

# Introduction générale

Le développement des technologies d'information a participé d'une manière topique à l'amélioration de la productivité des différents services dans plusieurs secteurs, y compris le secteur éducatif, ce qui est le cas pour l'École Nationale des sciences de l'informatique ENSI.

D'ailleurs, l'ENSI supporte plusieurs services dont la plupart exigent une automatisation de leurs systèmes d'information.

Dans ce cadre, s'inscrit ce projet de conception et développement afin de pallier cette pénurie d'automatisation et dans lequel nous nous concentrerons sur le processus d'affectation des environnements aux enseignants qui se focalise sur la gestion des absences d'une part, et la gestion des séances de rattrapage d'autre part puisque trouver une salle libre à la date convenable pour assurer le rattrapage à un groupe donné n'est pas une tâche aisée tout en s'assurant d'informer les étudiants de toutes les modifications occasionnelles affectant leurs emplois du temps d'une manière préprogrammée.

Et ceci dans le but d'avoir une meilleure gestion des emplois du temps et une communication plus efficace entre les enseignants et leurs étudiants.

Afin d'exposer la totalité du travail achevé tout au long de la période consacrée à la réalisation du projet de conception et de développement nous avons opté de diviser le présent rapport en quatre chapitres. Le premier chapitre sera dédié à la présentation du projet dans laquelle on va élaborer une étude de l'existant. Le deuxième chapitre, afin d'analyser les besoins à remplir par l'application. En ce qui concerne le troisième chapitre il sera consacré à la conception du projet. Le quatrième chapitre décrira la réalisation de l'application et les différentes parties développées. Et enfin, nous terminons ce rapport par une conclusion générale, résumant notre travail et évoquant les probables perspectives de notre application.

# Présentation du projet

## 1.1 Introduction

Avant d'entamer la conception de notre projet, il est nécessaire d'avoir une vision claire de la démarche à suivre visant à proposer un plus vis-à-vis de l'existant, ajouter d'autres aspects, et améliorer les différentes fonctionnalités. Nous proposons ainsi une étude de l'existant partagée entre la stratégie d'affectation classique, totalement achevée par le responsable de l'administration, et celle qui adopte la solution Informatique. Ceci aurait un grand intérêt lors de la fixation des nouveaux apports à intégrer dans le but d'améliorer les conditions de la procédure d'affectation classique.

## 1.2 Étude de l'existant

Nous avons rencontré un responsable de l'administration pour nous renseigner un peu plus de la procédure classique d'affectation des salles disponibles en cas de rattrapage ou de devoir surveillé, de la manière de gestion des absences des enseignants , et de la procédure de renseignement des étudiants de toute modification affectant leur emploi du temps actuel. Et ceci dans le but d'avoir une idée sur les améliorations qu'il souhaite apporter.

## 1.3 Description de la procédure existante

A chaque fois qu'un enseignant à l'ENSI souhaite réaliser une séance de cours supplémentaire que ce soit pour faire un rattrapage ou pour passer un devoir surveillé hors de la semaine des DS il doit s'adresser à un responsable de l'administration afin de lui demander de chercher une salle disponible à la date souhaitée . En se basant sur les informations fournies par l'enseignant à savoir le jour et l'heure, Ce responsable cherche manuellement une salle disponible en consultant tous les emplois du temps de tous les groupes de classes. Le renseignement des étudiants de toute modification affectant leurs emplois du temps(Absence,Rattrapage) est assuré par un responsable de l'administration

qui rattache des notes dans le tableau d'affichage que les étudiants doivent consulter quotidiennement afin de rester au courant de tout changement.

### 1.4 Problèmes dégagés

La procédure d'affectation d'environnements aux enseignants précédemment décrite s'est bien avérée d'un an à un autre, inefficace et non raisonnable avec une perte de temps pour l'enseignant, l'étudiant et aussi pour le responsable d'administration.

Les défauts de la procédure classique se manifestent chaque jour sur le terrain impliquant non seulement un manque de communication entre l'administration et les autres acteurs(enseignants et étudiants), un retard dans les cours, un conflit lors de l'affectation des salles mais encore des efforts inéluctable fournis par les personnels de l'école responsables du déroulement de ce processus.

### 1.5 Solutions proposées

Notre application est conçue pour pallier tous les problèmes rencontrés tout au long du cursus d'affectation des environnements aux enseignants classique.

L'idée consiste à automatiser ce processus à partir d'une gestion efficace des données.

On propose les solutions suivantes :

- Permettre à l'enseignant de notifier ses étudiants de son absence sur l'application et par mail en mentionnant la date et la raison.
- Donner la possibilité à l'enseignant de trouver un horaire convenable et une salle libre à cet horaire afin de réaliser une séance de rattrapage ou de devoir surveillé et de notifier le groupe concerné au préalable par mail et sur l'application.
- Permettre à l'enseignant et à l'étudiant de visualiser leurs emploi du temps.

### 1.6 Méthode de développement

Selon des estimations, plus que 80 % des projets qui utilisent des méthodologies traditionnelles connaissent des retards et des dépassements parce que ces méthodologies visent à prédire la façon dont les procédures devraient se passer selon un planning préétabli. Notre projet suit le principe de développement itératif basé sur l'écoute du client, c'est pour cette raison que notre choix s'est orienté vers les méthodes agiles de développement et de gestion de projet.

#### 1.6.1 Méthode Agile

Le but des méthodes de développement : " méthodes agiles " est la réduction du cycle de vie du logiciel et par conséquent l'accélération de sa réalisation en développant tout d'abord une version minimale, ensuite en intégrant les différentes fonctionnalités par un

## CHAPITRE 1. PRÉSENTATION DU PROJET

---

processus itératif basé sur l'écoute du client et des tests tout au long du cycle de développement. L'origine des méthodes agiles est liée à l'instabilité de l'environnement technologique et au fait que le client est souvent incapable de définir ses besoins de manière définitive dès le début du projet. Le terme «agile» fait ainsi référence à la capacité d'adaptation aux changements de contexte et aux modifications de spécifications intervenant pendant le processus de développement. La traduction de la méthode Agile est la suivante :

- Donner de l'importance aux individus et interactions plutôt que processus et outils.
- développement logiciel au lieu de documentation lourde,
- Adaptation aux changements plutôt que suivi d'un plan rigide.

### 1.6.2 Méthode adoptée

Dans cette partie, nous allons présenter la méthode de développement adoptée dans notre projet. En effet, nous avons choisi de travailler avec la méthode Scrum car celle-ci répond bien aux caractéristiques des méthodes agiles, ainsi que pour les raisons suivantes : Scrum est utilisée afin de développer et tester les courtes itérations, elle donne la possibilité de produire la plus grande valeur métier dans la durée la plus courte et permet d'adapter le logiciel créé suivant l'évolution du projet.

### 1.6.3 Scrum

C'est une méthode de développement agile orientée projet informatique dont les ressources sont régulièrement actualisées. Elle est basée sur les principes des méthodes agiles. Comme toutes les méthodes agiles, elle privilégie la livraison rapide d'un prototype afin d'avoir un retour rapide des clients et des donneurs d'ordre pour minimiser les pertes de temps. N[1]

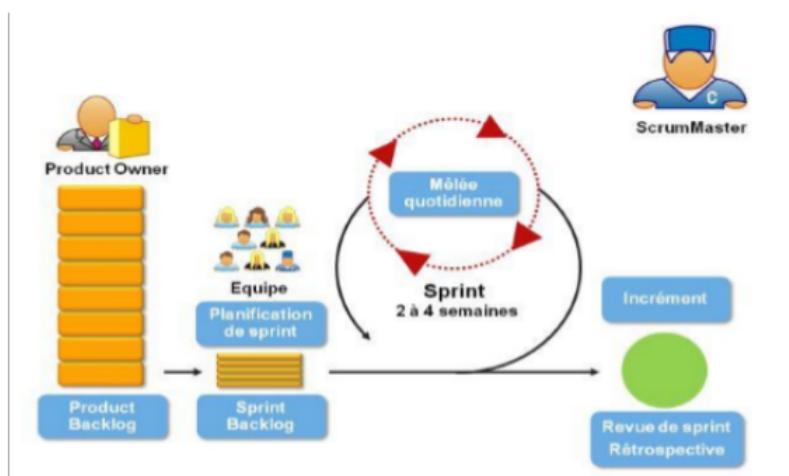


FIGURE 1.1 – Cycle de vie SCRUM N[2]

## 1.7 Conclusion

L'objectif de ce premier chapitre était principalement d'expliquer et de détailler les notions fondamentales à aborder dans notre sujet ainsi que la méthode de développement adoptée . Dans le second chapitre nous allons présenter notre perception d'alternatives dépassant les problèmes rencontrés dans le premier chapitre.

# Chapitre 2

## Spécification et Analyse

### Introduction

La phase d'analyse et spécification des besoins est la première phase du cycle de développement du projet au cours de laquelle on va identifier les besoins des utilisateurs. En effet c'est une étape primordiale dans la détermination de la qualité et des fonctionnalités qu'offre notre application.

### 2.1 Analyse des besoins

On va déterminer à ce niveau tous les besoins fonctionnels et non fonctionnels auxquels doit répondre cette application.

#### 2.1.1 Besoins fonctionnels

\*L'administrateur a comme fonctionnalité principale :

- Gérer les utilisateurs y compris les étudiants et les enseignants.

\*L'enseignant a la possibilité de :

- Créer un compte : Pour se faire l'enseignant doit remplir les différentes informations (nom,prénom, mot de passe...).
- Notifier ses étudiant de son absence en leurs fournissant les informations nécessaires à savoir,la date et la raison.

-Créer une proposition de séance de rattrapage ou de devoir surveillé en indiquant l'horaire souhaité .Le système vérifie ainsi la disponibilité , suggère des dates et salles dans le cas de non disponibilité de l'horaire et notifie l'enseignant et ses étudiants des détails du devoir surveillé ou du rattrapage .

-Consulter son emploi du temps.

\*L'étudiant a la possibilité de :

- Créer un compte : Pour se faire l'étudiant est sensé remplir certaines informations ( nom,prénom,groupe, mot de passe).

- Consulter son emploi du temps.
- Recevoir les notifications par mail et sur son espace dans l'application concernant les devoirs surveillés , les rattrapages ou les absences validées par son enseignant.

### 2.1.2 Besoins non fonctionnels

À part les besoins fondamentaux cités précédemment, notre système doit prendre en considération certaines contraintes additionnelles et répondre aux critères suivants :

\*L'extensibilité et la flexibilité :

Possibilité d'ajouter ou de modifier de nouvelles fonctionnalités. Ainsi l'application ne se limitera pas seulement aux services proposés au début de sa conception.

\*La sécurité et l'intégrité : L'application devrait avoir l'aptitude à protéger ses données contre les accès non autorisés. Autrement dit les données des utilisateurs devraient être protégées au maximum.

\*Temps de réponse et Rapidité :

il est clair que la vitesse d'affichage affecte fortement l'efficacité de notre application. C'est pourquoi l'application développée devrait fournir des résultats cohérents et efficaces et dans un temps négligeable suite aux différentes requêtes issues des utilisateurs.

\*L'ergonomie :

L'application devrait présenter une interface graphique simple et facile à manipuler.

## 2.2 Spécification des besoins

À ce niveau , nous allons modéliser notre application à partir des cas d'utilisation qui permettent de donner une vision globale du comportement fonctionnel d'un système et de représenter une unité discrète des interactions entre les utilisateurs(les acteurs) et le système.

### 2.2.1 Définition des acteurs

On distingue 3 acteurs :

**-Étudiant :** c'est un acteur bénéficiaire de quelques services offerts par cette application. En effet, il peut consulter son propre emploi du temps, il reçoit des notifications par mail et sur l'application web qui servent à lui informer de tout changement affectant son emploi du temps.

**-Enseignant :** L'enseignant est l'acteur principal de notre système qui a la possibilité d'accéder à l'application afin de Créer une proposition de rattrapage ou de devoir surveillé en choisissant le groupe et l'horaire qui lui convient. Il peut aussi informer ses étudiants par avance de son absence pour une date donnée. En plus il peut consulter son emploi du temps et son accueil.

**-Administrateur :** C'est un acteur qui effectue les opérations CRUD(Create,read, update, delete) sur les entités étudiant et enseignant.

## 2.2.2 Diagrammes de spécification

### 2.2.2.1 Diagramme des cas d'utilisation général

Ce diagramme représente l'ensemble des cas d'utilisation générales de notre application en s'appuyant sur les différents fonctionnalités exigés ,ainsi que l'interaction entre les acteurs et ces cas d'utilisation d'une manière formelle .

Comme le montre la figure suivante , on trouve 3 types d'acteurs mis en jeu possédant chacun un ensemble de cas d'utilisation qui le relie avec le système et qui sont respectivement :

Administrateur, enseignant et étudiant.

