Abstract

Ce projet a pour objectif de développer une solution automatisée pour la gestion des surveillances d'examens au sein d'un établissement d'enseignement. L'application lit les données à partir d'un fichier Excel, calcule les charges de travail des enseignants, détermine les séances à surveiller et effectue les répartitions nécessaires. Elle vise à réduire les erreurs humaines, améliorer l'équité dans la répartition des tâches, et optimiser l'utilisation des ressources (enseignants et salles). La solution utilise Python et la bibliothèque Pandas pour le traitement des données. Les résultats sont exportés automatiquement sous forme de fichiers Excel prêts à l'usage. Ce projet s'inscrit dans une démarche d'amélioration continue de la gestion administrative. Il constitue également une opportunité d'appliquer des compétences en programmation, analyse de données et modélisation.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

La gestion des emplois du temps et des charges de travail constitue un pilier fondamental pour le bon fonctionnement des établissements d'enseignement supérieur. Une organisation efficace des calendriers, des heures de cours, des surveillances et des examens permet non seulement d'optimiser les ressources humaines, mais également de garantir une meilleure qualité d'apprentissage pour les étudiants. En effet, une planification claire et accessible facilite la coordination entre les différents acteurs éducatifs, renforce la transparence et améliore la productivité de l'ensemble du système pédagogique.

Traditionnellement, cette gestion se faisait de manière manuelle, souvent à l'aide de documents papier ou de feuilles Excel non centralisées, ce qui engendrait des risques d'erreurs, des pertes d'informations et un manque de réactivité face aux changements imprévus. Aujourd'hui, grâce à l'évolution des technologies de l'information, il est possible de mettre en place des solutions numériques qui automatisent ces tâches, réduisent la charge administrative et offrent un accès personnalisé aux utilisateurs.

C'est dans ce cadre que s'inscrit le présent mini-projet, réalisé dans le cadre de ma formation à l'université. Ce projet consiste en l'étude, la conception et le développement d'une application web dédiée à la gestion des calendriers de surveillance. L'application vise à répondre aux besoins des étudiants, des enseignants et de l'administration en centralisant plusieurs fonctionnalités : gestion des comptes utilisateurs, consultation des emplois du temps, visualisation des supports de cours et des actualités, automatisation de la répartition des heures de surveillance en fonction de la charge hebdomadaire d'enseignement, ainsi que la gestion du calendrier des examens à partir d'un fichier Excel ou via une saisie manuelle.

Le présent rapport est structuré en quatre chapitres comme suit :

Le premier chapitre, intitulé "Cadre général du projet", introduit l'environnement du projet, décrit l'organisme de formation, l'analyse de l'existant ainsi que la solution proposée. Il détaille également la planification adoptée pour mener à bien ce projet.

Le deuxième chapitre, "Analyse et spécification des besoins", expose les exigences fonctionnelles et non fonctionnelles de l'application, ainsi que les différents utilisateurs ciblés et leurs interactions avec le système.

Le troisième chapitre, "Étude conceptuelle", présente la modélisation du système à l'aide des diagrammes UML et définit l'architecture générale de l'application.

Enfin, le quatrième chapitre, "Réalisation", décrit l'environnement de développement, les outils utilisés, ainsi que les interfaces graphiques développées, accompagnées de scénarios d'utilisation. Ce rapport se conclut par une synthèse des résultats obtenus et des

perspectives d'amélioration possibles.

Chapitre 1

Cadre général du projet

Introduction

Dans ce chapitre, nous décrivons le cadre général du projet. Tout d'abord nous commençons par une présentation de l'organisme d'accueil au sein duquel nous avons réalisé ce projet. Ensuite, nous détaillons l'étude et la critique de l'existant. Puis, dans la troisième partie nous présentons une étude des solutions existantes en discutant leurs avantages et leurs inconvénients. Dans la quatrième partie, nous décrierons la solution à réaliser. Enfin, nous terminerons ce chapitre par présenter la méthodologie de développement à adopter au cours de la réalisation de ce projet.

1.1 Cadre général du projet

Ce travail s'inscrit dans le cadre d'un projet pour la matière "Mini-projet" en deuxième année ingénierie au sein de l'Institut Supérieur des Sciences Appliquées et de Technologie de Sousse.

Ce projet s'est déroulé sur un semestre et sera présenté puis remis à l'Institut.

1.2 Organisme d'accueil : "ISSAT"

1.2.1 Présentation

Figure 1.1 : ISSAT sousse logo [1]

Un établissement scientifique relevant de l'Université de Sousse (Tunisie). Il est créé en application des dispositions du décret n°1385-2001 du 7 juin 2001.

L'Institut accueille ses activités dans les locaux dédiés à l'Institut Préparatoire aux Études d'Ingénieur. Dans ce cadre, les ressources de la bibliothèque ainsi que le matériel scientifique acquis au cours des trois années de la vie de l'Institut Préparatoire sont exploités.

1.2.2 Organigramme de ISSAT sousse

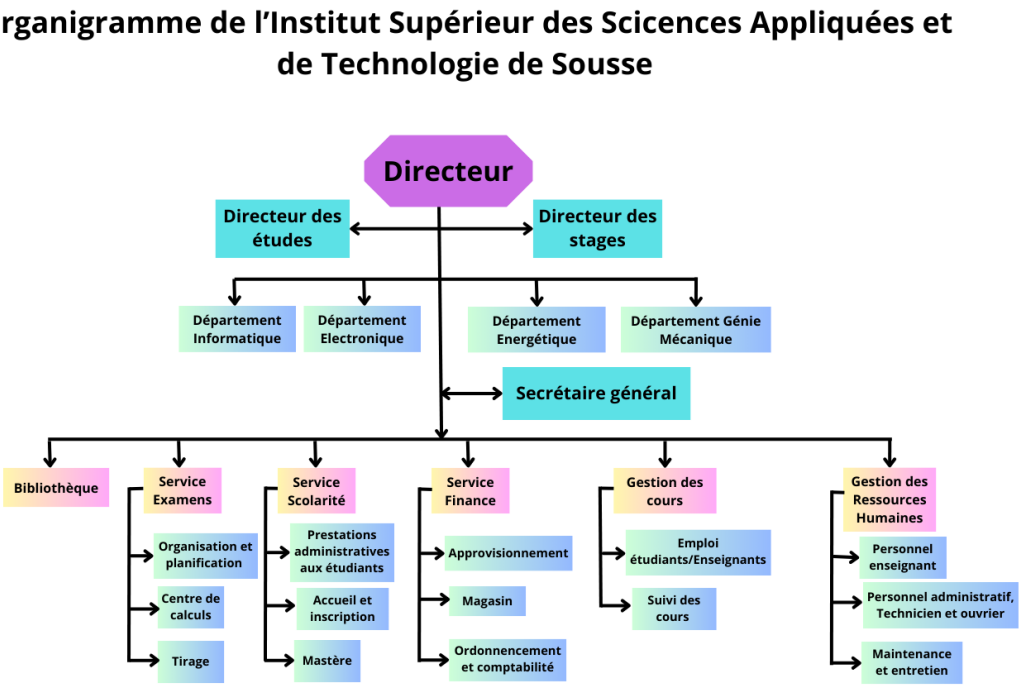

Figure 1.2 : Organigramme de ISSAT sousse

L'organigramme de l'ISSAT Sousse est structuré autour de plusieurs entités clés :

- **Direction générale :** Placée sous la responsabilité du directeur de l'institut, cette entité supervise l'ensemble des activités administratives et académiques.

- **Secrétariat général :** Assiste le directeur dans la gestion quotidienne et assure la coordination entre les différents services.
- **Départements académiques :** L'ISSAT Sousse comprend quatre départements qu'on a déja présenté dans la partie précédente.
- **Services administratifs :** Ces services soutiennent les fonctions académiques et incluent la scolarité, les ressources humaines, la gestion des enseignants...
- **Conseil scientifique :** Composé de membres académiques, ce conseil est chargé de la validation des programmes d'études et de la supervision des activités de recherche.

1.3 Etude et critique de l'existant

1.3.1 Problématique

La planification de la surveillance des examens est une tâche chronophage et complexe pour les établissements d'enseignement supérieur. Elle doit prendre en compte un ensemble de contraintes telles que la disponibilité des enseignants, la répartition équitable des charges de surveillance, les conflits d'horaires et les imprévus de dernière minute.

