|  |  |
| --- | --- |
| REPUBLIQUE DE COTE D’IVOIRE  armoirie  Union-Discipline -Travail | |
| Ministère de l’Enseignement Supérieur et  de la Recherche Scientifique | **Année académique : 2021/20****22** |
| Université Polytechnique de Bingerville |  |
|  |  |

**MÉMOIRE DE FIN DE CYCLE**

**Pour l’obtention de la**

***Licence de Méthodes Informatiques Appliquées à la Gestion des Entreprises (MIAGE)***

**CONCEPTION ET REALISATION D’API REST DE GESTION DES ABSENCES DES ETUDIANTS CAS DE UPB**

Présenté par :

**SIE PODA JOSIAS**

En Stage du 10/07/2022. Au 14/10/2022.

**Encadrant Académique :**

NIANGORAN Aristhophane Kerandel

Enseignant à l’UPB

**JURY**

|  |  |  |
| --- | --- | --- |
| **PRESIDENT :** | Dr | Enseignant-Chercheur à l’INPHB |
| **RAPPORTEUR :** | M. |  |
| **ASSESSEUR :** | .. |  |
| **MEMBRE :** | …. |  |

|  |  |
| --- | --- |
| REPUBLIQUE DE COTE D’IVOIRE  armoirie  Union-Discipline -Travail | |
| Ministère de l’Enseignement Supérieur et  de la Recherche Scientifique | **Année académique : 2021/2022** |
| Université Polytechnique de Bingerville |  |
|  |  |

**MÉMOIRE DE FIN DE CYCLE**

**Pour l’obtention de la**

***Licence de Méthodes Informatiques Appliquées à la Gestion des Entreprises (MIAGE)***

**CONCEPTION ET REALISATION D’API REST DE GESTION DES ABSENCES DES ETUDIANTS CAS DE UPB**

Présenté par :

**SIE PODA JOSIAS**

En Stage du 10/07/2022. Au 14/10/2022.

**Encadrant Académique :**

NIANGORAN Aristhophane Kerandel

Enseignant à l’UPB

**JURY**

|  |  |  |
| --- | --- | --- |
| **PRESIDENT :** | Dr | Enseignant-Chercheur à l’INPHB |
| **RAPPORTEUR :** | M. |  |
| **ASSESSEUR :** | .. |  |
| **MEMBRE :** | …. |  |

# **Dédicace**

**Je dédie ce travail :**

* **A ma famille, elle qui m’a doté d’une éducation digne, son amour a fait de moi ce que**

**Je suis aujourd’hui.**

* **Particulièrement à ma mère feu DOGUIET Ouenan Solange, pour le goût à l’effort qu’elle a suscité en moi, de par sa rigueur et son amour.**

# **Remerciement**

Je tiens à exprimer toute ma reconnaissance à mon encadrant de mémoire, monsieur **NIANGORAN** **Aristhophane Kerandel**.

Je le remercie de m’avoir encadré, orienté aidé et conseillé.

J’adresse mes sincères remerciement à tous les professeurs, intervenant et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé mes réflexions et ont accepté de me rencontrer et de répondre à mes questions durant mes recherches.

Je tiens à remercier spécialement les professeurs de MERISE, de base de données et d’atelier génie logiciel, qui m’ont fourni les outils nécessaires à la réussite de mes études.

Enfin, je remercie mes amis **BAH Amadou Ali, ANGAMAN Brou Cedric, GBEHE Ange Daniel** et **YAO Damo Ivan**, qui ont toujours été là pour moi. Leur soutien inconditionnel et

leurs encouragements ont été d’une grande aide.

A tous ces intervenants, je présente mes remerciements, mon respect et ma gratitude.