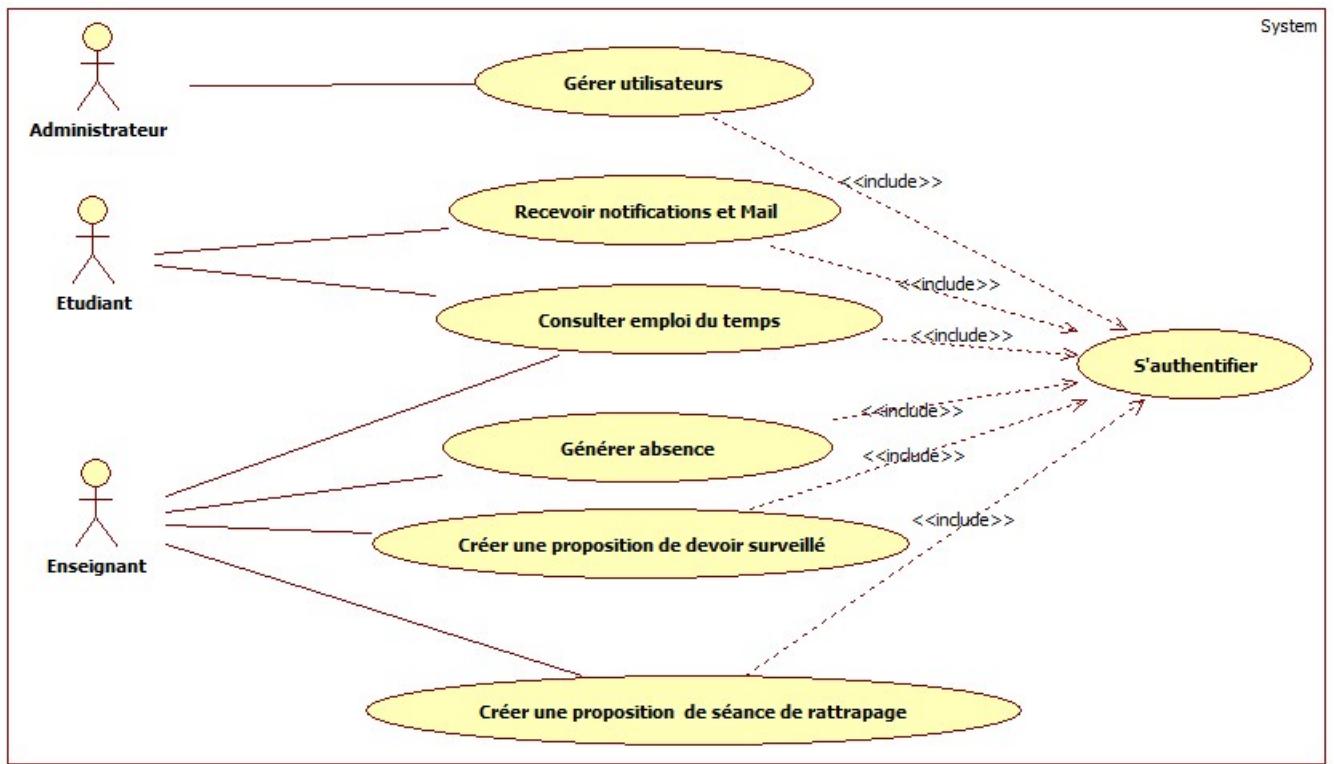


FIGURE 2.1 – Diagramme de cas d'utilisation général

### 2.2.2.2 Diagramme de cas d'utilisation raffiné

#### -Diagramme de cas d'utilisation «S'authentifier»

La figure 2.2 montre le raffinement du cas d'utilisation «S'authentifier».

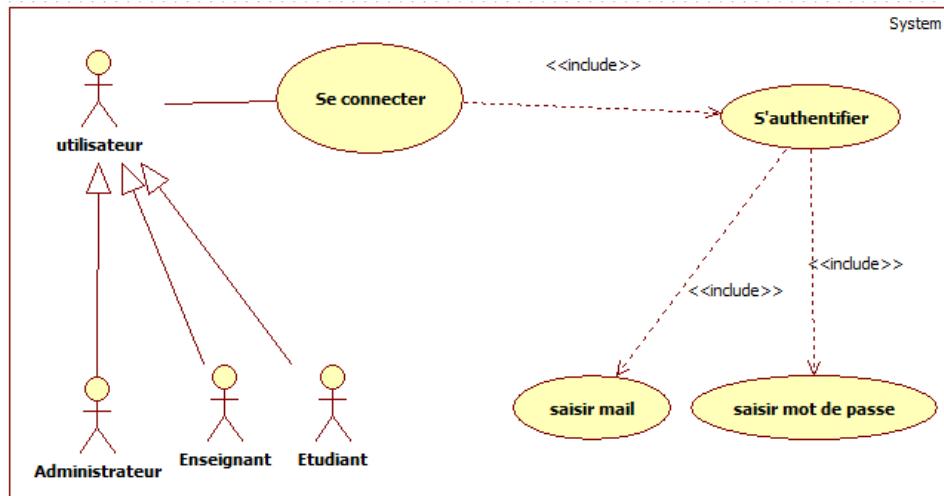


FIGURE 2.2 – Raffinement de cas d'utilisation «S'authentifier»

#### Description textuelle du cas d'utilisation «S'authentifier».

Le tableau suivant contient la description du cas d'utilisation «S'authentifier».

Nom	Authentification
Acteur	Utilisateur
Pré condition	utilisateur déjà inscrit
Post condition	utilisateur connecté
Scénario nominal	1) L'utilisateur clique sur le lien connection. 2) Le système génère la page d'authentification. 3)L'utilisateur saisit les droits d'accès (mail, mot de passe). 4)Le système vérifie les informations entrées par l'utilisateur par rapport à celles qu'il a dans sa base de données. 5)Le système redirige l'utilisateur à la page principale de l'application.
Scénario alternatif	Non-validité des coordonnées : Dans le cas où le mot de passe saisi est erroné, le système affiche un message d'erreur et demande à l'utilisateur de saisir à nouveau son mot de passe, ou bien lorsque l'utilisateur fournit un mail inconnu, le système retourne un message d'erreur et demande à l'utilisateur d'entrer un autre mail

TABLE 2.1 – Description du cas d'utilisation «S'authentifier».

-Diagrammes de cas d'utilisation de l'administrateur :

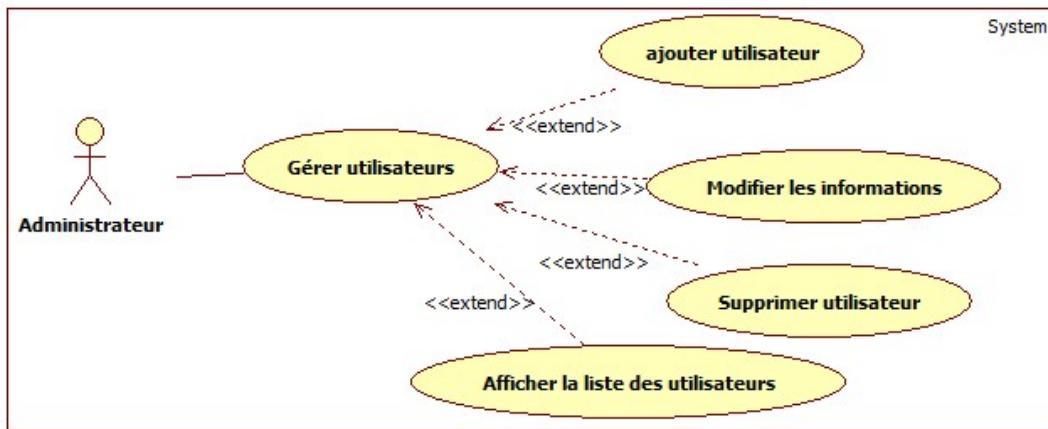


FIGURE 2.3 – Diagramme de cas d'utilisation de l'Administrateur

Résumé	L'admin a la possibilité d'ajouter plusieurs utilisateurs(étudiants, enseignants).
Pré conditions	Administrateur authentifié
Post conditions	Utilisateur pouvant se connecter à l'application avec son nouveau compte.
Scénario nominal	1)L'admin affiche la liste de tous les utilisateurs inscrits. 2)Il clique sur le bouton ajouter 3)Il saisit les différents paramètres du user (nom, prénom, mail, mot de passe). 4)le système valide l'ajout
Scénario alternatif	Exception 1 :Si les champs sont vides, l'ajout ne sera pas effectué car l'opération est impossible Exception 2 :Si l'utilisateur est déjà existant, le système affiche un message d'erreur

TABLE 2.2 – Scénario du cas d'utilisation « Ajouter »

Résumé	L'administrateur peut modifier les informations des utilisateurs ayant chacun un rôle spécifique
Pré conditions	Utilisateur existant et administrateur authentifié
Post conditions	Utilisateur modifié
Scénario nominal	1)L'admin affiche la liste de tous les utilisateurs inscrits. 2)Il clique sur le bouton modifier 3)Il saisit les nouvelles informations. 4)Le système valide la modification

TABLE 2.3 – Scénario du cas d'utilisation « Modifier »

## CHAPITRE 2. SPÉCIFICATION ET ANALYSE

---

Résumé	L'administrateur peut supprimer un utilisateur
Pré condition	Utilisateur existant et administrateur authentifié
Post condition	Utilisateur supprimé
Scénario nominal	1) L'admin affiche la liste de tous les utilisateurs inscrits. 2) Il clique sur le bouton supprimer 3) Le système valide la suppression

TABLE 2.4 – Scénario du cas d'utilisation « Supprimer »

-Diagrammes de Cas d'utilisation de l'enseignant :

\* Générer absence :

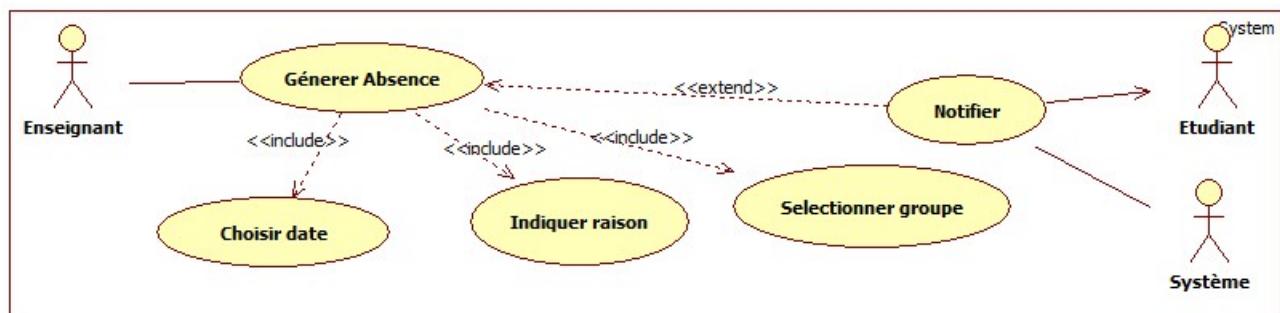


FIGURE 2.4 – Diagramme de cas d'utilisation « Générer absence »

Description textuelle du cas d'utilisation « Générer absence » :

L'enseignant a la possibilité de créer une absence. Pour se faire il doit indiquer la date, la raison et le groupe. Ensuite l'étudiant sera notifié par le système et par mail.

\* Créer une proposition de séance de rattrapage :

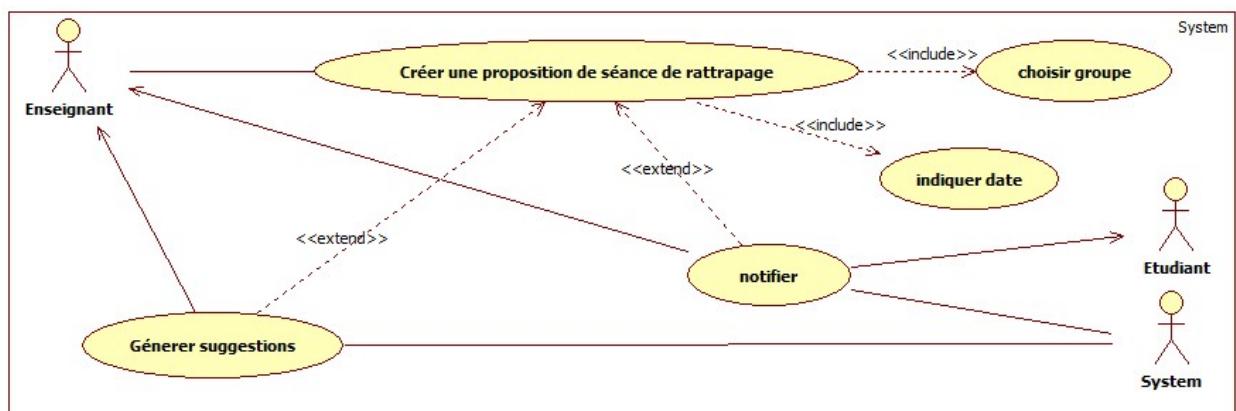


FIGURE 2.5 – Diagramme de cas d'utilisation « Créer une proposition de séance de rattrapage »

## CHAPITRE 2. SPÉCIFICATION ET ANALYSE

---

### Description textuelle du cas d'utilisation « Créer une proposition de séance de rattrapage » :

L'enseignant a la possibilité de créer une proposition de séance de rattrapage après avoir choisi la date et le groupe . S'il y a un horaire et une salle disponible , le système informe l'enseignant et les étudiants avec un mail , ceux derniers seront aussi notifiés dans leurs espace. Sinon dans le cas où l'horaire n'est pas disponible le système va générer un ensemble de suggestions parmi lesquels l'enseignant peut choisir une.

Remarque :Le même scénario s'effectue dans le cas de création de proposition d'un devoir surveillé.

