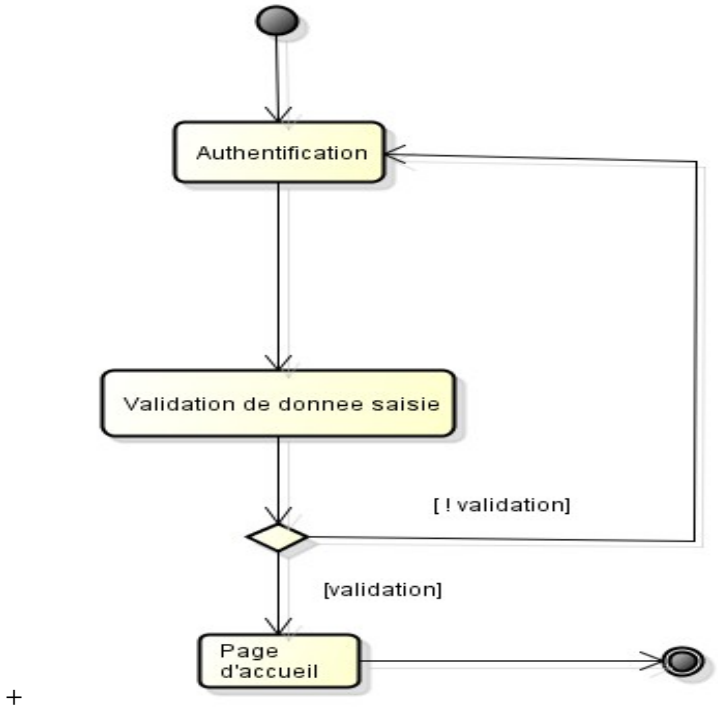


Figure 3: Diagramme d'activité d'authentification

Conclusion :

Tout au long de ce chapitre nous avons mené une conception détaillée du système d'information selon une approche objet afin de garantir la fiabilité et l'efficacité de la phase de réalisation de l'application.
Nous avons dressé une liste des acteurs constituants le système en exprimant leurs besoins avec les diagrammes de cas d'utilisation, puis nous l'avons détaillé en précisant comment les objets et les acteurs doivent collaborer ensemble selon une dimension temporelle par l'utilisation des diagrammes de séquence. Finalement, nous avons décrit l'aspect statique avec les diagrammes des classes.
A l'aide de l'étude de notre cas nous avons déterminé l'environnement de développement de notre application qui sera présentée dans le chapitre suivant.