# **Avant-propos**

**Ces dernières années, les évolutions technologiques ont bousculé les modes de consommation du web. Avec l’apparition de nouveaux terminaux tels que les Smartphones, les tablettes et l’arrivée des objets connectés, nos rapports à internet ont évolué. En effet, l’utilisateur souhaite avoir accès au contenu web sur tous ces supports. C’est une des raisons pour lesquels le style d’architecture REST est de plus en plus utilisé dans le développement d’API. REST s’est imposé comme une alternative à SOAP, par sa simplicité et sa souplesse. Dans ce contexte, les développeurs peuvent facilement intégrer leurs données existantes et les rendre accessibles par un web service.**

# **Sommaire**

INTRODUCTION

PREMIERE PARTIE : GENERALITE

CHAPITRE I : CONTEXTE DE L’ETUDE

1. PRESENTATION D’UPB
2. CAHIER DE CHARGE

CHAPITRE II : METHODES DE DEVELOPPEMENT

1. PRESENTATION DES METHODES
2. CHOIX DES METHODES

DEUXIEME PARTIE : ETUDE CONCEPTUELLE

CHAPITRE I : ETUDE PRELIMINAIRE

1. ANALYSE ET CONCEPTION
2. DELIMITATION DU SYSTEME

CHAPITRE II : CAPTURE DES BESOINS FONCTIONNELS

1. DIAGRAMME DES CAS D’UTILISATION
2. DESCRITPTION TEXTUELLE

CHAPITRE III : ANALYSE DU SYSTEME

1. ELABORATION DES DIAGRAMMES DE SEQUENCE
2. ELABORATION DU DIAGRAMME DE CLASSES

TROISIEME PARTIE : IMPLEMENTATION ET RESULTAT

CHAPITRE I : NOTION D’API REST

1. PRESENTATION DES API REST
2. OUTILS D’IMPLEMENTATION

CHAPITRE II : IMPLEMENTATION

1. RESULTAT
2. COUT DU PROJET

CONCLUSION

# **Sigles et abréviations**

**API** : Application Programming Interface.

**REST** : Representational State Transfer.

**HTTP** : HyperText Transfer Protocole.

**URL** : Uniform Resource Locator.

**JSON** : JavaScript Object Notation.

**JWT**: JavaScript Object Notation Web Token.

**UML**: Unified Modeling Language.

**MERISE** : Méthode d’Etude et de Réalisation Informatique pour les Systèmes d’Entreprise.

**SGBD** : Système de Gestion de Base de Données.

**SGBDR**: Système de Gestion de Base de Données Relationnelle.

**SQL**: Structured Query Language.

**UPB**: Université Polytechnique de Bingerville.

**NTIC**: Nouvelles technologies de l’information et de la communication.

**LMD**: Licence Master Doctorat.

**RG** : Règle de Gestion.

# **Liste des Figures**

[Figure 1 : Organigramme de l'UPB 4](#_Toc114649098)

[Figure 2 : Cursus universitaire d’UPB 5](#_Toc114649099)

[Figure 3 : Durée et crédits pour chaque cycle 6](#_Toc114649100)

[Figure 4 : Diagramme de cas d’utilisation 20](#_Toc114649104)

[Figure 5 : Diagramme de séquence d’authentification 32](#_Toc114649105)

[Figure 6 : Diagramme de séquence d’absence 33](#_Toc114649106)

[Figure 7 : Diagramme de séquence de justification 34](#_Toc114649107)

[Figure 8 : Diagramme de séquence de demande d’absence 35](#_Toc114649108)

[Figure 9 : Diagramme de class 37](#_Toc114649109)

[Figure 10 : Architecture client/serveur 1 39](#_Toc114649110)

[Figure 11 : Notion d’API dans l’architecture client/serveur 40](#_Toc114649111)

[Figure 12 : Architecture client/serveur 2 42](#_Toc114649112)

[Figure 13 : Communication client/serveur 44](#_Toc114649113)

[Figure 14 : Ressource absence 49](#_Toc114649115)

[Figure 15 : Ressource administrateur 50](#_Toc114649116)

[Figure 16 : Ressource niveau 50](#_Toc114649117)

[Figure 17 : Ressource justification 51](#_Toc114649118)

[Figure 18 : Ressource filière 51](#_Toc114649119)

[Figure 19 : Ressource enseignant 52](#_Toc114649120)

[Figure 20 : Ressource parent 53](#_Toc114649121)

[Figure 21 : Ressource étudiant 53](#_Toc114649122)