### -Diagramme de Cas d'utilisation de l'étudiant :

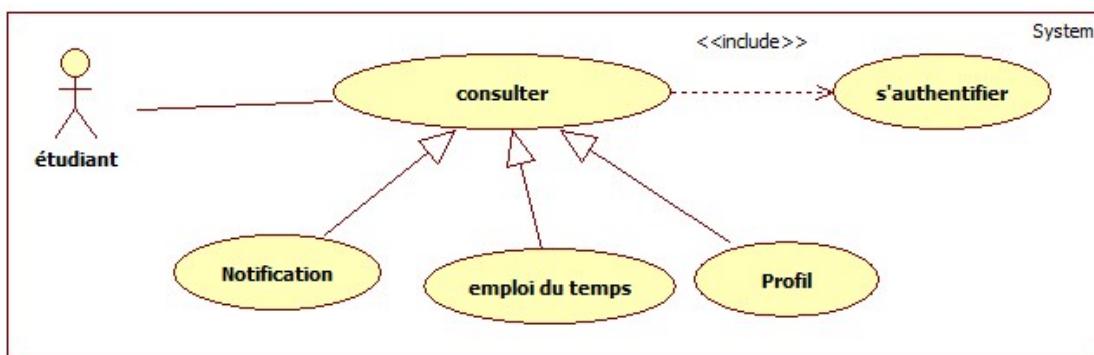


FIGURE 2.6 – Diagramme de cas d'utilisation « Consulter espace étudiant »

### Description textuelle du cas d'utilisation « Consulter espace étudiant

L'étudiant accède à l'application pour consulter soit son emploi du temps, son profil contenant ses coordonnées ou ses notifications . En effet à chaque fois qu'un enseignant crée une proposition de rattrapage ou de devoir surveillé ou génère une absence les étudiants seront notifiés automatiquement.

### 2.2.2.3 Diagramme de séquence «S'authentifier»

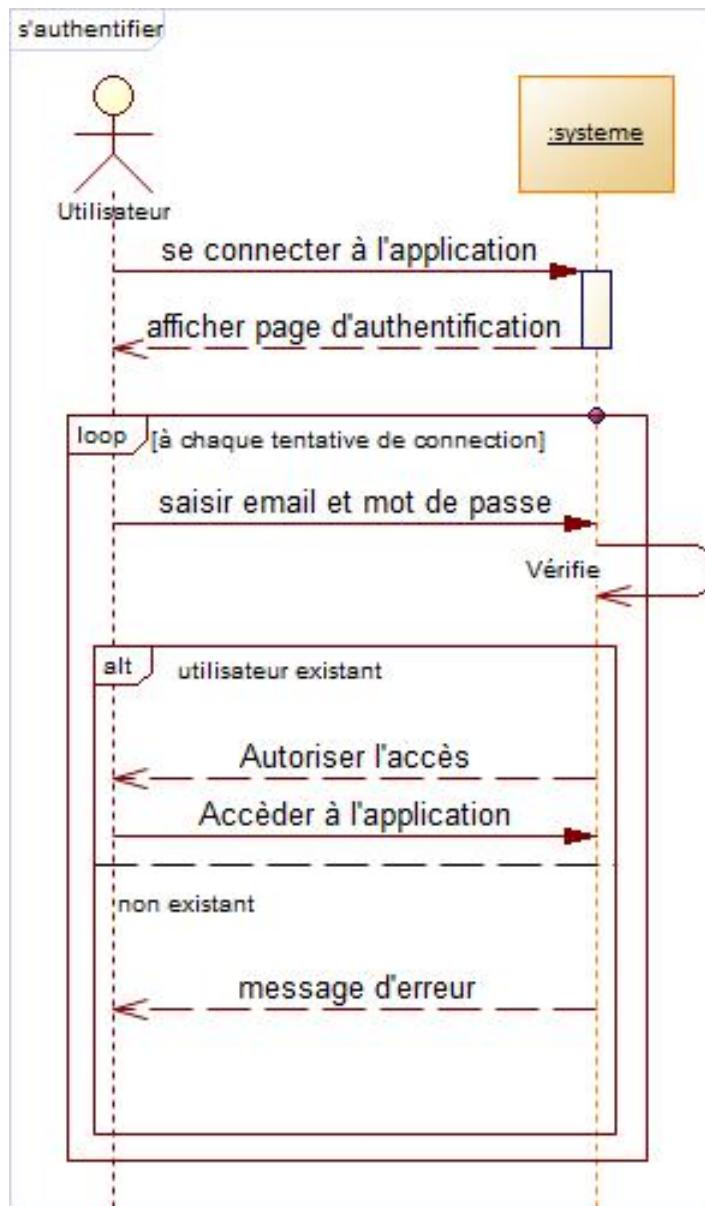


FIGURE 2.7 – Diagramme de séquence « S'authentifier »

#### Description textuelle du diagramme de séquence « S'authentifier »

Tout utilisateur de l'application doit s'authentifier afin d'accéder à son espace dédié. Donc si les droits d'accès entrés (mail et mot de passe) sont corrects l'utilisateur peut accéder à l'application. Sinon le système affiche un message d'erreur.

#### 2.2.2.4 Diagramme de séquence «Ajouter utilisateur»

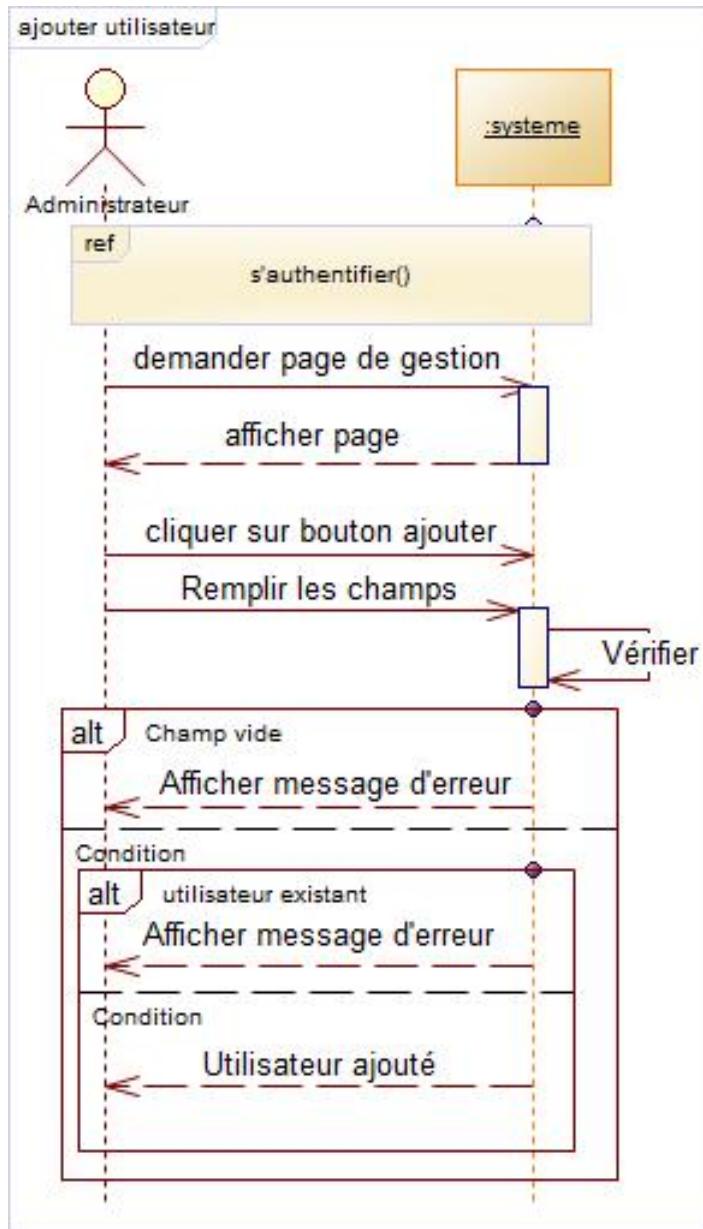


FIGURE 2.8 – Diagramme de séquence « Ajouter utilisateur»

#### Description textuelle du diagramme de séquence « Ajouter utilisateur »

L'administrateur de l'application est celui qui est responsable d'ajouter des utilisateurs(étudiants, enseignants). Pour se faire il doit s'authentifier en tant qu'administrateur tout d'abord, ensuite il demande la page de gestion d'utilisateurs (soit celle des étudiants soit celle des enseignants selon son besoin ) et le système lui affiche la page demandée. Ensuite il clique sur le bouton ajouter, il remplit tous les champs et valide l'ajout d'un nouvel utilisateur.

### 2.2.2.5 Diagramme de séquence «Modifier utilisateur»

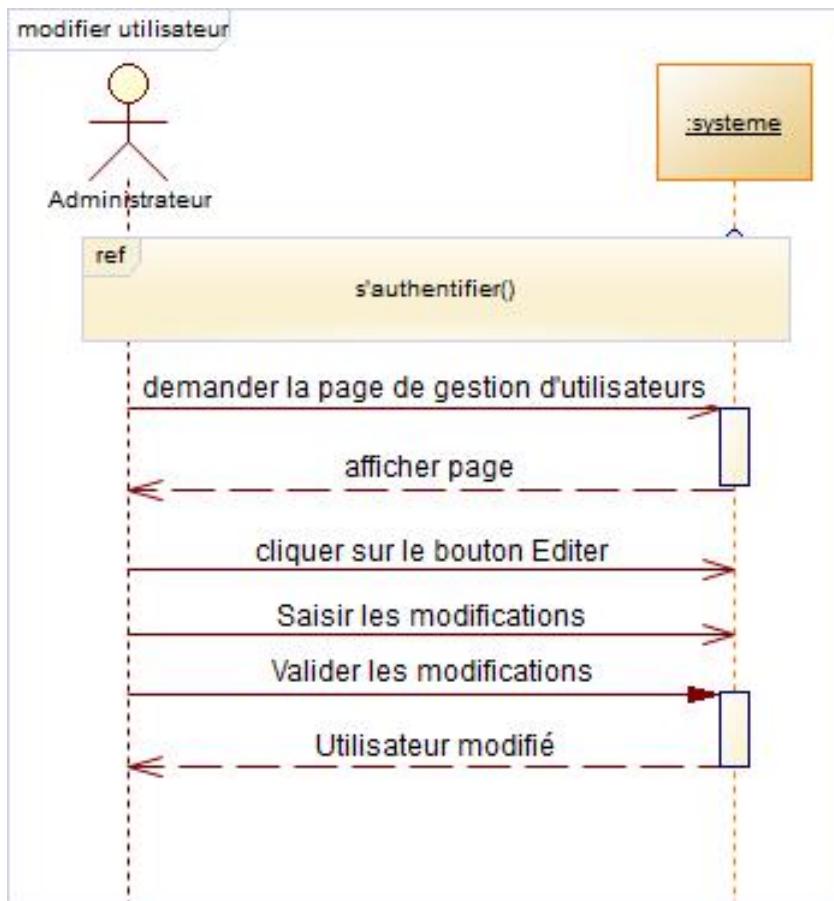


FIGURE 2.9 – Diagramme de séquence « Modifier utilisateur»

#### Description textuelle du diagramme de séquence « Modifier utilisateur »

Après avoir s'authentifier, l'administrateur demande la page de gestion d'utilisateurs (soit celle des étudiants soit celle des enseignants, selon son besoin) et le système lui affiche la page demandée. Ensuite il clique sur le bouton Editer et saisit les modifications qu'il souhaite apporter aux informations de l'utilisateur et les valide.

### 2.2.2.6 Diagramme de séquence « Créer une proposition de séance de rattrapage »

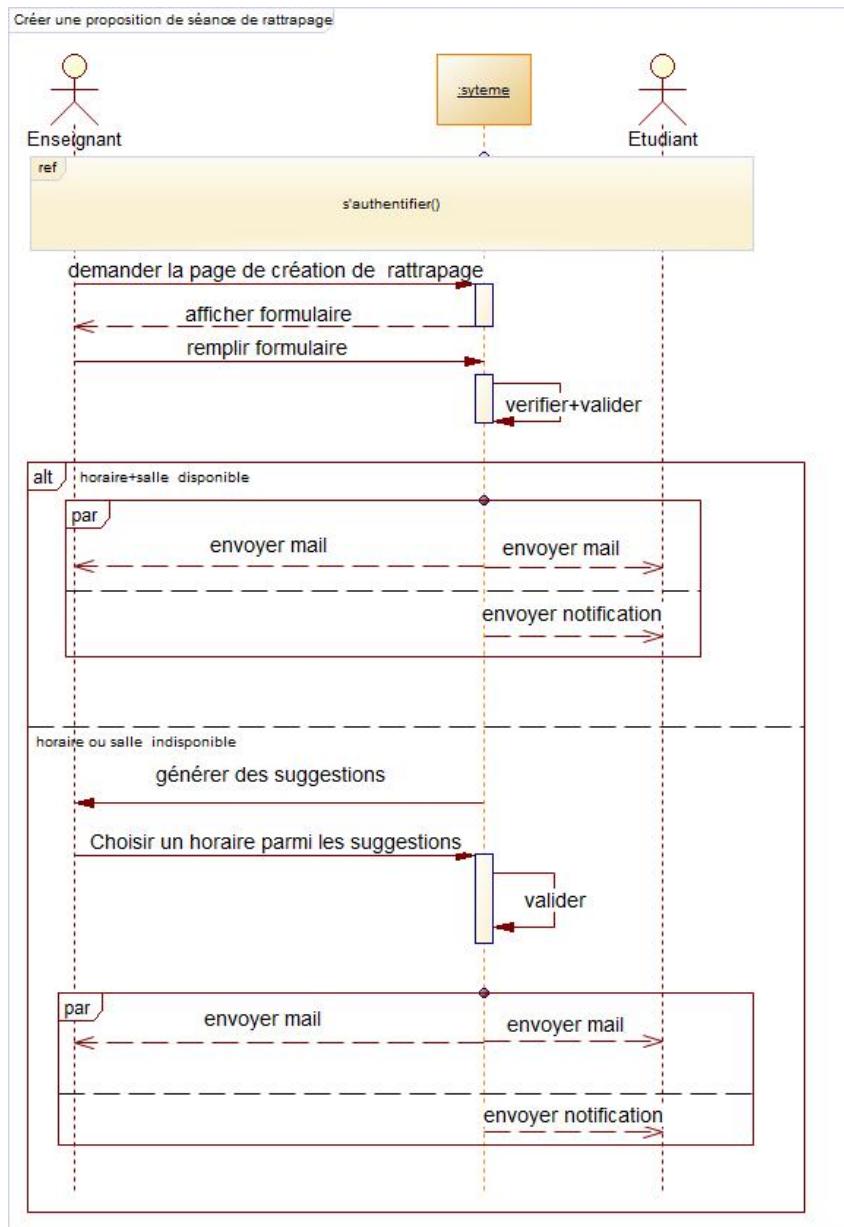


FIGURE 2.10 – Diagramme de séquence « Créer une proposition de séance de rattrapage »

#### Description

Après avoir s'authentifier l'enseignant demande la page de rattrapage et remplit le formulaire. Ensuite le système vérifie la disponibilité de l'horaire entré par l'enseignant et de la salle à cet horaire . Si l'horaire et la salle sont disponibles le système va envoyer une notification à tous les étudiants concernés et va envoyer un mail à l'enseignant et aussi à ses étudiants afin de leurs informer que le rattrapage va avoir lieu à une date donnée dans

une salle bien déterminée.