1.3.2 Etude de l'existant

Dans le cadre de la surveillance des examens, plusieurs établissements continuent d'utiliser des méthodes manuelles ou semi-automatisées pour planifier les emplois du temps des surveillants. Ces méthodes reposent généralement sur des fichiers Excel, des documents papier ou des tableaux imprimés ou de même des échanges par e-mail entre l'administration et les enseignants. Bien que relativement simples à mettre en place, ces solutions présentent de nombreuses limites :

- manque de prise en compte des contraintes individuelles des enseignants, comme leurs disponibilités ou leurs préférences
- risque d'erreurs humaines comme les oublis, les doublons ou les conflits d'horaires
- difficulté à faire des modifications de dernière minute
- absence de centralisation et automatisation nuit à l'efficacité de l'organisation

Certaines plateformes plus avancées existent, mais elles sont souvent coûteuses, peu personnalisables ou trop complexes pour répondre à des besoins spécifiques comme la gestion des indisponibilités, des conflits d'horaires ou la répartition équitable des surveillances.

1.4 Critique de l'existant

Ces différentes approches manuelle ou semi-informatisée présente plusieurs limites importantes. Elle est lente, demande un effort considérable de coordination et reste très sensible aux erreurs humaines, notamment lors de la saisie ou de la modification des plannings. L'absence d'une solution centralisée et automatisée empêche une gestion fluide et efficace du processus, ce qui impacte directement l'organisation des sessions d'examen.

1.5 Solution proposée

Cette situation met en évidence le besoin d'un système adapté et automatisé capable de générer un plannings de surveillance optimisé tout en respectant les contraintes pédagogiques et humaines.

Pour cela, nous proposons de concevoir et de réaliser un système informatique dédié à la génération automatique de table de surveillance des examens. Ce système tiendra compte des contraintes spécifiques des enseignants (disponibilités, charge de travail équitable, préférences...) tout en assurant une répartition cohérente et optimisée des surveillants.

L'application offrira une interface conviviale permettant à l'administrateur de créer et visualiser le planning. Les enseignants pourront consulter leur emploi de temps. Grâce à une automatisation intelligente, la solution permettra de gagner en efficacité, en fiabilité, tout en réduisant considérablement le temps consacré à cette tâche par l'administration.

1.6 Méthodologie adoptée

Une méthodologie de développement est un cadre utilisé pour structurer, planifier et contrôler le développement d'une application [14]. C'est le fait de modéliser un système avant sa réalisation afin de mieux comprendre son fonctionnement et assurer sa cohérence.

Afin d'assurer la bonne gestion et avancement du projet, nous avons décidé d'opter pour la méthodologie Agile Scrum. En effet, ce modèle favorise la communication et le travail d'équipe afin d'apporter une amélioration continue aux livraisons de produits grâce à des processus itératifs et incrémentaux, le tout dans le but de satisfaire les clients.

La méthodologie Agile Scrum se caractérise par le travail de groupe, ses réunions quotidiennes et le suivi continu des projets, car il est plus facile de faire face à des changements fréquents. Ce modèle garantit la livraison de produits de qualité conformément aux besoins et aux attentes des clients.

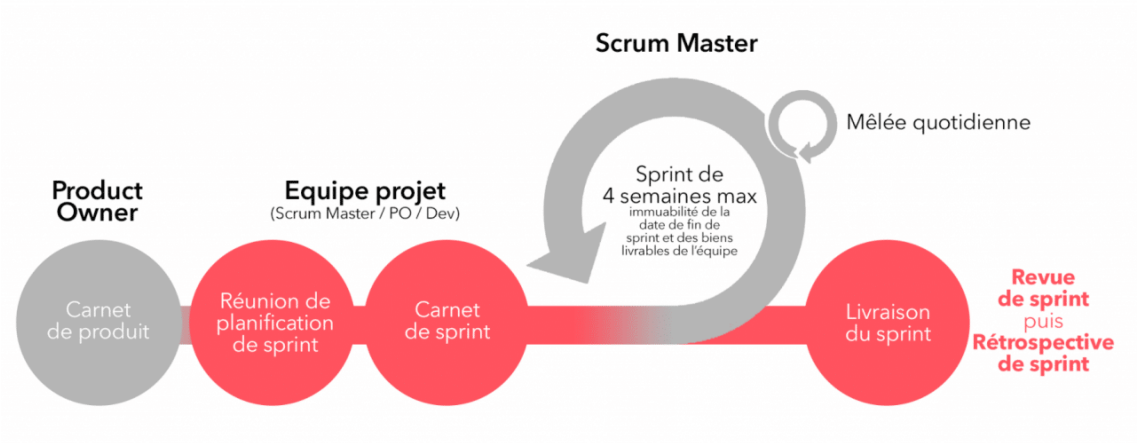

Figure 1.3 : Méthodologie Agile Scrum [2]

La figure 1.3 décrit le flux de travail avec Scrum. Dans notre cas, le projet est réalisé en monôme. Donc, nous essayons d'adapter Scrum à nos besoins. Ainsi, nous utiliserons l'ensemble des bonnes pratiques suivantes :

✓L'extraction des fonctionnalités à développer après la réalisation du Backlog du produit.
✓le découpage du projet en un ensemble de sprints.
✓le découpage de chaque sprint en un ensemble de tâches.

✓une réunion quotidienne avec l'encadrant pour le suivi de l'avancement.
✓une réunion hebdomadaire au cours de laquelle on essaie de répondre à ces trois questions :

- Quelles sont les tâches effectuées ?
- Quelles sont les difficultés rencontrées ?
- Quelles sont les tâches futures à réaliser ?

Dans le cadre de notre projet, **Madame Nawel** a assuré le rôle de *Scrum Master*. Elle a encadré et guidé notre équipe tout au long du processus de développement en facilitant la communication entre les membres, en éliminant les obstacles et en veillant au respect de la méthodologie Scrum.

Pour maintenir un rythme de travail constant et assurer le bon avancement du projet, des **réunions hebdomadaires** ont été organisées. Ces réunions ont permis de faire le point sur les tâches réalisées, d'identifier les difficultés rencontrées et de planifier les actions à venir.

Grâce à cette organisation agile, notre équipe a pu progresser efficacement, s'adapter aux éventuels changements et assurer une collaboration fluide et continue, tout en respectant les délais fixés et en garantissant la qualité des livrables.

Conclusion

Dans ce chapitre, nous avons tout d'abord présenté l'organisme d'accueil. Ensuite nous avons exposé l'étude de l'existant avant de finir par une description de la méthodologie adoptée et la planification des différentes tâches à faire. Dans le chapitre suivant, nous allons entamer par la première phase de conception de notre projet qui consiste à définir les divers acteurs and les principales fonctionnalités de notre application.

Chapitre 2

Analyse et spécification des besoins

Introduction

Ce chapitre vise à présenter les besoins fonctionnels et non fonctionnels attendues de notre application, ainsi que ses principaux utilisateurs. Cette étude est cruciale car elle permet de garantir que les exigences définies dans le cahier des charges sont respectées et que le périmètre de la solution est clairement défini. En d'autres termes, cette analyse vise à assurer que l'application développée réponde aux attentes des utilisateurs et soit en mesure de répondre aux besoins identifiés.

2.1 Identification des acteurs

Tout système interactif, doit assurer et faciliter l'interaction entre ses utilisateurs (utilisateur humain ou non). Un acteur représente le rôle d'une entité externe exploitant le système à travers ses différentes interfaces. Nous spécifions dans le cadre de ce projet les acteurs du système, qui se divisent en trois catégories comme suit :

- **Administrateur**
- **Professeurs**
- **Etudiants**

2.2 Besoins fonctionnels

Les fonctionnalités principales que notre application doit fournir selon le type des acteurs sont :

- **Administrateur :**
  - ➢ Créer table de surveillance des examens

➢ Créer des actualités

- **Enseignant :**
  - ➢ Consulter son planning de surveillance
  - ➢ Consulter les actualités
- **Étudiant**
  - ➢ Consulter les actualités

2.3 Besoins non fonctionnels

Les spécifications non fonctionnelles sont les exigences non liées à la fonctionnalité du système, mais définissent plutôt la manière dont le système doit fonctionner. Notre application doit fournir les nécessités suivantes afin d'assurer de meilleurs résultats et d'être plus efficace.