[Figure 22 : Ressource utilisateur 54](#_Toc114649123)

# **Liste des Tableaux**

[Tableau 1 : Tableau des filières 7](#_Toc114063586)

[Tableau 2 : Tableau des modèles MERISE 15](#_Toc114063587)

[Tableau 3 : Tableau comparatif de la méthode MERISE ET du langage UML 15](#_Toc114063588)

[Tableau 4 : Tableau résumant les caractéristiques de notre machine 45](#_Toc114063595)

[Tableau 5 : Tableau récapitulatif des différentes modalités de facturation sur la plateforme Heroku 53](#_Toc114063605)

# **Introduction**

Les technologies de l’information et de la communication ont été la révolution la plus importante qui a marqué ces dernières années, grâce aux applications qui sont devenues une partie indispensable pour le bon fonctionnement des entreprises. Avant l'avènement de ces Technologies de l'Information et de la Communication (TIC), toutes les informations étaient traitées manuellement, causant ainsi de nombreux problèmes, comme la perte de temps ou la diminution de la qualité de l'information, et il était difficile de rechercher cette information et de gérer divers papiers. Pour cette raison, toute organisation moderne qui prétend fonctionner au mieux peut se tourner vers les TIC pour répondre rapidement et efficacement à ses besoins. Par conséquent, ils semblent être un moyen sûr pour une gestion rapide, efficace et efficiente pour tout type de structure.  
 Les universités sont des organisations où les technologies de l’information et de la communication peut beaucoup aider. En effet, depuis 2011 les dispositifs de l’enseignement supérieur et de la recherche scientifique de la Côte d'Ivoire ont fait l’objet d’importantes réformes parmi lesquelles : la mise en œuvre de la réforme Licence-master-doctorat (LMD), l’utilisation des Technologies de l’Information et de la Communication (TIC) dans les pratiques pédagogiques. À travers la mise en œuvre de la réforme LMD, la Côte d'Ivoire veut améliorer la qualité de l’enseignement supérieur et mettre en place une réforme pédagogique qui favorise l’employabilité des diplômés. L’intégration des TICS dans les activités de recherche apparaît ainsi comme une solution fiable pouvant contribuer à minimiser la pression démographique estudiantine exercée sur les infrastructures. Cependant, jusqu'à aujourd'hui, le traitement manuel des informations est encore présent dans les universités, il est donc important d’intégrer les technologies de l'information et de la communication dans le milieu universitaire. C'est également pour faciliter la gestion des absences des étudiants d’UPB qu'un projet intitulé « **Conception et réalisation d’API REST de gestion des absences des étudiants cas de UPB** » nous a été soumis.  
Alors, à quel type d'environnement, notre projet est-il adapté ? Quels sont les besoins réels de nos utilisateurs ? Quelle étape a conduit à la réalisation de notre projet ?  
Afin de répondre à nos questions, nous allons détailler le travail que nous avons effectué en trois (03) sections. En effet, dans la première partie, nous présenterons le contexte et le cadre du projet, de sorte que dans la deuxième partie, nous analyserons les besoins des utilisateurs et concevrons le futur système. Nous terminerons par la troisième partie, qui est dédiée à la réalisation de notre **API REST** et à certains aspects inhérents à cette réalisation.

# **PREMIERE PARTIE : GENERALITE**

## **CHAPITRE I : CONTEXTE DE L’ETUDE**

Dans ce chapitre, nous présenterons l’université Polytechnique de Bingerville, ainsi que notre projet et la solution existante.

### **PRESENTATION D’UPB**

#### **Présentation**

L’Université Polytechnique de Bingerville (UPB) est une nouvelle Université qui a ouvert ses portes en 2018, elle se localise à Abidjan dans la commune de Bingerville, cité CEFAL. Elle envisage se tisser aux rangs des plus grandes institutions africaines de formation supérieure dans les sciences et technologies. Elle compte ainsi répondre au besoin sans cesse croissant de formation de qualité dans le domaine des sciences et technologies en Afrique en général et en Côte d’Ivoire en particulier. On y dispense des enseignements dans les filières scientifiques dont l’informatique, les télécommunications, l’électronique, l’économie, les mathématiques et plusieurs autres formations pratiques.