Dans le cas où l'horaire ou la salle soient indisponibles le système génère un ensemble de suggestions contenant l'horaire et la salle et l'enseignant va choisir la suggestion qui lui convient.

**Remarque :** Le même scénario se présente dans le cas où l'enseignant veut créer une proposition d'un devoir surveillé.

#### 2.2.2.7 Diagramme de séquence « Générer absence »

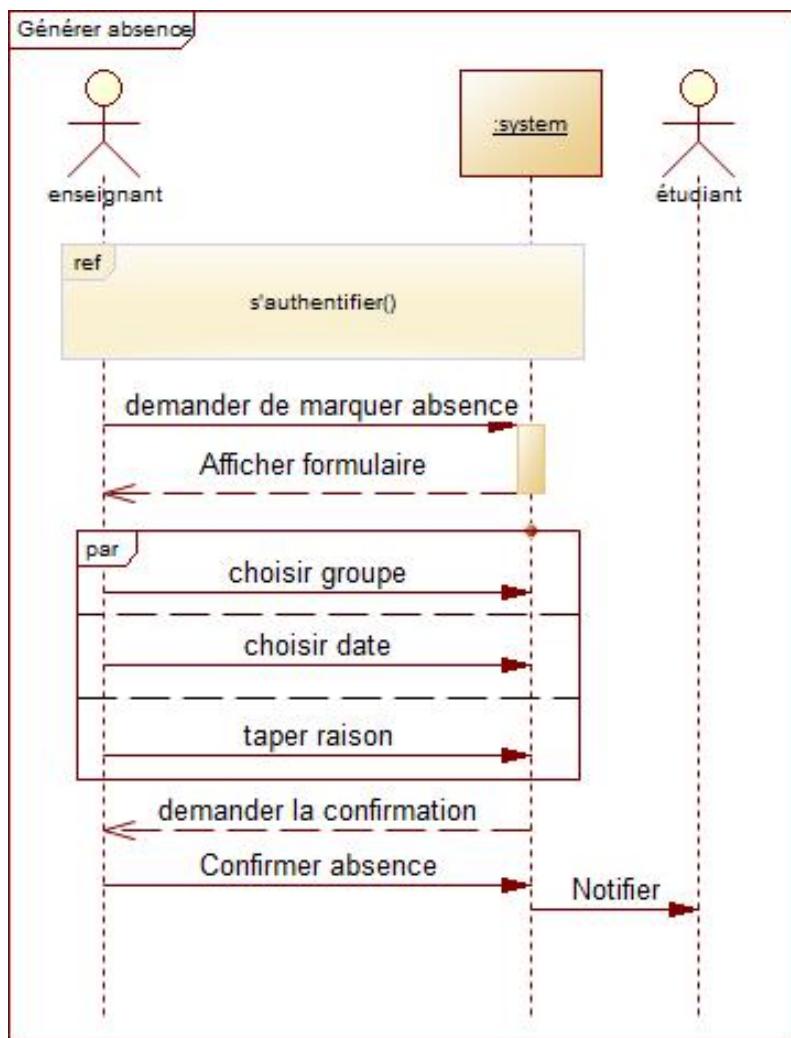


FIGURE 2.11 – Diagramme de séquence « Générer absence »

Ce diagramme décrit comment se fait la génération d'une absence par un enseignant. En effet, après avoir s'authentifier l'enseignant demande au système de générer une absence. Il choisit le groupe concerné par cette absence, la date et donne la raison, ensuite il confirme son absence et une notification serait envoyée aux étudiants appartenant au groupe spécifié afin de leurs informer de cette absence.

## 2.3 Conclusion

L'analyse et spécification des besoins donne une vision plus éclatante du sujet et une compréhension plus approfondie des tâches à accomplir. Elle aboutit également à prévoir les problèmes à rencontrer et analyser les solutions permettant de les contourner. Par conséquent, nous devons spécifier et chercher les moyens pour poser ces solutions afin de réaliser une meilleure conception.

C'est pour cette raison que nous avons essayé dans ce chapitre de présenter les besoins fonctionnels et non fonctionnels du projet. Ces besoins vont être le point de départ à partir duquel nous allons entamer la conception de cette application web. Cette conception est l'objet du prochain chapitre.

# Chapitre 3

## Conception

### Introduction

Après avoir analysé et modéliser les besoins, nous allons entamer la conception de notre présent projet.

Dans ce chapitre, on va exposer deux vues complémentaires de l'application :

- Une vue globale qui traduit son architecture globale .
- Une vue détaillée qui explique les différents choix pris.

### 3.1 Conception Architecturale

#### 3.1.1 Patron de conception

« Un patron de conception (Design Pattern) représente une solution (structure commune et répétitive de composants en interaction) à un problème récurrent dans un contexte donné. Ils décrivent les bonnes pratiques de conception capitalisées de l'expérience collective des informaticiens ».B[1]

Tout au long de la phase de conception, nous avons eu recours à des patrons de conception adéquates pour cette application.

Ces patrons sont :

- MVC (Model-View-Controller) :

C'est un patron de conception qui peut être utilisé dans la couche présentation vu que l'organisation d'une interface graphique est délicate.

Par conséquent, l'architecture MVC ne prétend pas en éliminer tous les problèmes, mais fournit une première approche pour le faire. L'idée est de bien séparer les données, les vues et les traitements. N[3]

La figure 3.1 représente ces 3 composants.

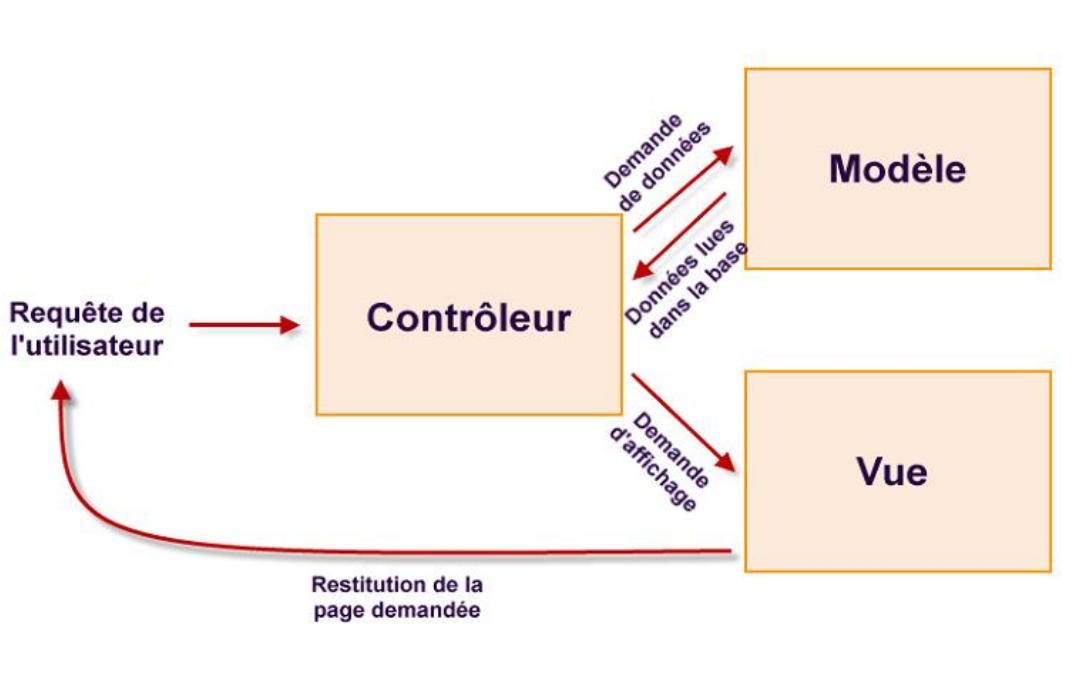


FIGURE 3.1 – Le modèle MVC N[4]

➤**Le modèle :**

Est responsable de la gestion des données manipulées dans le programme. En effet, à travers les requêtes SQL , le modèle récupère les données dans la base de données et les met en forme afin que ces dernières soient traitées ensuite par le Contrôleur. En plus il fournit des méthodes de mise à jour de ces données tel que la suppression, l'insertion et le changement de valeur.

➤**La vue :**

La vue concerne principalement la représentation des résultats des traitements effectués par le modèle à l'interface avec laquelle le client interagit grâce au browser "navigateur web", de telle sorte qu'elle n'effectue aucun traitement.

➤**Le contrôleur :**

Le contrôleur est chargé de la synchronisation du modèle et de la vue. Il reçoit tous les événements de l'utilisateur et enclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle et ensuite avertit la vue que les données ont changé pour que celle-ci se mette à jour. Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier.N[5]

### 3.1.2 Architecture générale

Nous présenterons par la suite l'architecture opérationnelle et l'architecture applicative de notre application web.

#### 3.1.2.1 Architecture opérationnelle

Dans notre projet nous avons choisi de travailler avec le style architecturale physique 3 tiers qui est défini comme suivant :

➤ **Tier client :**

C'est un service web (navigateur web) permettant au client d'accéder à tous les interfaces de l'application .

➤ **Tier serveur d'application :**

Comporte les différentes couches de notre application telle que les composants métiers et les méthodes d'accès à la base de données.

➤ **Tier persistance :**

correspond à la partie gérant l'accès aux données de l'application donc ce tier représente le serveur de base de données exploité pour la partie persistance des données dans cette application.

La figure 3.2 explique bien les relations entre les 3 tiers de cette architecture.

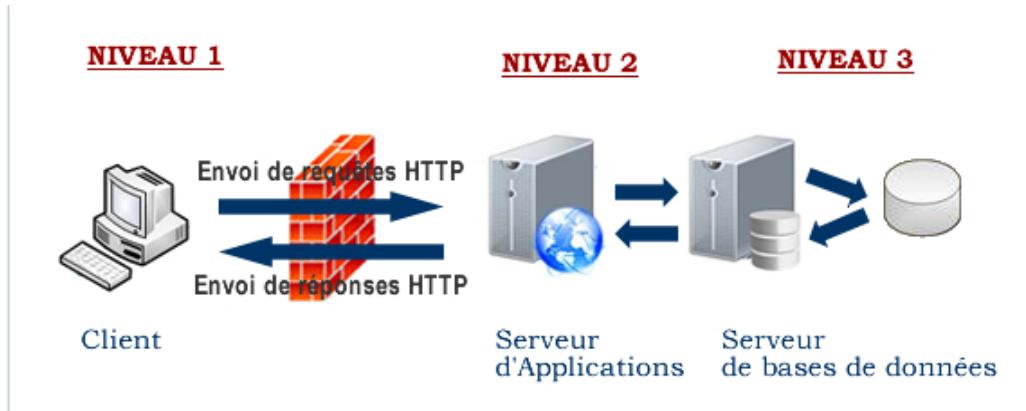


FIGURE 3.2 – Architecture 3 tiers de l'application N[6]

#### 3.1.2.2 Architecture applicative

Dans l'architecture opérationnelle nous nous sommes intéressés à différencier entre les différents niveaux physiques de l'application. Par contre, l'architecture applicative tient compte du découpage logique de l'application. Dans notre solution, dans le but de garantir l'évolutivité, la maintenabilité et l'exploitabilité, nous avons choisi de travailler avec une architecture applicative multicouche celle de 3 couches.