- **Fiabilité**
  Le système doit garantir une affectation correcte sans conflits d'horaires ni double planification pour un même enseignant
- **Sécurité**
  L'application doit être bien sécurisée pour la protéger contre des menaces et garder les données confidentielles. Ainsi, chaque enseignant a accés à sa propre table de surveillance et l'administrateur seul, peut consulter le planning final.
- **Disponibilité**
  Le système doit être accessible pendant toute la période de préparation et de déroulement des examens. Ainsi, les données doivent être sauvgardé pour éviter toute perte d'information.
- **Utilisabilité**
  L'interface doit être intuitive et simple à utiliser pour des utilisateurs non techniques.

2.4 Diagramme de cas d'utilisation

Un cas d'utilisation permet de structurer les besoins des utilisateurs et les objectifs correspondants d'un système. L'étude approfondie des spécifications des besoins permet de dégager plusieurs cas d'utilisation. Nous représentons dans la figure 2.1 ci-dessous, le diagramme de cas d'utilisation globale de notre application qui nous présente une vue générale du comportement de notre plateforme.

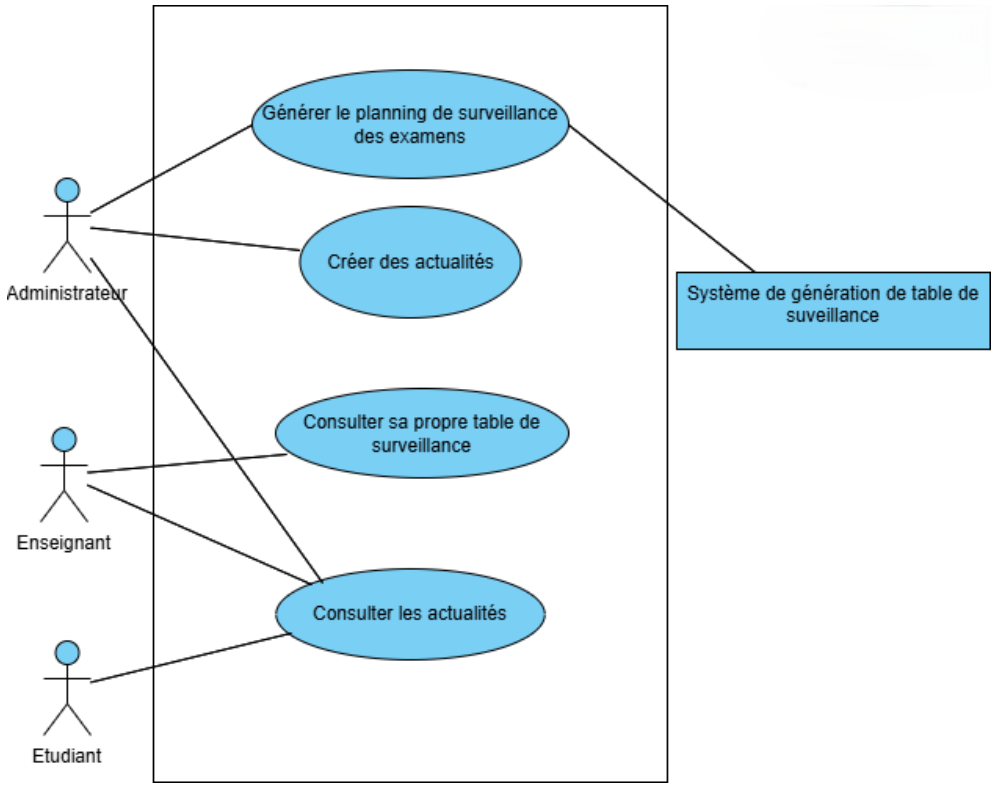

Figure 2.1 : Diagramme de cas d'utilisation globale

2.5 Description détaillée des cas d'utilisation principaux

- **Génération de planning de surveillance des examens**

Le tableau ci-dessous illustre une description textuelle détaillée du cas d'utilisation "Génération de planning de surveillance".

Titre	Génération de planning de surveillance
Objectif	Générer automatiquement un planning de surveillance des examens en prenant en compte des contraintes et des paramètres d'entrée.
Acteurs	Administrateur
Pré-condition	- Les fichiers nécessaires sont disponibles (fichier d'informations sur les enseignants)
Post-Condition	- Un planning valide est généré automatiquement. - Il est disponible sous forme de fichier Excel. - Il peut être consulté par l'administrateur.

Scénario nominal	1. L'administrateur accède à l'espace de gestion de planning 2. Il clique le bouton "Générer le planning de surveillance". 3. L'algorithme traite les données et génère un fichier Excel de planning 4. Le fichier est affiché sur l'écran
Scénario d'exception	- Si un ou plusieurs fichiers sont manquants ou corrompus, le système affiche un message d'erreur

TAB. 2.1 : Description textuelle du cas d'utilisation "Générer le planning de sureveillance des examens"

- **Créer des actualités**

Le tableau ci-dessous illustre une description textuelle détaillée du cas d'utilisation "Créer des actualités".

Titre	Créer des actualités
Objectif	Permettre à l'administrateur de publier des actualités ou annonces visibles par les utilisateurs du système
Acteurs	Administrateur
Pré-condition	L'utilisateur doit être authentifié et posséder le rôle d'administrateur
Post-Condition	L'actualité est créée et devient immédiatement visible par les utilisateurs
Scénario nominal	1. L'administrateur se connecte à l'application avec ses identifiants. 2. Il accède à l'espace d'administration. 3. Il sélectionne la section "Actualités". 4. Il clique sur le bouton "Créer une actualité". 5. Il saisit le titre, le contenu et éventuellement la date de publication. 6. Il valide la création de l'actualité. 7. L'actualité est enregistrée et immédiatement affichée aux utilisateurs.

Tab. 2.2 : Description textuelle du cas d'utilisation "Créer des actualités"

Conclusion

Dans ce chapitre nous avons défini les différents acteurs et fonctionnalités du système. Pour parvenir à un tel résultat, nous avons commencé par spécifier les besoins fonctionnels et non fonctionnels de l'application d'une manière générale. Ensuite pour bien comprendre les fonctionnalités de notre système, nous avons présenté le diagramme de cas utilisation global du système, puis nous l'avons complété et détaillé à l'aide de diagrammes de cas d'utilisation plus raffinés correspondant à la description des différentes fonctionnalités offertes par le système. Le chapitre suivant sera consacré à l'architecture et la conception détaillée de notre application.

Chapitre 3

Etude conceptuelle

Introduction

Après avoir analysé nos besoins dans le chapitre précédent, nous passons vers une phase importante and indispensable dans le cycle de vie de toute application, elle s'agit de la phase de conception. Cette étape est primordiale afin d'assurer le bon déroulement du projet, son objectif consiste à détailler les tâches à entreprendre et à préparer les bases pour l'étape de réalisation. Nous commençons dans la première partie du chapitre par décrire l'architecture physique et logicielle globale de notre application. Ensuite, dans la deuxième partie, nous détaillons l'aspect statique et dynamique de l'application en présentant quelques diagrammes.

3.1 Architecture globale de l'application

3.1.1 Architecture 3-tiers

L'évolution des systèmes d'information repose sur une meilleure subdivision des tâches afin de permettre aux utilisateurs finaux d'exploiter les données de manière plus efficace. Par conséquent, de nombreuses architectures sont apparues afin d'améliorer la qualité des applications. Pour notre projet, nous avons choisis l'architecture 3-tiers.

Dans le contexte des systèmes d'information à trois tiers, la subdivision des tâches se reflète dans la séparation des couches de présentation, de logique métier et de données. Chaque couche est responsable d'un ensemble spécifique de tâches, ce qui permet une gestion plus efficace des données et une meilleure exploitation par les utilisateurs finaux, nous parlons donc d'une architecture distribuée.

Comme nous l'avons mentionné, l'architecture 3-tiers se compose de trois couches :

Couche présentation : Cette couche se concentre sur la collecte des données et la présentation des résultats. Elle facilite l'interaction entre l'utilisateur et l'application

grâçe aux interfaces qu'elle offre.

Couche logique métier : Représente le serveur du système. Elle assure la communication entre les deux autres couches, elle traite les informations collectées de la couche de présentation en implémentant les règles et les algorithmes spécifiques à l'application, puis elle envoie les informations traitées vers la couche de données. Dans cette architecture, les couches présentation et données ne peuvent pas communiquer directement entre elles.

Couche données : Elle est chargée de la gestion et de la manipulation des données. Elle assure le stockage, la récupération et la mise à jour des informations, garantissant leur disponibilité pour les autres couches et les utilisateurs finaux.