Dans le contexte de la réforme Licence-master-doctorat (LMD), l’UPB, à l’instar de la plupart des structures d’enseignement supérieur de Côte d’Ivoire, a adapté ses enseignements aux nouvelles exigences qu’impose ladite réforme. Il s’agit entre autres, d’adopter une nouvelle approche de la méthodologie de l’enseignement et de l’apprentissage. Pour ce faire, les programmes de formation sont modelés, avec pour point de mire l’employabilité de nos étudiants.

#### **Organisation de l’UPB**

##### **Les principales Directions et instances**

• **La Direction Générale** dirigée par une administration général rompue, expert dans la gestion administrative.

• **Le Secrétariat Général** dirigé par un professionnel de la haute administration.

• **La Direction Académique** dirigée par un Enseignant-Chercheur rompu dans la gestion des affaires pédagogiques.

• **La Direction de la Scolarité** dirigée par des responsables pédagogiques

• **Le service comptabilité** dirigé par un professionnel de la comptabilité.

• **Les Responsables de filières** qui sont des enseignants permanents travaillant au sein de la direction académique.

• **Le Conseil Scientifique** dirigé par un enseignant-chercheur de rang magistral ayant une longue expérience dans le domaine de l’enseignement, de la recherche et de l’innovation scientifique. Le Conseil Scientifique (CS) est consulté entre autres sur les orientations des politiques de recherche, de documentation scientifique et soutenances. Il est consulté sur les orientations pédagogiques au sein de l’Université. Il donne son avis sur les parcours de formation. Il donne également son avis sur les demandes d’équivalence pour des étudiants extérieurs désirant.

##### **Organigramme de l’UPB**

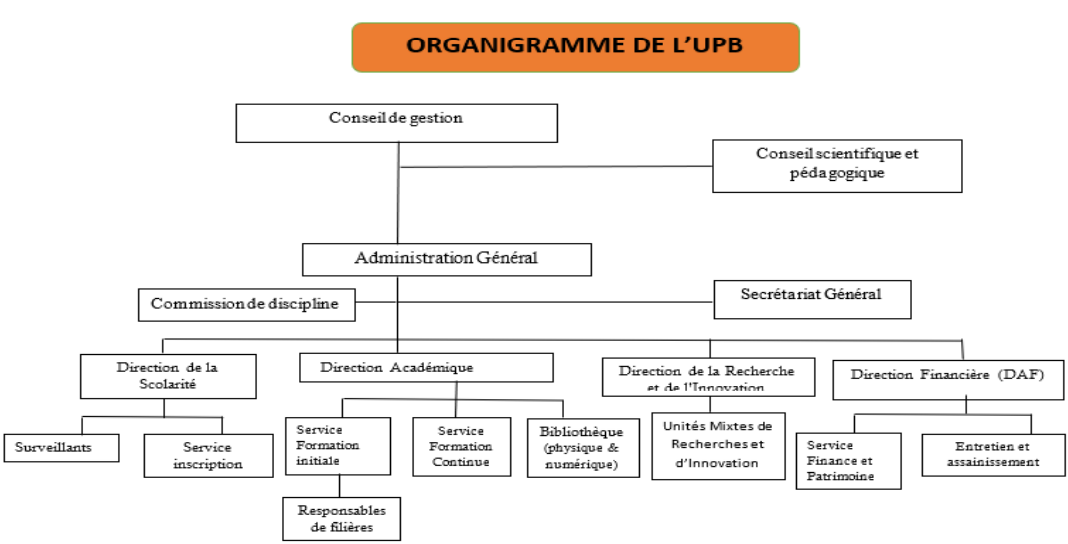


Figure 1 : Organigramme de l'UPB

#### **Enseignements**

##### **Structure des études**

###### **Schéma général du cursus universitaire à l’UPB**

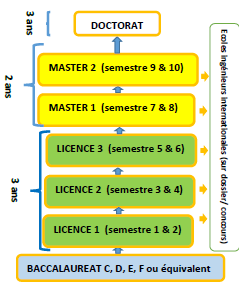


Figure 2 : Cursus universitaire d’UPB

###### **Organisation des enseignements**

L’année académique comprend deux semestres d’études de 16 semaines chacun dont deux semaines réservées aux examens. Le calendrier annuel, les emplois du temps et le calendrier des évaluations élaborés par la direction académique de l’UPB sont communiqués à toutes les parties prenantes (administration, enseignants, apprenants, etc.) sur le site web de l’école et par courriel et/ou par affichage.