La figure 3.3 schématisse l'architecture qui sera exploitée dans ce projet :

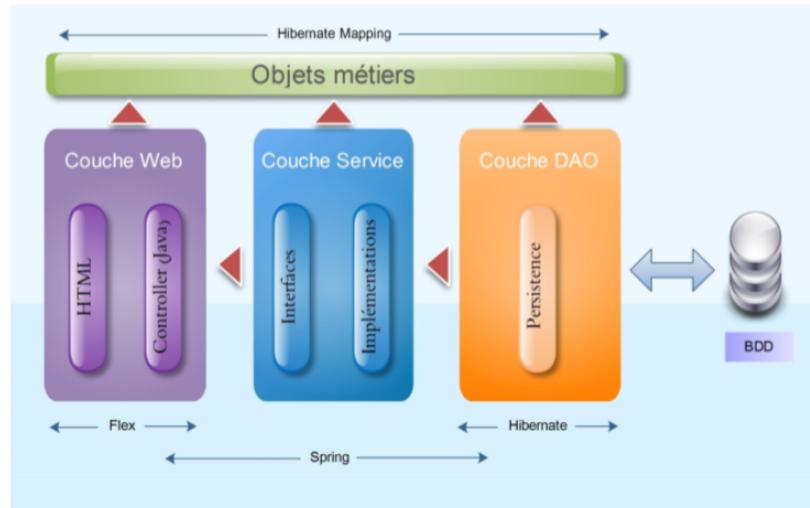


FIGURE 3.3 – architecture applicative de l’application.

Cette architecture vise principalement à séparer les préoccupations (données, métier et web) grâce à la séparation stricte des couches applicatives. En effet, les trois couches de cette architecture applicative sont :

- La couche DAO : permet l'accès à la base de données en utilisant l'ORM Hibernate.
- La couche Service : contient le code métier de l'application ,elle construit et harmonise les accès à la couche DAO .
- La couche Web : C'est la couche vue par le client.Elle appelle les traitements réalisés par la couche Service en fonction des actions exécutées par le client et récupère les données retournées. Elle met ensuite en forme ces données pour les afficher au client.

Ces trois couches sont strictement séparées les unes des autres de façon qu'elles soient indépendantes entre elles. Bien évidemment, chaque couche ne connaît que les interfaces de la couche inférieure. De cette manière, chaque couche annonce, à partir de ses interfaces ,l'ensemble des traitements qu'elle fournit aux couches supérieures. Le lien entre interface et implémentation, qui introduit un couplage technologique est géré par Spring.

### 3.1.2.3 Architecture physique

L'architecture physique d'un logiciel décrit l'ensemble des entités matérielles supportant l'application. Généralement nous parlons d'architecture physique quand nous travaillons sur une application appelée distribuée (ou répartie). En effet un système reparti est défini comme étant un ensemble de programmes distribués sur un réseau de communication et qui a pour but de coopérer et réaliser un service.

La figure suivante décrit les entités matérielles de notre application :

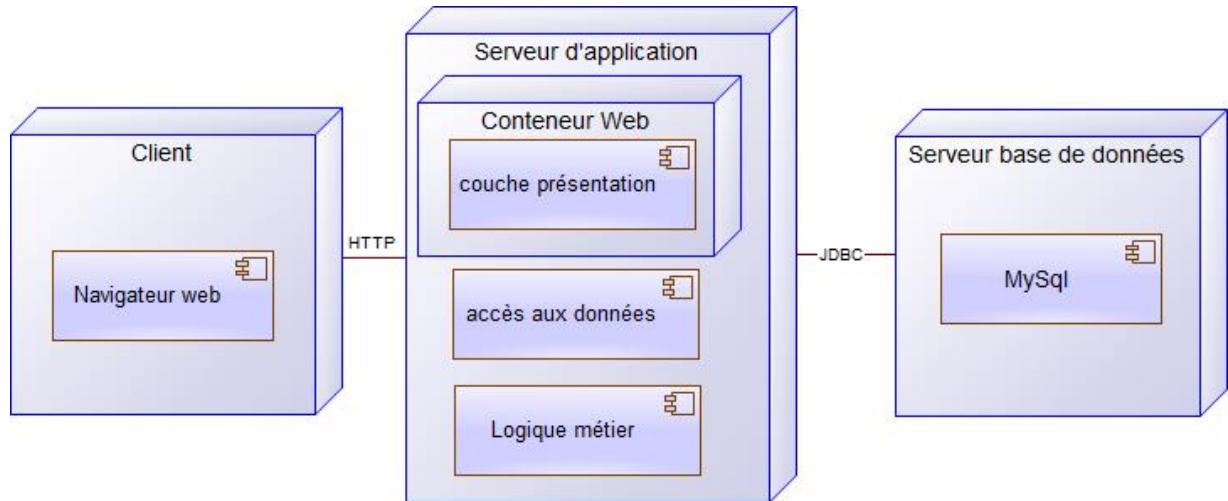


FIGURE 3.4 – Diagramme de déploiement

Le diagramme ci-dessus englobe l'ensemble des serveurs installés dans le but de garantir le fonctionnement de notre application, ainsi que les composants déployés sur les serveurs et les instances de la base de données. Le client a un Navigateur web, le serveur web intercepte les requêtes HTTP provenant du client qui seront exécutés par le serveur d'application. Ce serveur d'application consulte la couche d'accès aux données qui récupère les données à partir de la base de données.

## 3.2 Conception détaillée

### 3.2.1 Diagramme de package

C'est un diagramme qui donne une représentation graphique de haut niveau de la façon dont notre application est organisée et permet de déterminer les liens de généralisation et de dépendance entre les packages.

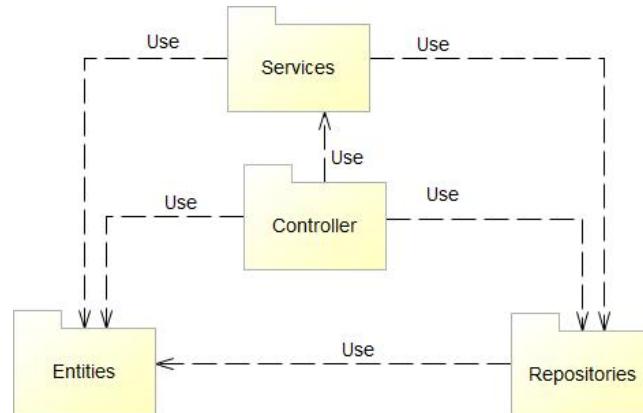


FIGURE 3.5 – Diagramme de paquetage

### 3.2.2 Conception du diagramme de classe

Le diagramme de classe est considéré comme étant le diagramme le plus important lors de la modélisation orientée objet. En effet son importance provient du fait qu'il modélise d'une manière claire la composition interne du système.

Moyennant ce diagramme nous parvenons à présenter d'une manière abstraite les différentes classes de la couche modèle. (Model Layer).

La figure 3.6 schématisé le diagramme de classe qui expose un essai pour modéliser notre système dans une approche centrée utilisateur.

En effet toutes les classes figurant dans ce diagramme servent à alimenter la base de données.

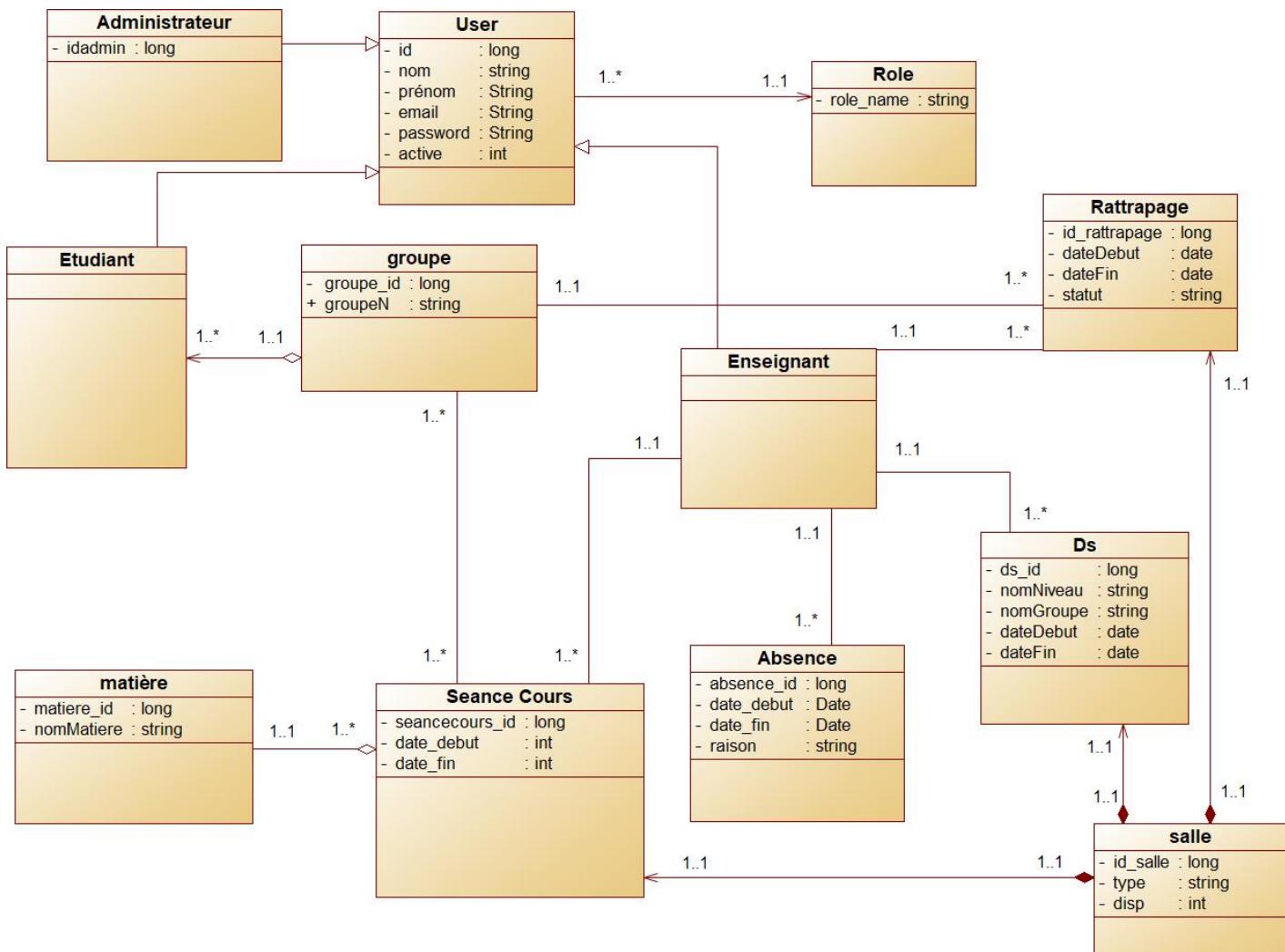


FIGURE 3.6 – Diagramme de classe

### Description du diagramme de classe

**\*Classe Utilisateur :** C'est une classe qui représente l'utilisateur de notre application qui peut être un administrateur, un enseignant ou étudiant. En effet chaque utilisateur a seulement accès à son espace dédié après avoir été authentifié.

**\*Classe Enseignant :** C'est une classe qui hérite de la classe mère User , elle représente l'acteur principal de notre application. En effet, chaque enseignant a la possibilité d'assurer des séances de cours,des rattrapages ou des devoirs surveillés, et de générer des absences .

**\*Classe Etudiant :** C'est une classe qui hérite de la classe mère utilisateur(User) représentant l'entité étudiant en relation avec le groupe concerné.

**\*Classe Administrateur :** C'est une classe qui hérite de la classe mère User représentant l'entité administrateur responsable de la gestion des utilisateurs.

**\*Classe Rôle :** Cette classe définit le rôle affecté à chaque utilisateur, où chaque utilisateur peut avoir un seul rôle et chaque rôle peut être affecté à plusieurs utilisateurs.

**\*Classe rattrapage :** Représente l'ensemble d'informations concernant un rattrapage (DateDebut, DateFin,statut...).Un rattrapage peut se réaliser dans une salle par un seul enseignant et pour un seul groupe .

**\*Classe séance cours :** Représente l'ensemble de détails concernant une séance de cours(dateDebut,dateFin,...). Elle concerne une seule matière à la fois.

**\*Classe absence :** Cette classe définit les informations d'une absence(DateDebut, DateFin, raison..) . En effet un enseignant peut marquer plusieurs absences à la fois.

**\*Classe Salle :** Cette classe représente les informations concernant une salle qui peut contenir soit une séance de cours soit un rattrapage soit un devoir surveillé.

**\*Classe matière :** Une matière est caractérisée par son nom et sa id unique. Elle est enseignée dans plusieurs séances de cours.

**\*Classe DS :** La classe DS indique le nom du groupe nomGroupe , les dates de début et de fin DateDebut,DateFin .Un devoir surveillé se fait dans une seule salle par un seul enseignant.

**\*Classe groupe :** Caractérise les différents groupes existants qui sont distingués par le groupe\_id et désignés par l'attribut groupeN.

### **3.2.3 Conception de la Base de Données**

Après avoir décrit les classes de notre application ,passons maintenant à la modélisation des différentes relations qui lient ces classes dans le schéma qui suit.

#### **3.2.2.1 Concept de modèle conceptuel de données**

Le modèle conceptuel de données "MCD"(ou MERISE) vise à représenter formellement les informations décrivant les objets et les associations qui les relient, et ceci en faisant une abstraction des contraintes d'implémentation.