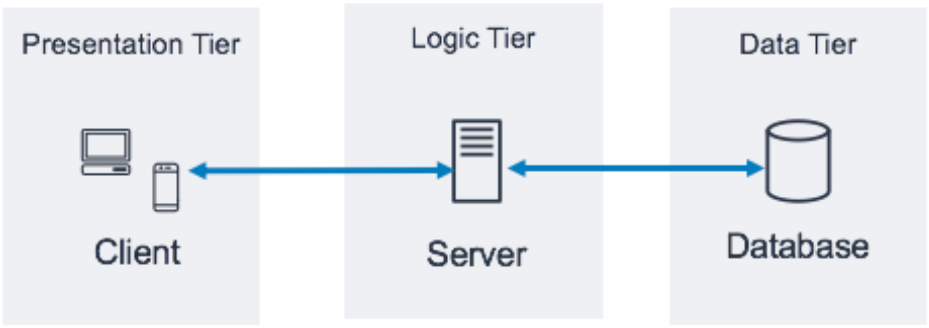

Figure 3.1 : Architecture 3-tiers [3]

L'architecture 3-tiers représentée dans la figure ci-dessus 3.6, montre une séparation qui a pour but de rendre indépendante chacune des couches afin de faciliter la maintenance et les évolutions futures de l'application. De plus, chaque couche communique seulement avec ses couches adjacentes (gauche et droite) et le flux traverse le système de couche en couche de manière continue (pas de saut de couche).

3.2 Modèle MVC

Dans la réalisation de notre projet, nous avons choisis le modèle MVC. Ce patron de conception, très utilisé dans le développement web, permet de séparer une application donnée en trois groupes principaux : les modèles, les vues et les contrôleurs. Chacune de ces couches est conçue pour gérer un aspect de développement spécifique de l'application. De nos jours, le modèle MVC est un pattern architectural largement adopté et considéré comme une meilleure pratique dans le développement de logiciels, il garantit la maintenabilité, la modularité, la réutilisation du code et la rapidité de développement.

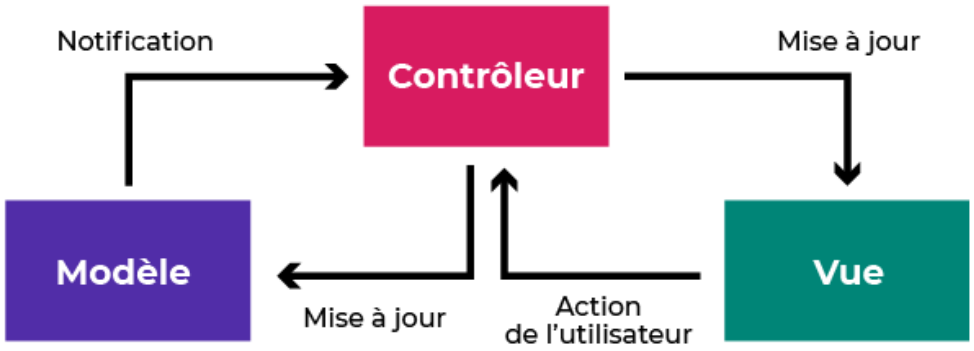

Figure 3.2 : Le modèle MVC [4]

Comme le montre la figure ci-dessus 3.2, le modèle MVC comporte trois couches :

Modèle : Le modèle contient les données et les opérations nécessaires pour les manipuler.

View : La vue est responsable de l'interface utilisateur de l'application. Elle assure l'affichage des données de la couche modèle et la réception de toutes les actions de l'utilisateur pour les transmettre à la couche contrôleur.

Contrôleur : Le contrôleur agit comme un intermédiaire entre le modèle et la vue. Il gère les interactions de l'utilisateur avec l'application, reçoit les entrées de l'utilisateur et les traite. Le contrôleur interagit avec le modèle pour effectuer les opérations appropriées et met à jour la vue en conséquence.

3.3 Conception détaillée

Dans ce qui suit, nous allons entamer la partie conception en utilisant deux types de diagrammes UML : les diagrammes de classe (vue statique) et quelques diagrammes de séquence (vue dynamique).

3.3.1 Etude de la vue statique

3.3.1.1 Diagramme de classe

Le diagramme de classe est un diagramme statique qui représente une vue statique de l'application à travers les différentes classes et les interfaces du système ainsi que les différentes associations entre ces derniers. Une classe est une représentation abstraite d'un ensemble d'objets. Elle contient les informations nécessaires à la construction d'un objet donné via la définition de ses attributs et ses méthodes. Les classes sont utilisées dans la programmation orientée objet [15]. Elles permettent de modéliser et découper une tâche complexe en plusieurs petits modules simples structurées, lisibles et surtout réutilisables.