###### **Programme des études**

Chaque programme de formation est structuré en Unité d’Enseignement (UE) elle-même composée d’au plus trois (03) Éléments Constitutifs d’Unité d’Enseignement (ECUE) ou matières. Les objectifs pédagogiques et le contenu propre à chaque ECUE concourent à la cohérence globale et aux objectifs de l’UE.

L’UE est affectée d’un nombre de crédits qui représentent le volume global de travail que l’apprenant est supposé fournir.

Selon leur importance, les UE sont classées en deux catégories :

- UE majeures (3 à 6 crédits) ;

- UE mineures (1 à 2 crédits).

La dernière année d’un cycle (licence ou master) inclut un projet interne (PI) et un projet de fin d’études (PFE). Pour le cycle de Master, le PI débute au début du 2d semestre du master 1. Une évaluation régulière des programmes de formation permet de tenir compte des évolutions technologiques et des besoins du marché de l’emploi.

##### **Durée et crédits pour chaque cycle**

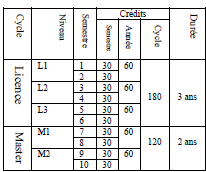


Figure 3 : Durée et crédits pour chaque cycle

#### **Inscription pédagogique**

Les enseignements de chaque parcours type sont obligatoires. L’apprenant est donc d’office inscrit à toutes les UE de son parcours et de son niveau dans le semestre pour un total de 30 crédits. Toutefois, il peut solliciter par courrier adressé au Directeur Académique une modification du nombre de crédits total dans le semestre. Dans ce cas, l’apprenant doit prendre une inscription pédagogique qui indique la liste des UE choisies. Cette inscription conditionne les examens de fin de semestre et d’enseignement des résultats de l’apprenant.

#### **Les filières d’UPB**

|  |  |
| --- | --- |
| **FILIERES** | **SIGLE** |
| Méthodes Informatique  Appliquées à la Gestion des Entreprises | **MIAGE** |
| Administration et Sécurité des  Systèmes et des Réseaux Informatiques | **ASSRI** |
| Electronique Energie Electrique  Automatique | **3EA** |
| Statistique et Economie  Appliquée | **SEA** |
| Sciences Economiques et de Gestion | **SEG** |

Tableau 1 : Tableau des filières

### **CAHIER DE CHARGE**

#### **Contexte du projet**

Jusqu'à aujourd'hui, le traitement manuel de l’information est encore présent dans les universités, causant ainsi de nombreux problèmes comme la perte de temps ou la diminution de la qualité de l’information. Vu ces nombreux problèmes, il est donc important d'intégrer les technologies de l'information dans l’enseignement supérieur.

#### **Objectif général**

L’objectif général est de mettre en place une **API REST** permettant et facilitant la gestion des absences des étudiants de l’Université Polytechnique de Bingerville.

#### **Objectifs spécifiques**

L’**API REST** à mettre en place devra permettre les tâches suivantes :

- Enregistrer un étudiant, un administrateur, un enseignant et un parent.

- Enregistrer un niveau d’étude, une filière.

- Enregistrer une absence d’un étudiant.

- Enregistrer une justification apportée aux absences.

- Faire une demande d’autorisation d’absence.

- Automatiser une demande d’absence.

- Comptabiliser le nombre d’heures d’absences.

- Faciliter l’enregistrement des différents documents justificatif.

- Faciliter la recherche et l’accès aux informations relatives aux absences.

- Donner l’information aux parents quant aux absences de l’étudiant.