#### **3.2.2.2 Modèle conceptuel de données**

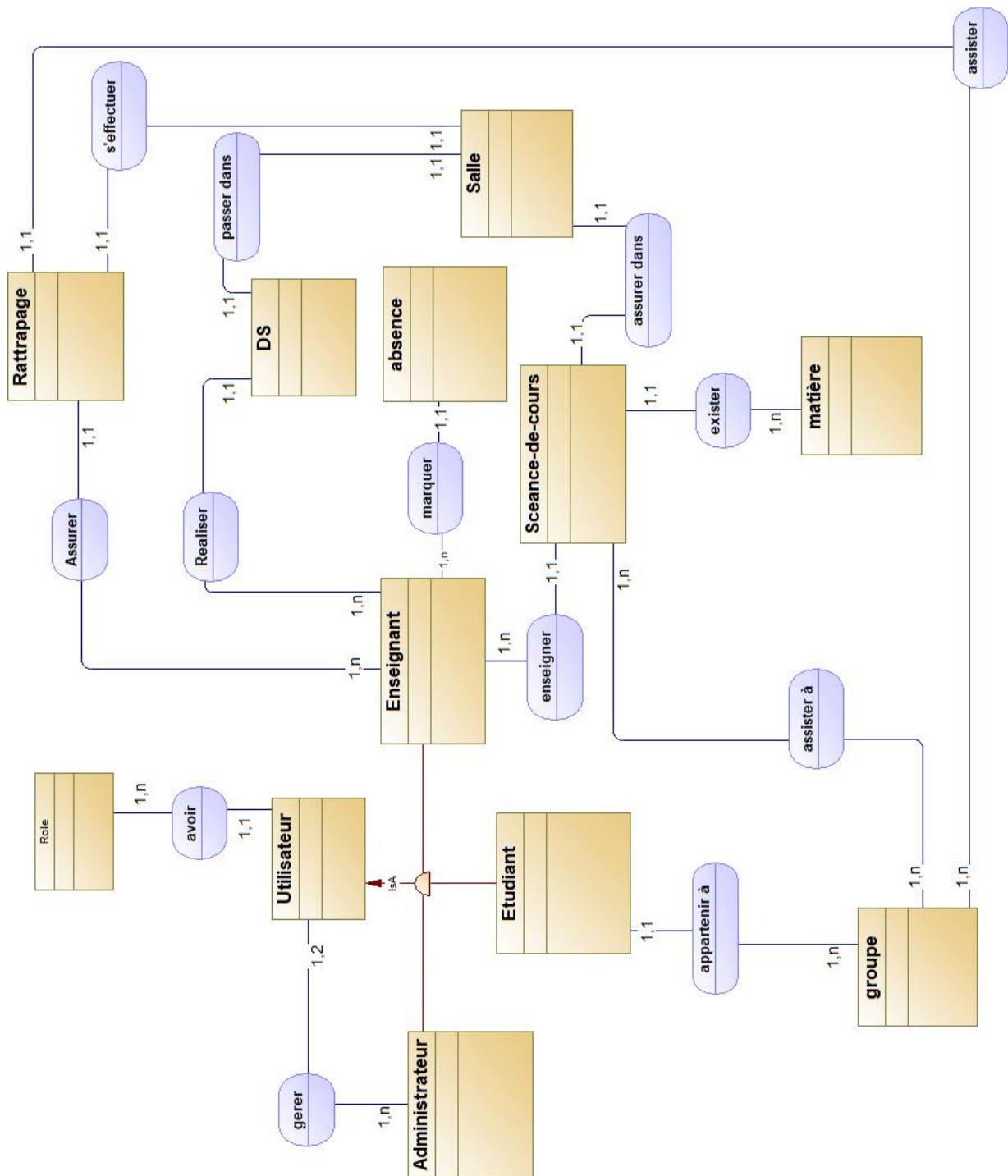


FIGURE 3.7 – Diagramme du modèle conceptuel

### 3.2.2.3 Schéma relationnel

Le modèle entité-association est transformé en un schéma relationnel normalisé (3 ème Forme Normale). Cette transformation aboutit à un ensemble de tables qui seront créées grâce à notre SGBD (système\_de\_gestion\_de\_base\_de\_données) et qui sont représentées dans la suite (Les clés primaires : soulignées et les clés étrangères : précédées par '#'). En effet, la base de données de notre application est composée de 12 tables :

- Table user(id,nom,prenom,email,username, password,active)
- Table admin(id,nom,prenom,email,username, password,active)
- Table enseignant(id,nom,prenom,email,username, password,active)
- Table etudiant(id,nom,prenom,email,username, password,# groupe\_id,active)
- Table role(role\_id,role).
- Table matiere(matiere\_id,nom\_matiere)
- Table salle(id\_salle,dispo,type)
- Table groupe(groupe\_id,groupeN,).
- Table absence(absence\_id,debut,Fin,raison,#id)
- Table seance\_cours(seancecours\_id,date\_debut,date\_fin,jour,#id,#matiere\_id,#salle\_id)
- Table rattrapage(id\_rattrapage,date\_debut,date\_fin,statut,#id,#groupe\_id)
- Table DS(id\_rattrapage,date\_debut,date\_fin,#id,#groupe\_id).

## 3.3 Diagrammes de séquences détaillés

Les diagrammes de séquences détaillés décrivent graphiquement le déroulement d'un use case .

### 1.Scénario d'authentification :

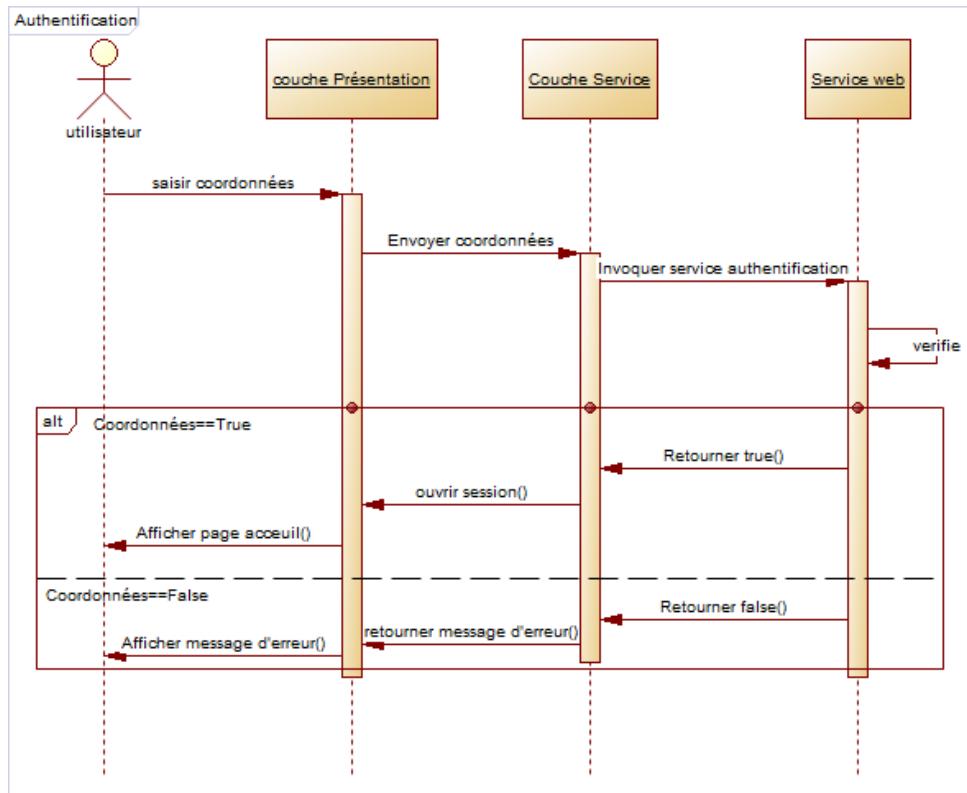


FIGURE 3.8 – Diagramme de séquence « Authentification »

### 3.4 Conclusion

Dans ce chapitre nous avons décrit le processus que nous avons suivi afin de bien concevoir notre application.

Nous avons décrit l'architecture logique de cette application dans le but de donner une vue globale concernant la décomposition de ce présent travail. Ensuite nous avons schématisé le diagramme de classe pour fournir une vue de notre système par le biais d'un ensemble de classes reliées entre elles. Nous avons en plus projeté dans ce chapitre le modèle entité association décrivant de façon formelle les données utilisées par le système d'information. Dans le chapitre suivant nous présenterons la partie réalisation au sein de laquelle nous décrirons l'environnement matériel et logiciel et nous présenterons l'implémentation des différentes parties conçues.

# Réalisation

## Introduction

Cette partie est consacrée à la présentation de tous les environnements logiciels que nous avons employé afin de réaliser notre projet, et à sa mise en œuvre avec les tests d'intégration ainsi que les perspectives.

### 4.1 Environnement de travail

Cette section est dédiée pour la présentation de l'environnement matériel et logiciel utilisés pour le développement de cette application.

#### 4.1.1 Environnement matériel

Afin de réaliser cette application nous avons à notre disposition deux ordinateurs caractérisés dans le tableau ci-dessous :

	Machine 1	Machine 2
Mémoire	8 Go	8 Go
Processeur	Intel Core i5	Intel Core i5
Système d'exploitation	Windows 10	Windows 10
Capacité	1 TO	1 TO

TABLE 4.1 – Environnement matériel

#### 4.1.2 Environnement logiciel

À propos l'environnement logiciel nous présentons par la suite la liste des différents composants déployés et la raison pour laquelle on les a choisi :

- ◆ Environnement de développement intégré : Eclipse IDE for Java EE Developers .

### ♦ Serveur de base de données : MySQL

C'est un système de gestion de bases de données relationnelles (SGBDR). C'est un logiciel libre, open source et fait partie des logiciels de gestion de base de données les plus utilisés au monde. Il supporte le langage de requête SQL.

### ♦ plateforme de développement Web : WampServer

Il s'agit de l'une des plus célèbres interfaces pour gérer une base de données MySQL. Cette interface permet d'exécuter, très facilement et sans avoir beaucoup de connaissances en bases de données, des requêtes comme les créations de table de données, les insertions, les mises à jour, suppressions et modifications de structure de la base de données.

### ♦ Outils de modélisation UML :

♦ PowerDesigner : C'est un logiciel de conception créé par la société SAP, qui permet de modéliser les traitements informatiques et leurs bases de données associées. Nous avons utilisé ce logiciel durant notre projet pour schématiser les diagrammes de séquence, le diagramme de classe ,le diagramme de déploiement et le modèle entité association.

♦ StarUml : c'est un logiciel de modélisation UML gratuit, open source, flexible et extensible. Grace à ce logiciel nous avons pu représenter les différents diagrammes use case.

### ♦ Editeur de maquettes IHM : Balsamiq Mockups

c'est un outil permettant de créer facilement des prototypes d'IHM électronique. Avec Balsamiq Mockups il est ainsi possible de prototyper tout type d'applications (desktop, web, smartphone, ...)

## 4.2 Choix des technologies

### 4.2.1 Choix du langage Java (Java 8)

Le Langage JAVA est à la fois un langage de programmation puissant élaboré dans le but d'être international et inter plate-forme, il est aussi un environnement de développement qui s'avère continuellement étendu prêt à fournir de nouvelles caractéristiques et bibliothèques offrant la possibilité de gérer de façon agile des problèmes habituellement complexes dans les langages de programmation classiques, tels que le multithreading, la programmation réseau, les accès aux bases de données, l'informatique repartie et la programmation web. En ce qui concerne le programmeur, Java lui permet de diminuer le temps qu'il passe dans le développement d'une application à l'aide de la réutilisation du code développé. Ces raisons nous ont poussés à choisir ce langage.

En effet , dans notre projet on a utilisé la version Java 8 et qui offre de nouvelles fonctionnalités, des performances accrues et des corrections de bugs afin d'améliorer l'efficacité de développement et d'exécution des programmes Java.

Voici un bref résumé des améliorations apportées à la version Java 8 :

-Expression lambda et méthodes d'extension virtuelle.

- API de date/heure : Cette nouvelle API permet aux développeurs de gérer la date et l'heure de façon plus naturelle, simple et compréhensible.
- Moteur JavaScript Nashorn : Une nouvelle implémentation légère et hautes performances du moteur JavaScript est intégrée au JDK et disponible pour les applications Java via les API existantes.
- Amélioration de la sécurité : Remplacement de la liste existante des méthodes sensibles à l'appelant tenue à jour manuellement par un mécanisme identifiant de façon précise ces méthodes et permettant le repérage des appelants de manière fiable.N[16]

### 4.2.2 Choix de la plateforme J2EE

Nous avons adopté la technologie J2EE pour les raisons suivantes :

- Sa maturité compte tenu aux autres solutions adoptées pour les applications web, étant donné qu'elle est présente sur le marché depuis plusieurs années.
- Elle profite d'outils de développement open source .
- Elle dispose de plusieurs frameworks minimisant la complexité de mise en œuvre des applications.

### 4.2.3 Choix du framework Spring

Spring est un framework libre conçu pour mettre en place l'infrastructure d'une application java, dont il facilite le développement et les tests.

Spring est un conteneur léger , autrement dit une infrastructure similaire à un serveur d'application J2EE.Donc ce framework prend en charge la création d'objets et la mise en relation de ces objets par l'intermédiaire d'un fichier de configuration qui décrit les objets à fabriquer et les relations de dépendances entre ces eux. Le majeur 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). C'est pour cette raison que Spring est qualifié de conteneur " léger ".

Spring s'articule essentiellement sur l'intégration de 3 concepts clés :

- Inversion de contrôle : assurée de deux façons différentes : la recherche de dépendances et l'injection de dépendances.
- Programmation orientée aspect.
- Couche d'abstraction.