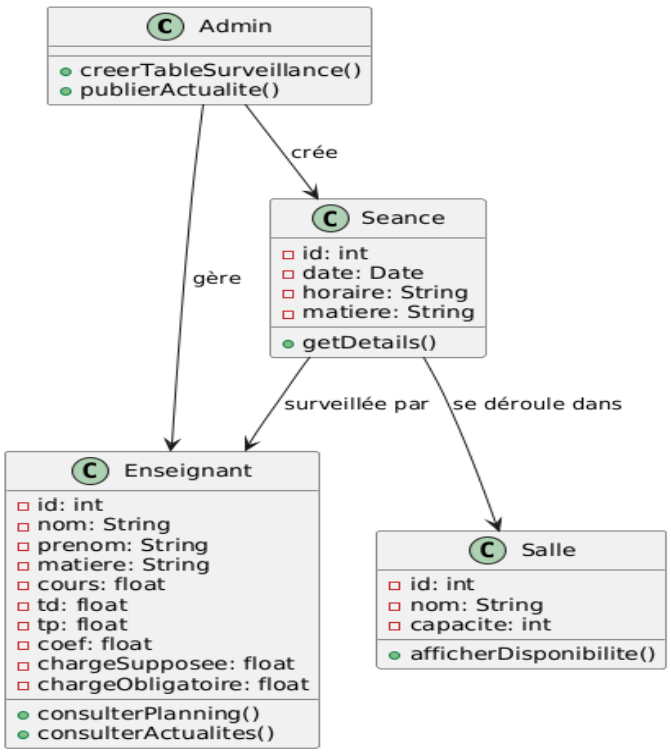

Figure 3.3 : Diagramme de classe

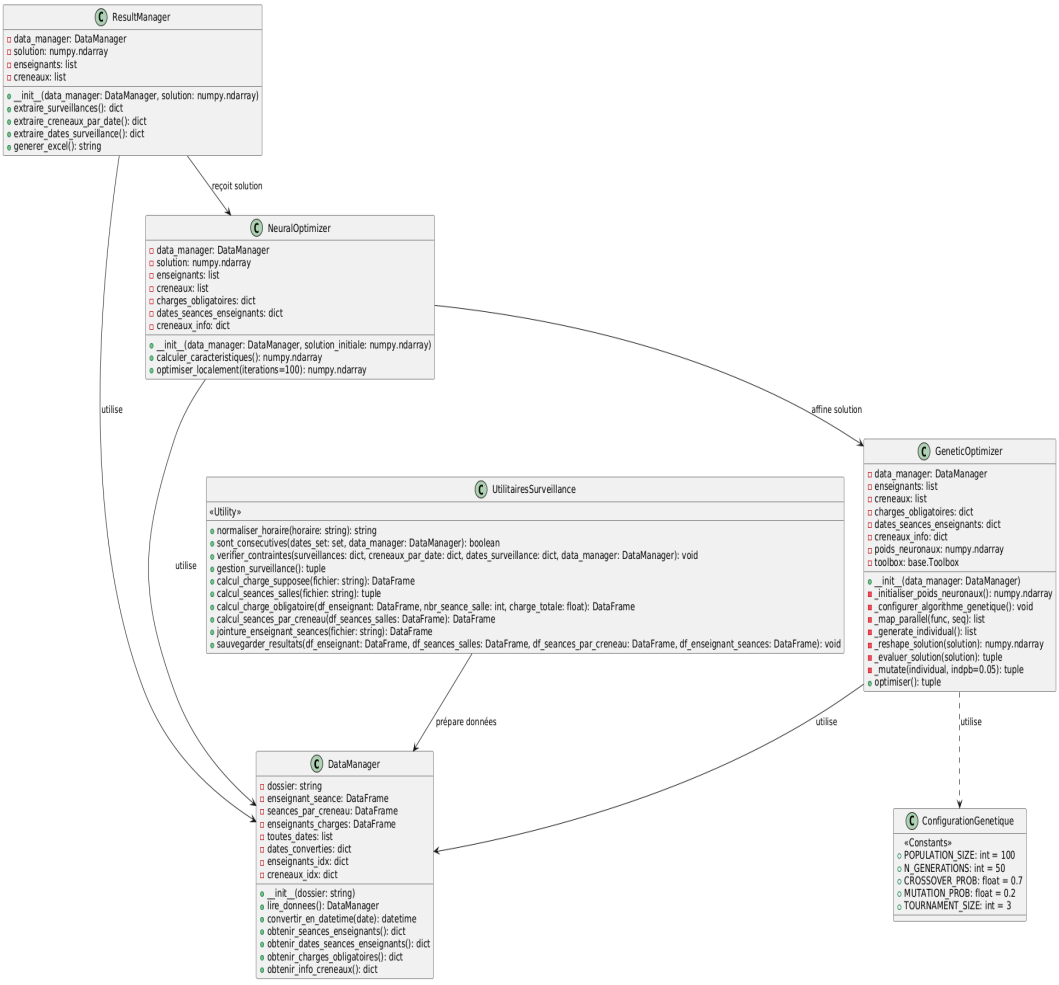

Figure 3.4 : Diagramme de classe de l'algorithme

3.3.2 Etude de la vue dynamique

3.3.2.1 Diagramme de séquence

Un diagramme de séquence permet de représenter les communications avec et au sein du logiciel et des interactions entre les objets [16]. En effet, il montre les interactions selon un point de vue temporel. Dans cette section, nous présenterons quelques diagrammes de séquences pour des scénarios susceptibles d'avoir lieu dans notre système.

- **Diagramme de séquence du cas d'utilisation "Génération de planning"**

Dans ce diagramme de séquence, l'utilisateur interagit avec l'interface utilisateur pour initier la génération du planning. Cette action est transmise au contrôleur qui, à son tour, sollicite le "Système de génération de table de surveillance" en appelant la fonction 'Générerplanning()'. Ce système entreprend alors une série d'opérations pour charger des données à partir de différents fichiers Excel : 'ChargerFichier(surveillance.xlsx)', 'ChargerFichier(enseignantseances.xlsx)', 'ChargerFichier(enseignantcharge.xlsx)', 'ChargerFichier(seancesparcreneau.xlsx)', et 'ChargerFichier(seancessalles.xlsx)'. Il effectue également une opération interne, 'CréerFichiersInformations()'. Après avoir chargé et potentiellement traité ces données via 'traiterDonnéesEtGénérerPlanning()', le système renvoie le fichier de planning généré ('retournerFichierGénéré()') au contrôleur, qui finalement l'envoie à l'interface utilisateur ('envoyerFichier(Planning.xlsx)').

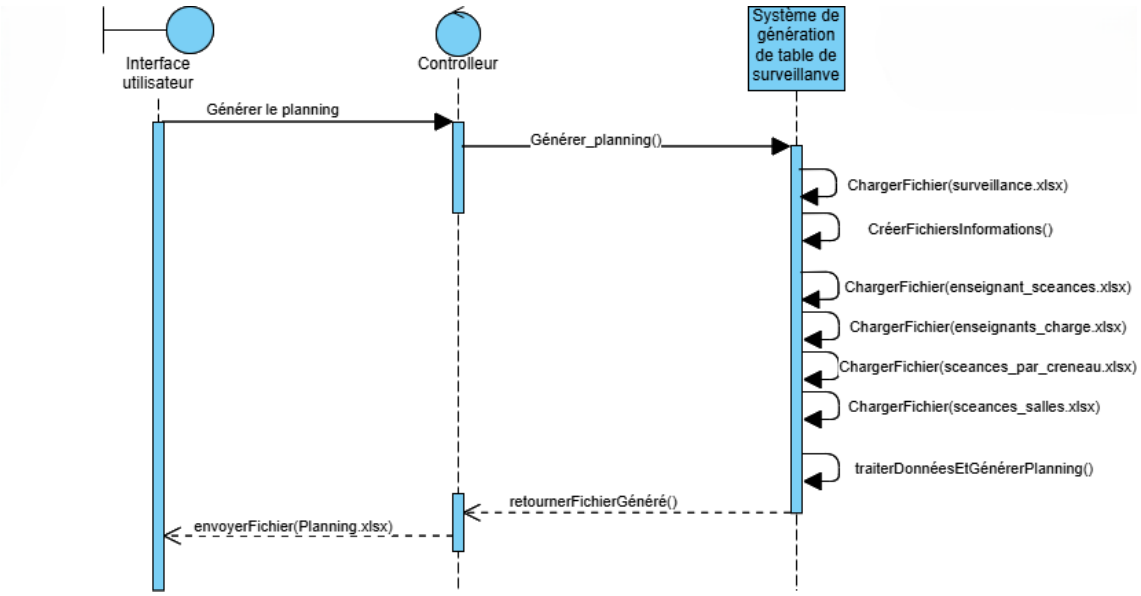

Figure 3.5 : Diagramme de séquence "Génération de planning"

- **Diagramme de séquence du cas d'utilisation "créer actualité"**

Le diagramme de séquence ci-dessus illustre le déroulement du cas d'utilisation "Créer des actualités", depuis l'authentification de l'administrateur jusqu'à l'enregistrement de l'actualité dans la base de données. L'administrateur accède à l'interface web, saisit les informations nécessaires (titre, contenu, date) via un formulaire, puis soumet ces données. Le contrôleur réceptionne la requête, fait appel au service pour valider et enregistrer l'actualité, qui est ensuite persistée dans la base de données. Une fois l'opération réussie, un message de confirmation est renvoyé à l'administrateur. Ce processus assure une gestion centralisée, sécurisée et fluide des publications d'actualités.

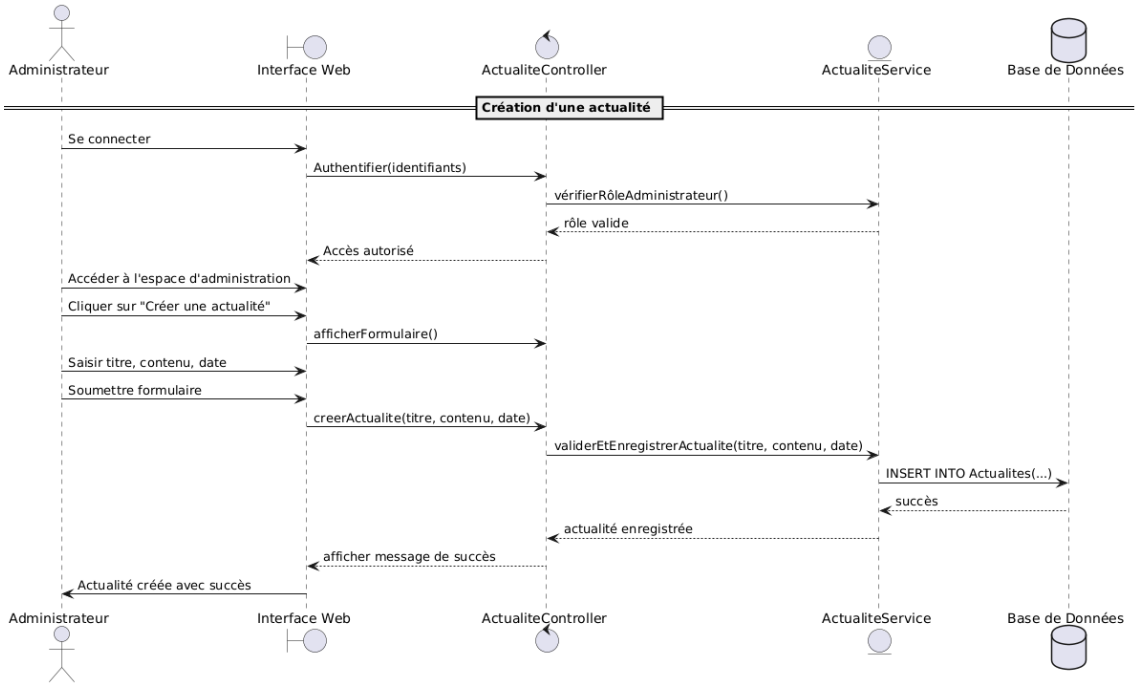

Figure 3.6 : Diagramme de séquence "créer actualité"

Conclusion

Dans ce chapitre nous avons présenté l'architecture générale de l'application, puis nous avons procédé à la conception détaillée en présentant quelques diagrammes. L'étude conceptuelle est une étape primordiale au bon déroulement d'un projet. Elle a pour but de lever toute ambiguïté relative aux détails de notre solution et de préparer le terrain pour l'étape de réalisation. Le chapitre suivant présente l'environnement de développement et les étapes suivies pour la réalisation de la solution.