#### **Étude de l’existant**

L’analyse de l’existant permet de comprendre la nature du système actuel, décrit la solution présente

du domaine d’étude en terme d’organisation et aussi de rechercher les points forts et les points faibles du système existant.

##### **Analyse de l’existant**

 Dans cette partie, nous présenterons la procédure actuelle de gestion des absences des étudiants d’UPB.

##### **Description du système actuel de gestion des absences.**

L’étude de l’existant nous a conduite à interroger la direction de la scolarité de l’université, et la procédure de gestion des absences est tel que :

- Il y a un responsable de gestion des absences pour chaque niveau.

- Lorsque la semaine arrive, une liste d’émargement est imprimée pour chaque classe et cette liste est ensuite récupérée par le délégué de chaque classe.

- Avant ou après les cours le professeur fait l’appel et coche les absents et la liste d’émargement est déposé à la scolarité.

- Lorsque les listes d’émargements sont déposées à la scolarité, chaque responsable de niveau comptabilise en fin de semaine le nombre d’heures d’absences de chaque étudiant dont il est responsable, si le nombre d’heures d’absence est supérieures à 5 heures les parents de l’étudiant sont informés par le responsable.

- La présence des apprenants est obligatoire à tous les enseignements (cours, travaux dirigés, travaux pratiques, séminaires, visites d’entreprises) et est rigoureusement contrôlée par les enseignants. En cas d’absence à une séance d’enseignement, l’apprenant n’est admis en classe à la séance suivante que sur présentation d’un billet d’entrée délivré par la direction de la scolarité.

Lorsque le nombre d’heures d’absences injustifiées est supérieur ou égal à 5 fois le nombre de crédit, l’ECUE est invalidé.

##### **Critique de l’existant**

Notre étude, nous a permis de trouver quelques points forts et aussi de déceler plusieurs anomalies suivantes :

- Trop de fiches d’émargement à imprimer.

- Volume important des informations traitées manuellement, ce qui provoque parfois des erreurs dans la gestion des absences.

## **CHAPITRE II : MÉTHODES DE DÉVELOPPEMENT**

Cette rubrique est dédiée aux étapes fondamentales pour le développement de notre système de gestion des absences des étudiants. Par conséquent, l'utilisation de méthodes d'analyse et de conception ou de langage de modélisation est essentielle à la réussite de tout projet de développement en informatique. Nous effectuons, ici, le choix d’une méthode ou d’un langage avant d’aborder en profondeur l’étude des besoins des utilisateurs. La phase de conception peut décrire le fonctionnement du futur système de manière claire pour faciliter sa réalisation. Les plus utilisées sont le langage de modélisation unifié UML et la méthode MERISE. Nous présentons ici successivement la méthode MERISE et le langage de modélisation unifié UML avec leurs principales approches et implémentations.

### **PRESENTATION DES METHODES**

#### **Présentation de UML**

##### **Définition de UML**

C’est un langage de modélisation graphique à base de pictogrammes crée en 1997 avec neuf diagrammes, conçus pour représenter, spécifier les artefacts de systèmes logiciels, de plus il est destiné à comprendre et décrire des besoins spécifiés et documentés des systèmes, esquissé des architectures logicielles, concevoir des solutions et communiquer des points de vue, comme il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique.

Il est le résultat de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de Grady Booch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard adopté par l'Object Management Group (OMG) en juillet 2005. Par ailleurs, UML est utilisé pour spécifier, visualiser, modifier et construire les documents nécessaires au bon développement d'un logiciel orienté objet et des bases de données relationnelles ou objets. De plus, il offre un standard de modélisation, pour représenter l'architecture logicielle. Grâce aux outils de modélisation UML, il est aussi possible de générer automatiquement totalement ou partiellement du code d'une application logicielle, à partir des divers documents réalisés. En outre, UML est un langage de modélisation graphique qui permet de représenter les divers aspects duSystème d’information. UML se décompose en plusieurs parties : **Les Vues**, **les Diagrammes** et **les Modèles d’éléments**.

##### **Les vues en UML**

###### **Présentation des vues**

Ce sont les observables du système. Elles décrivent le système d’un point de vue donné, qui peut être organisationnel, dynamique, temporel, architectural, géographique, logique. En combinant toutes ces vues, il est possible de définir ou de retrouver le système complet.