### 4.2.4 Choix du micro framework Spring Boot

Dans ce projet nous avons choisi d'utiliser "Spring Boot" qui présente un des derniers projets du portefeuille Spring.

Spring Boot est un projet ou un micro-framework qui vise à faciliter la configuration d'un projet Spring et d'accélérer le temps alloué au démarrage d'un projet. Ce projet

possède plusieurs atouts qui étaient derrière notre décision de l'adopter dans la phase de développement. Les principaux atouts de Spring Boot sont :

- Un site web (<https://start.spring.io/>) qui permet de générer d'une manière rapide la structure du projet en y incluant toutes les dépendances Maven nécessaires à l'application en question.
- L'utilisation de « Starters » pour gérer les dépendances. Spring a regroupé les dépendances Maven de Spring dans des « méga dépendances» afin de faciliter la gestion de celles-ci.
- L'auto-configuration, qui applique une configuration par défaut au démarrage d'une application.
- La configuration par les fichiers XML qui était la cause de plusieurs problèmes pour les développeurs est remplacée par un mode de configuration plus simple par des classes java
- Le déploiement applicatif est simplifié en offrant la possibilité d'intégrer directement un serveur Tomcat dans l'exécutable afin de faire tourner l'application au moment de démarrage
- La génération d'un fichier .war qui doit être déployé sur un serveur comme un Apache Tomcat.
- La mise à disposition des opérationnels, des métriques pour pouvoir suivre l'application déployée en mode production. C'est pourquoi Spring Boot utilise « Actuator » qui est un système permettant de montrer une application via des URLs spécifiques ou des commandes disponibles via SSH.N[7]

### 4.2.5 Choix de la base de donnée : MySQL

MySQL représente un serveur de bases de données relationnelles qui est Open Source et qui stocke les données dans des tables séparées au lieu 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.N[8]

Nous avons choisi de travailler avec MySQL pour les raisons suivantes : -Rapidité : Les développeurs de MySQL affirment que MySQL est le système de base de données le plus rapide.

-Facilité d'utilisation : MySQL a des hautes performances, mais simple à utiliser, beaucoup moins complexe à configurer et à administrer que les grands systèmes.

-Prise en charge du langage de requête : MySQL comprend le langage SQL (Structured Query Language), le langage standard de choix pour tous les systèmes de bases de données modernes.

#### 4.2.6 Choix du Font : BOOTSTRAP

Bootstrap met à notre disposition un ensemble d'outils avantageux facilitant la création des applications web.

Il contient des codes HTML5 et CSS, des boutons, des formulaires, plusieurs outils de navigation et autres éléments interactifs, ainsi que des extensions JavaScript en option.

Bootstrap est compatible avec les dernières versions des navigateurs majeurs, mais peut fonctionner de manière affaiblie sur des navigateurs anciens. Il accélère le développement web front end et le rend plus simple, il est open source, très populaire, et en évolution continue et rapide surtout avec l'avènement fréquent de nouvelles versions.

### 4.3 Présentation des interfaces

Les IHM (Interface Homme/Machine) représentent un facteur primordial dans la réussite d'un système d'information et surtout dans le cas des applications Web.

Ces interfaces doivent respecter les heuristiques d'utilité et d'utilisabilité et obéir aux règles d'ergonomie afin que l'utilisateur s'adapte rapidement à la logique de l'application.

Dans ce qui suit on va présenter les interfaces les plus importantes de notre application.

-Page d'authentification :

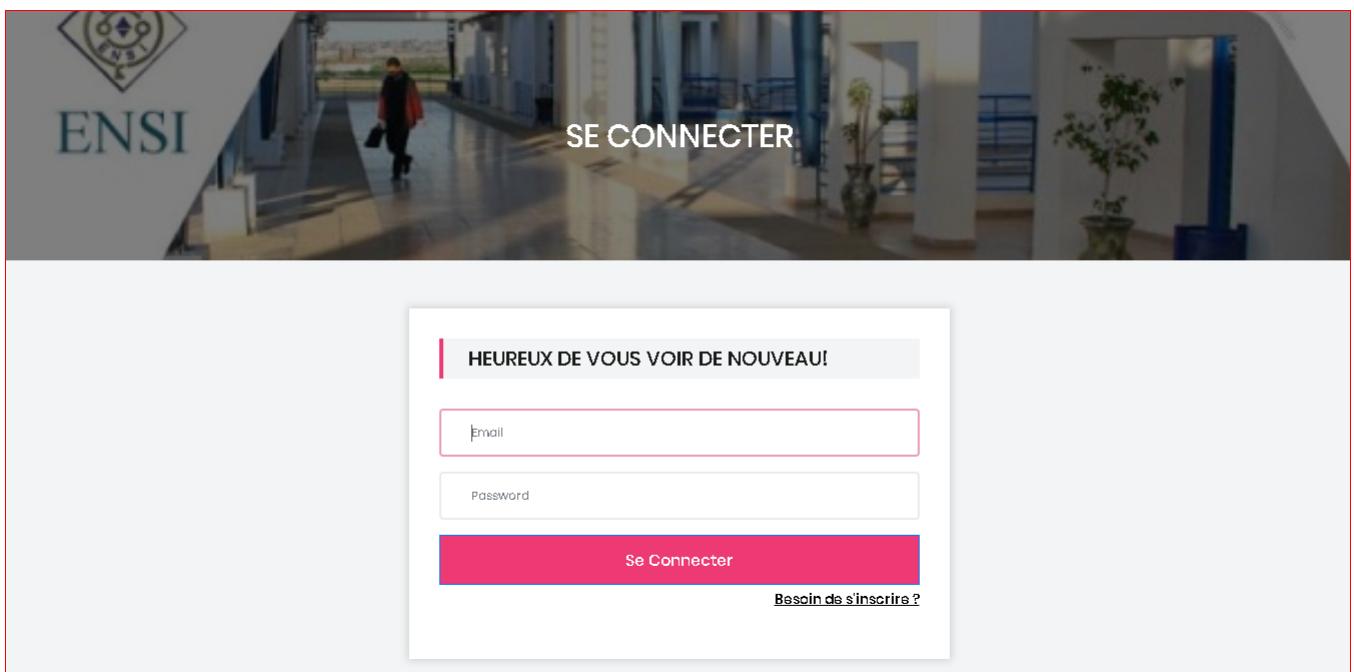


FIGURE 4.1 – Interface d'authentification

## CHAPITRE 4. RÉALISATION

-Page d'accueil du site :

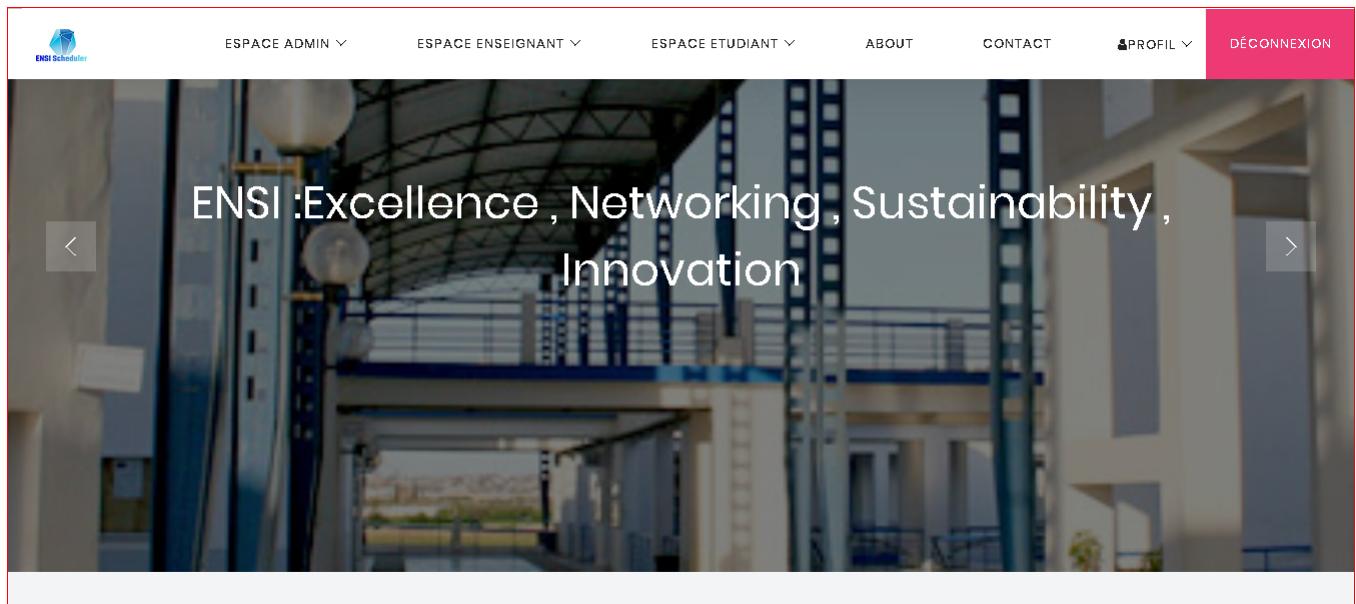


FIGURE 4.2 – Interface d'accueil du site

A screenshot of the ENSI website's 'Présentation de l'ENSI' (Presentation of ENSI) page. The page is divided into three main sections: 'DIRECTRICE' (with Narjes Bellamine Ben Saoud), 'PRÉSENTATION DE L'ENSI' (with a detailed text about the history and achievements of the university), and 'ACTUALITÉS' (with a thumbnail of a university building and a link to the 2018/2019 annual report). The 'Présentation de l'ENSI' section also includes a timeline of milestones from 1984 to 2014.

FIGURE 4.3 – suite Interface d'accueil du site

## CHAPITRE 4. RÉALISATION

---

-Page d'inscription des étudiant :

The screenshot shows a registration form titled "HEUREUX DE VOUS VOIR REJOINDRE !". It contains four input fields for "Prenom", "Nom", "Email", and "Mot De Passe". Below these is a section titled "choisir Groupe:" with a list of radio buttons for group selection. The groups listed are: II-1-A, II-1-B, II-1-C, II-1-D, II-1-E, II-1-F, II-2-A, II-2-B, II-2-C, II-2-D, II-2-E, II-2-F, II-3-ILSI, II-3-ISID, II-3-IF, II-3-RSR, II-3-SLE, and II-3-II. At the bottom left is a link "Déjà un membre?", and at the bottom right is a link "Vous êtes enseignant?". A large red "S'inscrire" button is located at the bottom center.

FIGURE 4.4 – Interface d'inscription des étudiants

-Page d'inscription des enseignants :

The screenshot shows a registration form titled "HEUREUX DE VOUS VOIR REJOINDRE!". It contains four input fields for "Prenom", "Nom", "Email", and "Mot De Passe". Below these is a section titled "Choisir Groupe:" with a list of radio buttons for group selection. The groups listed are: II-1-A, II-1-B, II-1-C, II-1-D, II-1-E, II-1-F, II-2-A, II-2-B, II-2-C, II-2-D, II-2-E, II-2-F, II-3-ILSI, II-3-ISID, II-3-IF, II-3-RSR, II-3-SLE, and II-3-II. At the bottom left is a link "Déjà un membre?", and at the bottom right is a link "Vous êtes Etudiant?". A large red "S'inscrire" button is located at the bottom center.

FIGURE 4.5 – Interface d'inscription des enseignants

## CHAPITRE 4. RÉALISATION

---

- page de création de demande d'absence pour l'enseignant :

The screenshot shows a web interface titled 'CREEZ VOTRE DEMANDE D'ABSENCE!' (Create Your Absence Request). At the top left, there is a breadcrumb navigation: 'Espace Enseignant > Marquer Absence'. The form consists of three input fields: 'Raison-Description' (Reason-Description), 'DateDebut dd/MM/yyyy' (Start Date dd/MM/yyyy), and 'DateFin: dd/MM/yyyy' (End Date dd/MM/yyyy). Below these fields is a large red 'Envoyer' (Send) button.

FIGURE 4.6 – Interface de création de demande d'absence

-page de création de demande de rattrapage pour l'enseignant :

The screenshot shows a web interface titled 'CREEZ VOTRE DEMANDE DE RATTRAPAGE !' (Create Your Compensation Request). At the top left, there is a breadcrumb navigation: 'Espace Enseignant > Marquer Absence'. The form consists of two input fields: 'DateDebut: aaaa-mm-jjTh:mm:ss' (Start Date: aaaa-mm-jjTh:mm:ss) and 'DateFin: aaaa-mm-jjTh:mm:ss' (End Date: aaaa-mm-jjTh:mm:ss). Below these fields is a dropdown menu showing 'II-1-A' with a downward arrow. Below the dropdown is a large red 'Envoyer' (Send) button.

FIGURE 4.7 – Interface de création de demande de rattrapage

## CHAPITRE 4. RÉALISATION

---

-Profil de l'enseignant :

The screenshot shows a web-based profile editing interface. At the top, a header bar contains the title "Editer Profil". Below this, a section titled "Informations personnelles" (Personal Information) contains four input fields: "Nom" (Name), "Prenom" (First Name), "Email", and "Mot de passe" (Password). Each field has a placeholder text inside it. At the bottom of the form is a blue "Enregistrer" (Save) button.