Chapitre 4

Réalisations

Introduction

Après avoir terminé l'étape de conception, nous abordons dans ce chapitre la réalisation qui est le dernier volet de ce rapport. Dans ce chapitre, qui vise à exposer la phase de réalisation, nous allons spécifier notre environnement de développement, détailler le fonctionnement des technologies utilisées et décrire le travail réalisé.

4.1 Environnement de développement

Dans cette section nous présentons l'environnement matériel mis à la disposition de notre projet ainsi que l'environnement logiciel qui nous a permis la réalisation de notre travail.

4.1.1 Environnement matériel

Lors du développement de la plate-forme, nous avons utilisé un ThinkBook 15 possédant les caractéristiques suivantes :

Système d'exploitation	Windows 10
Processeur	Intel core i5 12th gen
Disque Dur	500Go SSD
Mémoire Vive (RAM)	16GB

TAB. 4.1 : Environnement matériel

4.1.2 Environnement logiciel

4.1.2.1 Outils de développement

4.1.2.1.1 IntelliJ IDEA

IntelliJ IDEA est un environnement de développement intégré (IDE) populaire et puissant créé par JetBrains. Il s'agit d'un outil de développement versatile utilisé principalement pour la programmation dans des langages tels que Java, Kotlin, Groovy et d'autres. Ainsi, il offre plusieurs fonctionnalités qui aident les développeurs à augmenter leur productivité et à améliorer leur expérience de développement comme la vérification de syntaxe en temps réel, la saisie semi-automatique, le débogueur intégré et d'autres.

Figure 4.1 : IntelliJ IDEA [5]

4.1.2.1.2 XAMPP

XAMPP est un environnement de développement local gratuit et open-source qui permet de créer un serveur web complet sur son ordinateur. Il regroupe plusieurs logiciels essentiels, notamment Apache (serveur web), MySQL/MariaDB (base de données), PHP et Perl.

XAMPP est largement utilisé pour tester et développer des applications web en local avant de les déployer sur un serveur en ligne. Grâce à sa simplicité d'installation et à son interface simple, cet outil est extrêmement pratique pour les développeurs, en particulier lors des phases d'apprentissage ou de prototypage.

Figure 4.2 : XAMPP [6]

4.1.2.1.3 Visual Studio Code

Visual Studio Code (VS Code) est un éditeur de code source léger, gratuit et open-source développé par Microsoft. Il est largement utilisé par les développeurs pour sa rapidité, sa simplicité d'utilisation et son extensibilité grâce à un vaste catalogue d'extensions. Il prend en charge de nombreux langages de programmation et inclut un terminal intégré, ce qui en fait un outil complet pour le développement web et applicatif.

Figure 4.3 : Visual Studio Code [7]

4.1.2.1.4 Github

GitHub est une plateforme en ligne de gestion de code source et de collaboration basée sur le système de contrôle de version Git. Elle permet aux développeurs de stocker, suivre les modifications, et collaborer efficacement sur des projets de développement. GitHub facilite le travail en équipe grâce à de nombreuses fonctionnalités intégrées. Il est très utilisé dans le monde professionnel et open-source. C'est est un outil essentiel pour organiser le code, suivre l'évolution des projets et favoriser le travail collaboratif.

Figure 4.4 : Github [8]

4.1.2.2 Outils de conception

4.1.2.2.1 Visual paradigm

Visual Paradigm est un outil UML CASE de modélisation visuelle et de gestion de projets qui sert à concevoir, modéliser et documenter des systèmes logiciels. Nous avons cet outil afin de créer nos diagrammes dans la conception de notre application.

Figure 4.5 : Visual Paradigm [9]

4.1.2.2.2 PlantUML

PlantUML est un outil de modélisation open-source permettant de créer des diagrammes UML à partir d'un simple langage textuel. Il est particulièrement utile pour générer rapidement des diagrammes tels que les diagrammes de cas d'utilisation, de séquence, de classes, et bien d'autres. Cet outil est souvent intégré dans des environnements de développement comme Visual Studio Code ou IntelliJ IDEA.

Nous avons utilisé PlantUML pour concevoir certains diagrammes de notre application, notamment les diagrammes de séquence, grâce à sa simplicité d'utilisation et sa

compatibilité avec le format textuel, ce qui facilite la versioning et la collaboration en équipe.

Figure 4.6 : PlantUML [**plantumllogo**]

4.1.2.3 Technologies

Comme nous l'avons mentionné précédemment, nous avons décidé d'implémenter le type d'architecture de 3- tiers. A présent, nous allons présenter cette structure de point de vue des technologies utilisées dans la décomposition ci-dessous.
-La couche de données sera gérée par phpMyAdmin .
-La couche application sera gérée par Spring boot et Django avec le style d'architecture REST.
-La couche présentation sera réalisée avec Angular.

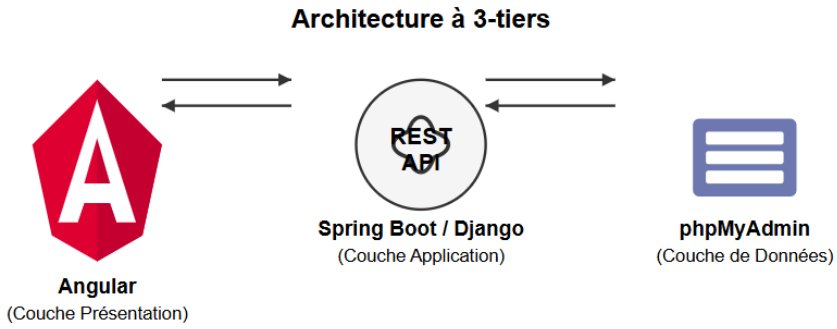

Figure 4.7 : Architecture de l'application cotée technologies

4.1.2.3.1 Spring Boot

Spring Boot est un framework open source basé sur Spring [10], conçu pour faciliter le développement d'applications Java robustes, évolutives et autonomes. Il vise à simplifier la configuration et le déploiement des applications Spring en fournissant des fonctionnalités par défaut et des conventions de configuration intelligentes.

Figure 4.8 : Spring boot [10]

4.1.2.3.2 Django

Django est un framework web open source écrit en Python [**django**], conçu pour le développement rapide d'applications web sécurisées, maintenables et performantes. Il suit le modèle MVC (appelé MVT dans Django : Model-View-Template) et fournit de nombreuses fonctionnalités prêtes à l'emploi, telles que le système d'authentification, l'administration automatique, la gestion des bases de données, et bien plus encore. Django encourage une architecture propre et une séparation claire des responsabilités dans l'application.

Nous avons utilisé Django pour réaliser la partie backend de notre application, notamment pour gérer les utilisateurs, les données et les interfaces d'administration.

Figure 4.9 : Django [**djangologo**]

4.1.2.3.3 Angular

Angular est un framework open-source développé par **Google**, destiné à la création d'applications web dynamiques en utilisant TypeScript. Il permet de construire des interfaces utilisateur réactives et modulaires grâce à une architecture basée sur les composants.

Angular est largement utilisé dans les projets professionnels pour sa scalabilité, clarté de son architecture et son écosystème complet qui inclut l'outil de ligne de commande Angular CLI facilitant la création, le test et le déploiement des applications.

Figure 4.10 : React [**Angularlogo**]

4.1.2.3.4 RestAPI

RestAPI (également appelée RESTful API) est une interface de programmation d'application qui respecte les contraintes du style d'architecture REST et permet d'interagir

avec les services web RESTful [17]. Elle permet à différentes applications et systèmes de communiquer entre eux via des requêtes HTTP comme (GET, POST, PUT, DELETE) en utilisant des formats de données tels que JSON ou XML.

Figure 4.11 : Rest api [11]

4.1.2.3.5 Html5

HTML5 (HyperText Markup Language 5) est la dernière version du langage de balisage utilisé pour structurer et présenter le contenu des pages web. HTML est compris et interprété par tous les navigateurs web modernes. Il offre une compatibilité multiplateforme.

Figure 4.12 : Html5 [12]

4.1.2.3.6 SCSS

SCSS (Sassy CSS) est une extension du langage de style CSS (Cascading Style Sheets). Il s'agit d'une syntaxe plus avancée et puissante qui permet aux développeurs de styliser leurs pages web de manière plus efficace et modulaire. SCSS est une extension de syntaxe de Sass.

Figure 4.13 : Scss [13]

4.2 Réalisation

Dans cette section, nous allons présenter les différentes interfaces de l'application.

4.2.1 Login

Lors de l'ouverture de l'application, la première page qui s'affiche est la page d'authentification. Elle sert à vérifier si l'utilisateur dispose d'un compte ou pas. L'utilisateur fera face à l'interface présentée ci-dessous dans la figure 4.14, où il doit entrer son nom d'utilisateur et son mot de passe pour s'identifier et avoir accès à la plateforme.

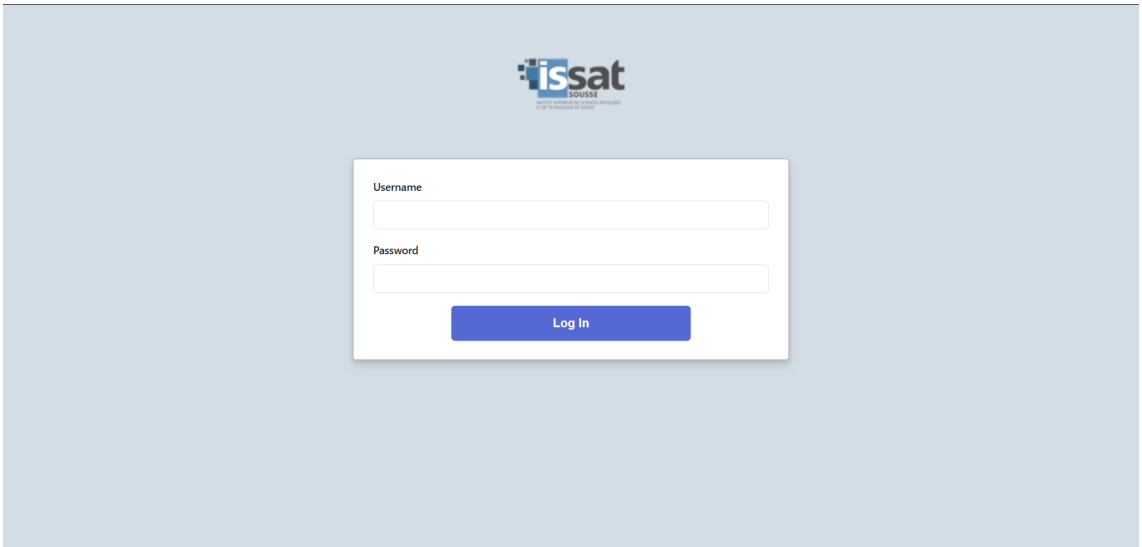

Figure 4.14 : Page d'authentification

4.2.2 Page d'actualités

Une fois l'utilisateur est authentifié, il peut accéder à l'interfaces de notre plateforme. La première interface qu'il va rencontrer, présentée dans la figure ci-dessous 4.15, s'agit de la page d'actualités qui nous permet de naviguer et de consulter les différentes actualités rédigées et publiées par l'administrateur. Ainsi, l'administrateur, seul, posséde un bouton pour la rédaction d'actualités.

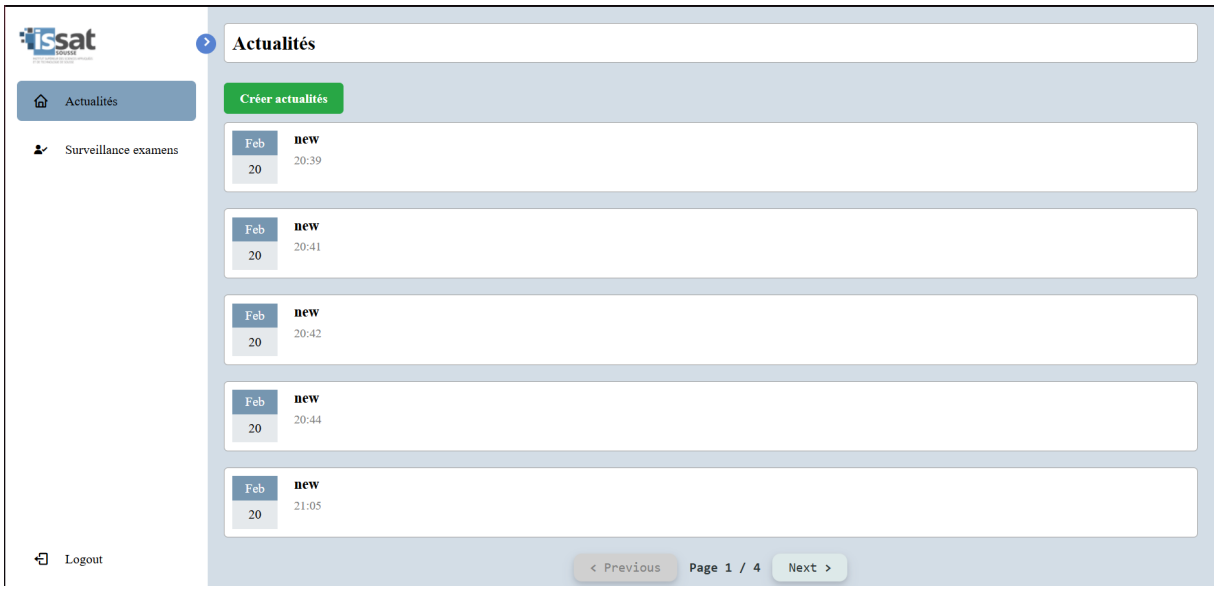

Figure 4.15 : Page d'actualités

4.2.3 Page de création d'actualités

L'interface de publication d'actualité destinée à l'administrateur se présente sous forme d'un formulaire. Elle comprend un champ "Titre" permettant de résumer brièvement l'annonce, un champ "Contenu" de type zone de texte pour saisir la description détaillée de l'actualité, et un champ "Fichier joint" permettant de téléverser un document PDF ou une image (affiche, consigne, formulaire...). Le formulaire inclut également un bouton "Publier" pour valider l'actualité.

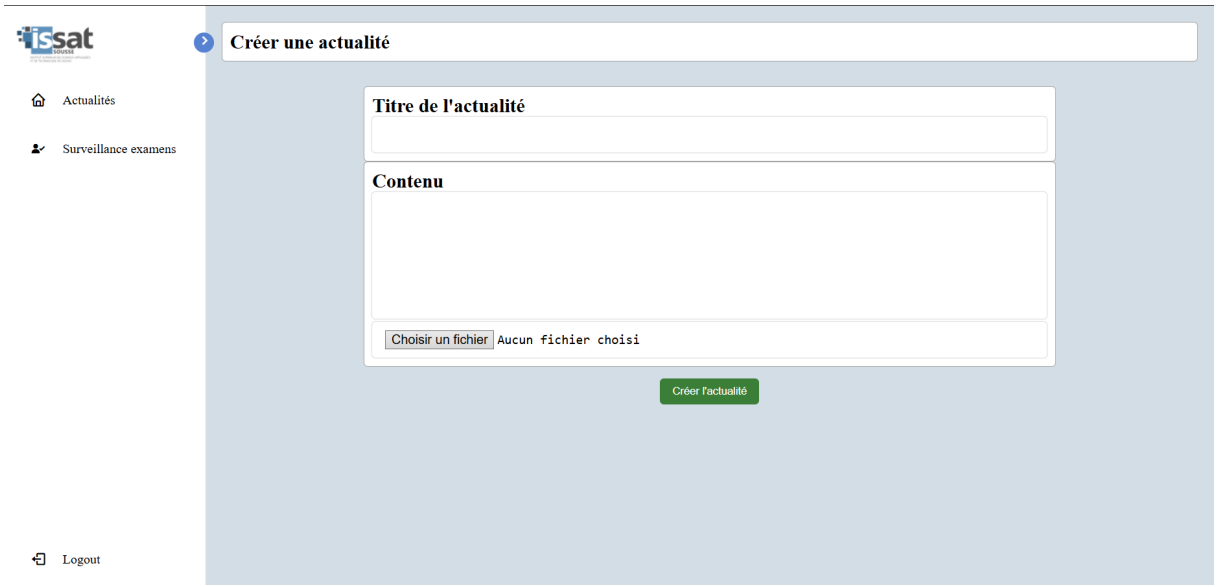

Figure 4.16 : Page de création d'actualités

4.2.4 Page calendrier de surveillance des examens

Cette page, sous la forme d'un calendrier qu'on peut intéraggir avec en utilisant la souris, permet à chaque enseignants de consulter le temps de surveillance qui lui concerne.