FIGURE 4.8 – Interface de l'enseignant pour éditer son profil

-Page de gestion des enseignants par l'administrateur :

The screenshot shows a management interface for teachers. At the top, a header bar features the word "GESTION". Below this, a breadcrumb navigation shows "Accueil > Gestion Enseignants". The main area is titled "GESTION ENSEIGNANTS" and contains a sub-section "Gérer Enseignants". A table displays two teacher records with columns for "id", "email", "Nom", "prenom", "password", and actions. The first record has an id of 17, email fliss, Nom imtiaz, prenom imtiaz.fliss@ensi-uma, and password (redacted). The second record has an id of 13, email jemni, Nom leila, prenom leila, and password (redacted). A blue "+ Ajouter" (Add) button is located at the top right of the table area.

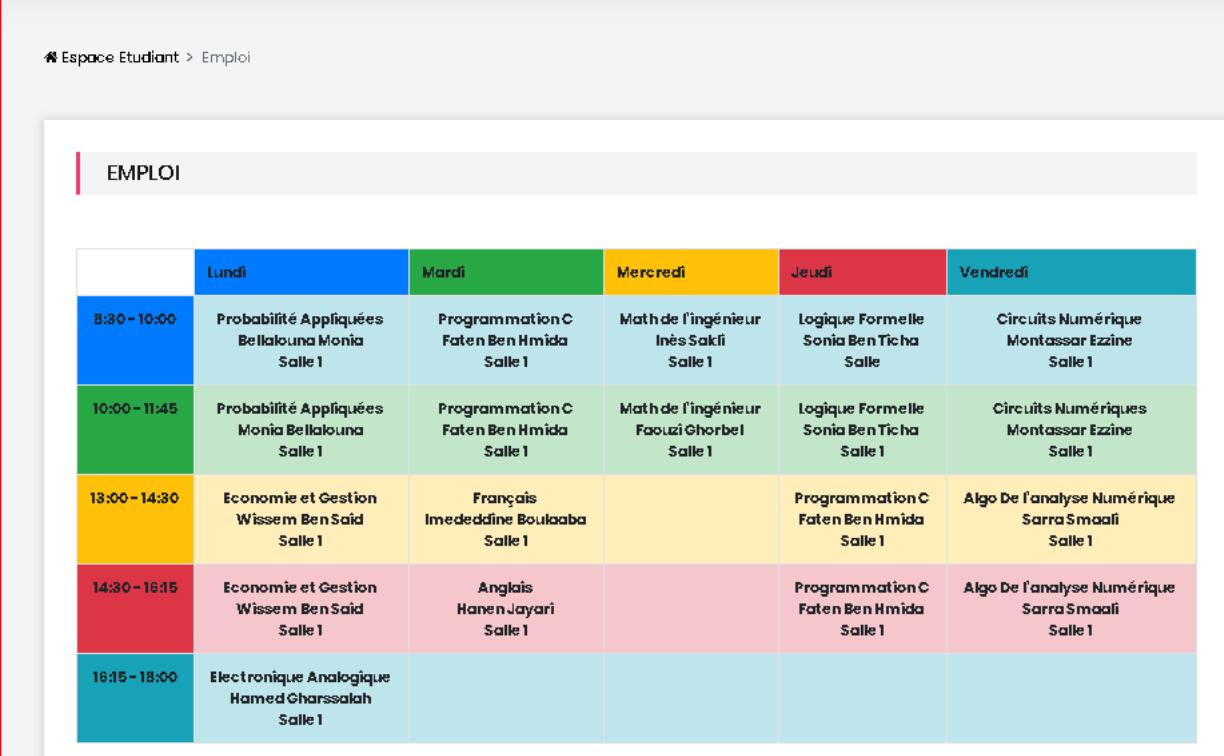
id	email	Nom	prenom	password	
17	fliss	imtiaz	imtiaz.fliss@ensi-uma	*****	
13	jemni	leila	leila.jemni.benayed@i	*****	

FIGURE 4.9 – Interface de gestion des enseignants par l'administrateur

## CHAPITRE 4. RÉALISATION

---

- Emploi du temps de l'étudiant :

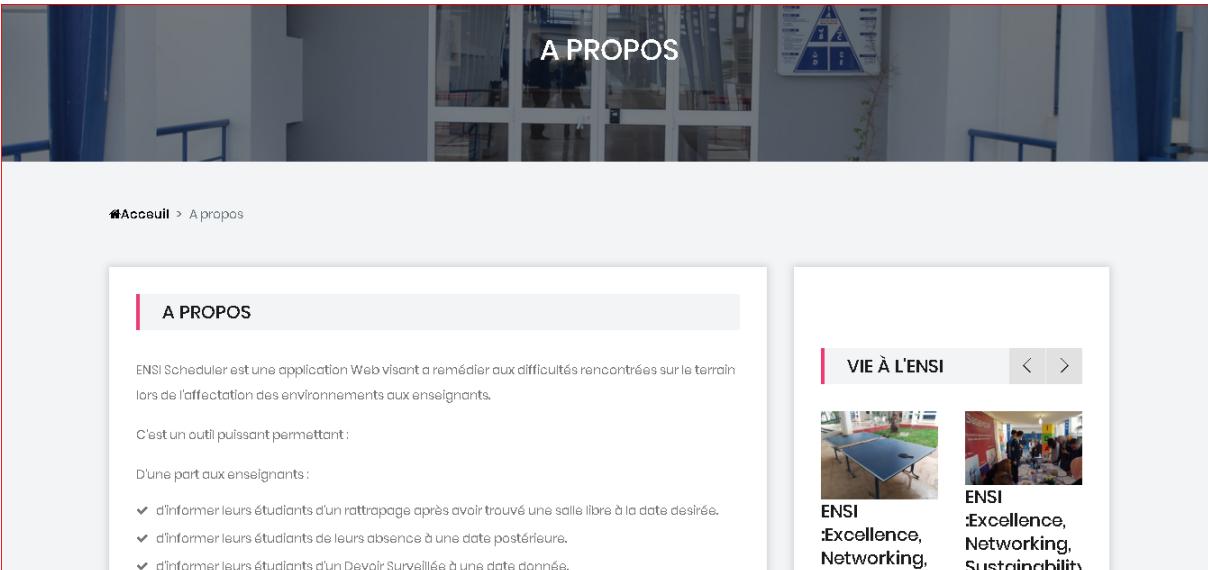


The screenshot shows a weekly schedule for a student. The days of the week are listed at the top: Lundi, Mardi, Mercredi, Jeudi, Vendredi. The schedule is organized by time slots: 8:30 - 10:00, 10:00 - 11:45, 13:00 - 14:30, 14:30 - 16:15, and 16:15 - 18:00. Each slot contains a subject name, teacher's name, and room number.

	Lundi	Mardi	Mercredi	Jeudi	Vendredi
8:30 - 10:00	Probabilité Appliquées Bellalouna Monia Salle 1	Programmation C Faten Ben Hmida Salle 1	Math de l'ingénieur Inès Sakli Salle 1	Logique Formelle Sonia Ben Ticha Salle	Circuits Numérique Montassar Ezzine Salle 1
10:00 - 11:45	Probabilité Appliquées Monia Bellalouna Salle 1	Programmation C Faten Ben Hmida Salle 1	Math de l'ingénieur Faouzi Ghorbel Salle 1	Logique Formelle Sonia Ben Ticha Salle 1	Circuits Numériques Montassar Ezzine Salle 1
13:00 - 14:30	Economie et Gestion Wissem Ben Said Salle 1	Français Imededdine Boulaaba Salle 1		Programmation C Faten Ben Hmida Salle 1	Algo De l'analyse Numérique Sarra Smaali Salle 1
14:30 - 16:15	Economie et Gestion Wissem Ben Said Salle 1	Anglais Hanen Jayari Salle 1		Programmation C Faten Ben Hmida Salle 1	Algo De l'analyse Numérique Sarra Smaali Salle 1
16:15 - 18:00	Electronique Analogique Hamed Gharssalah Salle 1				

FIGURE 4.10 – Interface d'emploi du temps d'un étudiant

-page à propos :



The screenshot shows the 'About' page of the ENSI Scheduler application. It features a header 'A PROPOS' and a main content area with text and images.

**A PROPOS**

ENSI Scheduler est une application Web visant à remédier aux difficultés rencontrées sur le terrain lors de l'affectation des environnements aux enseignants.

C'est un outil puissant permettant :

D'une part aux enseignants :

- ✓ d'informer leurs étudiants d'un rattrapage après avoir trouvé une salle libre à la date désirée.
- ✓ d'informer leurs étudiants de leurs absences à une date postérieure.
- ✓ d'informer leurs étudiants d'un Devoir Surveillance à une date donnée.

**VIE À L'ENSI**

**ENSI**  
:Excellence,  
Networking,  
Sustainability

FIGURE 4.11 – Interface à propos

## CHAPITRE 4. RÉALISATION

---

-page contact :

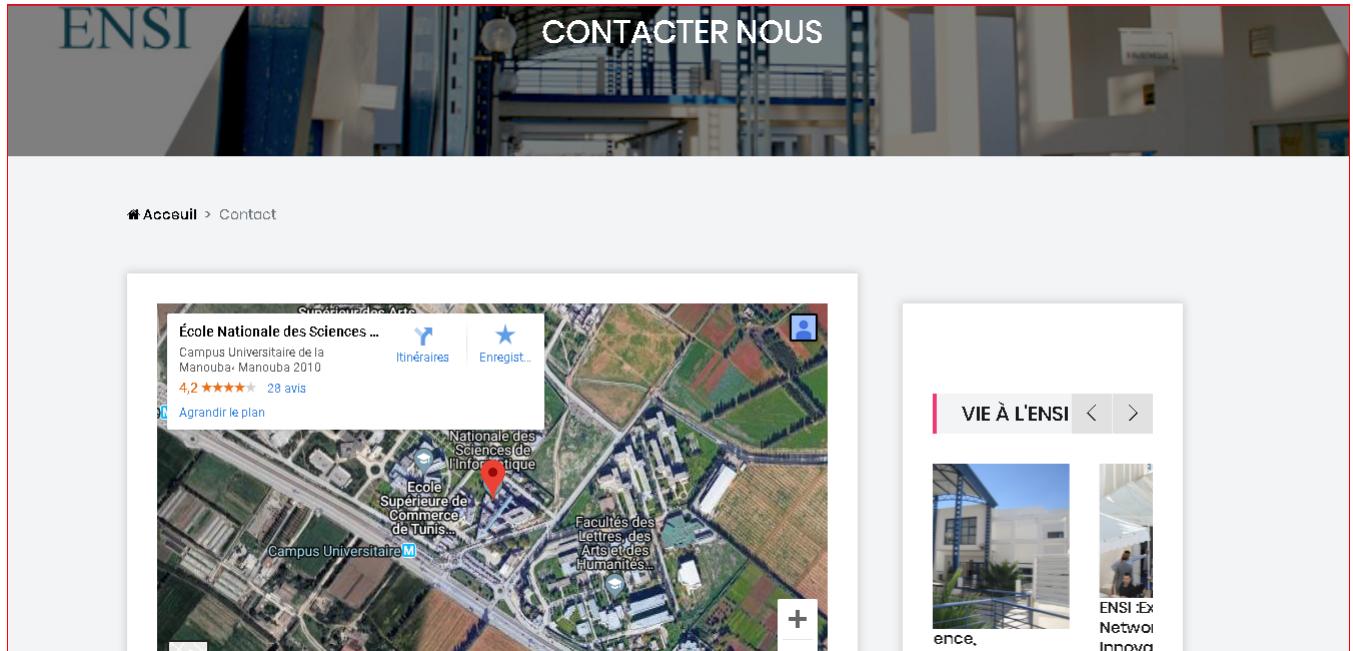


FIGURE 4.12 – Interface Contact

## 4.4 Conclusion

Au cours de ce chapitre , nous avons cité l'ensemble des environnements matériels et logiciels que nous avons déployé ainsi que les technologies utilisées. Ensuite nous avons présenté les différentes interfaces de notre application web avec les explications nécessaires.

# Conclusion et perspectives

Le présent manuscrit expose les différents étapes par lesquelles nous nous sommes astreints afin de mettre en place un prototype d'une application Web, permettant de gérer et d'affecter les environnements aux enseignants de l'ENSI. Pour atteindre ce résultat, nous avons commencé par placer le projet dans son contexte général, suivi d'une étude et d'une critique de l'existant sur le plan pratique. Ensuite, nous avons présenté les différents besoins et exigences relevées. Nous avons par la suite entamé la phase de conception, en allant de l'architecture adoptée, jusqu'à aboutir à une conception détaillée qui s'intéresse aux aspects statiques et dynamiques de notre système. Nous avons clôturé par la phase de réalisation, au cours de laquelle nous avons interprété notre modélisation conceptuelle en un prototype fonctionnel.

Malgré les contraintes temporelles nous sommes parvenus à achever une grande partie de notre projet et arriver à satisfaire la plupart des besoins escomptés, qui sont principalement en backoffice.

Ce présent travail nous a été très enrichissant du point de vue des connaissances obtenues. Il nous a apporté une opportunité pour s'initier à la technologie J2EE dont l'apprentissage voire la maîtrise nous sera indispensable dans la vie professionnelle.

À part les nouvelles connaissances acquises, ce projet nous a suscité de renforcer nos connaissances théoriques au niveau de la programmation Java et la gestion des bases de données relationnelles. Outre le côté technique, ce travail fût une occasion pour apprendre le travail en groupe, la gestion du temps et savoir la répartition des efforts .