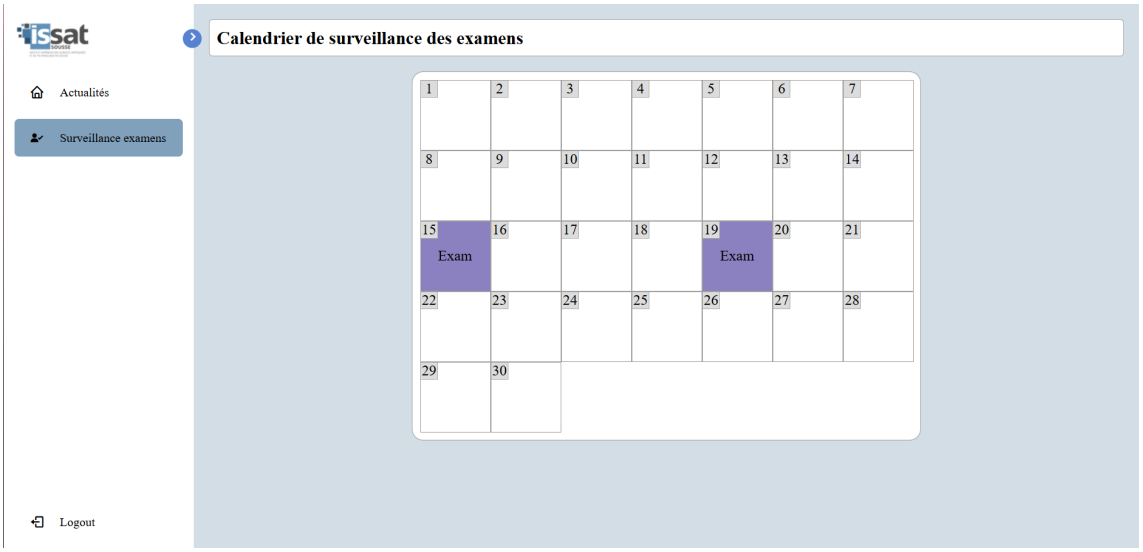

Figure 4.17 : Page calendrier de surveillance des examens

4.2.5 Page de génération de planning de surveillance

Cette interface permet à l'administrateur de générer automatiquement un planning de surveillance des examens, si il éxiste déja, il peut le télécharger ou de le supprimer.

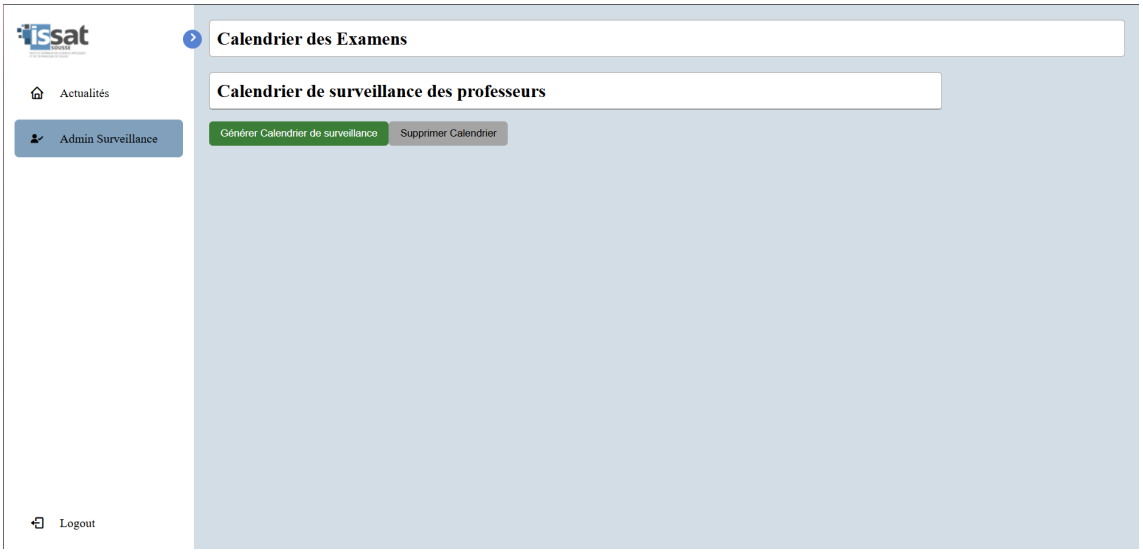

Figure 4.18 : Page de génération de planning de surveillance

Conclusion

Dans ce chapitre, nous avons détaillé la réalisation de notre application. Ainsi, nous avons décrit l'environnement matériel et logiciel utilisé pour la réalisation de l'application. Par la suite, nous avons présenté notre application en l'illustrant par des interfaces des fonctionnalités les plus significatives et importantes offertes par notre plateforme. Quant à la partie suivante, elle sera dédiée pour la conclusion générale de notre rapport ainsi qu'aux perspectives que nous souhaitons réaliser pour les versions qui suivent.

Conclusion et perspectives

Dans le cadre de ce projet réalisé au sein de l'ISSAT, nous avons conçu et développé une application web dédiée à la gestion de la surveillance des examens. Ce système vise à faciliter et automatiser l'organisation des séances de surveillance, l'affectation des enseignants, ainsi que la gestion des salles d'examen. Il répond à un besoin concret de coordination et d'optimisation des ressources humaines et matérielles dans un contexte universitaire souvent marqué par des contraintes de planification.

Notre démarche a débuté par une phase d'analyse de l'existant et de recueil des besoins fonctionnels, suivie par la conception générale et détaillée du système. Nous avons mis en place une architecture modulaire reposant sur des technologies robustes comme Spring Boot et Django pour le backend et Angular pour le frontend. Chaque fonctionnalité a été développée de manière itérative, avec une attention particulière portée à la simplicité d'utilisation, la sécurité des accès et l'efficacité du traitement des données.

Ce projet nous a permis d'approfondir nos compétences techniques et de nous familiariser avec les pratiques professionnelles du développement logiciel, notamment à travers la gestion de projet agile, la modélisation UML, l'intégration d'authentification sécurisée, et la communication avec des bases de données relationnelles. Nous avons ainsi consolidé notre savoir-faire tout en découvrant les réalités du travail en équipe dans un environnement professionnel.

Cependant, notre solution reste perfectible. À l'avenir, plusieurs pistes d'amélioration peuvent être envisagées : ajout d'un système de notification automatique aux enseignants affectés, génération dynamique des plannings au format PDF ou Excel, mise en place d'un tableau de bord analytique pour les administrateurs, ou encore intégration d'un module de reconnaissance faciale ou de badge pour l'enregistrement des présences. Ces perspectives permettront d'enrichir l'application et de la rendre encore plus complète et efficace.

En conclusion, ce projet fut une expérience enrichissante à la fois sur le plan technique et humain, et constitue une étape significative dans notre parcours d'ingénieurs.

Netographie

[1] ISSAT Sousse. https://issatso.rnu.tn/.

[2] Processus Scrum. https://www.sooyoos.com/publication/scrum/.

[3] Architecture 3 tiers. https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html.

[4] Modèle MVC. https://openclassrooms.com/fr/courses/7160741-ecrivez-du-code-python-maintenable/7188702-structurez-une-application-avec-le-pattern-d-architecture-mvc.

[5] IntelliJ IDEA logo. https://www.jetbrains.com/idea/.

[6] XAMPP. https://www.apachefriends.org.

[7] pgAdmin. https://www.pgadmin.org.

[8] GitLab. https://about.gitlab.com.

[9] Visual Paradigm. https://www.visual-paradigm.com.

[10] Spring Boot. https://spring.io/projects/spring-boot.

[11] Rest API. https://www.payoda.com/designing-rest-apis-the-right-way/.

[12] Html5. https://commons.wikimedia.org/wiki/File:HTML5_logo_and_wordmark.svg.

[13] SCSS. https://sass-lang.com.

[14] Méthodologie. https://fr.scribd.com/document/561761314/PFEBI.

[15] Diagramme de classe. https://fr.wikipedia.org/wiki/Diagramme_de_classes.

[16] Diagramme de séquence. https://www.lri.fr/~longuet/Enseignements/16-17/Et3-UML/Et3-5DiagSequence.pdf.

[17] RestAPI. https://www.redhat.com/fr/topics/api/what-is-a-rest-api.