###### **Les différents types de vues**

- **Vue des cas d’utilisation** : c’est la description du modèle vu par les acteurs du système. Elle correspond aux besoins attendus par chaque acteur.

- **Vue logique :** c’est la définition du système vu de l’intérieur. Elle explique comment peuvent être satisfaits les besoins des acteurs.

- **Vue d’implémentation :** cette vue définit les dépendances entre les modules.

**- Vue de processus :** c’est la vue temporelle et technique, qui met en œuvre les notions de tâches concurrentes, stimuli, contrôle, synchronisation.

**- Vue de déploiement** : cette vue décrit la position géographique et l’architecture physique de chaque élément du système.

###### **Les diagrammes en UML**

Un diagramme UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle. Il décrit le contenu des vues, qui sont des notions abstraites. Chaque type de diagramme UML possède une structure et véhicule une sémantique précise. Il existe deux grands types de diagrammes UML : les diagrammes statiques et les diagrammes dynamiques.

###### **Les diagrammes statiques**

Les diagrammes statiques servent à décrire les aspects structurels d’un logiciel ou d’un système d’information et ils présentent plusieurs niveaux d’abstraction et de mise en œuvre.

- **Diagramme de classe UML :** ce modèle, le plus courant dans le développement de logiciels, est utilisé pour décrire la conception logique et physique d’un système et pour présenter ses classes.

- **Diagramme d'objets :** représentation d’une instance du diagramme de classes du système, il est souvent utilisé comme un moyen de vérifier l’exactitude d’un diagramme de classe.

- **Diagramme des composants :** également connu sous le nom de diagramme de flux des composants, il montre les groupes logiques d’éléments et leurs relations. En d’autres termes, il offre une vue plus simplifiée d’un système complexe en le désossant en plus petits composants.

- **Diagramme de structure composite** : il est rarement utilisé en dehors du développement de logiciels. En effet, bien qu’il soit similaire à un diagramme de classes, il va bien plus loin en décrivant la structure interne de plusieurs classes et en présentant leurs interactions.

- **Diagramme de déploiement UML :** Un diagramme de déploiement est un type de diagramme utilisé en UML pour décrire les composants matériels utilisés dans les implémentations de systèmes ainsi que les environnements d'exécution et les artefacts déployés sur le matériel.

- **Diagramme du paquet UML** : Les diagrammes du paquet vous permettent d'organiser les éléments d'un modèle. Ils sont généralement utilisés pour représenter l'organisation de haut niveau d'un projet logiciel. Le diagramme de package peut montrer à la fois la structure et les dépendances entre les sous-systèmes ou les modules.

- **Diagramme de profil :** il s’agit plus d’un langage que d’un diagramme. Un diagramme de profil permet de créer de nouvelles propriétés et sémantiques pour les diagrammes UML en définissant des stéréotypes personnalisés, des valeurs balisées et des contraintes.

###### **Les diagrammes dynamiques**

Ici, l’accent est mis sur les aspects dynamiques du système ou du processus logiciel. Ces diagrammes présentent la fonctionnalité d’un système et mettent en évidence ce qui est prévu dans le système modélisé.

- **Diagramme d’activité :** il s’agit d’un processus pas-à-pas comportant un début et une fin clairs. Ce diagramme comprend un ensemble d’activités qui doivent se produire pour atteindre un but. Il montre comment chaque activité mène à la suivante et la façon dont elles sont toutes liées.

- **Diagramme des cas d’utilisation :** il décrit ce que fait un système, mais pas la manière dont il y parvient. Un cas d’utilisation correspond à un ensemble d’événements qui se produisent lorsqu’un “acteur” utilise un système pour terminer un processus.

- **Diagramme global d’interaction** : souvent complexe, il ressemble au diagramme d’activité puisque les deux montrent une séquence d’activités étape par étape. Un diagramme global d’interaction est en réalité un diagramme d’activité composé de différents diagrammes d’interactions.

- **Diagramme de séquence :** Un diagramme de séquence est utilisé en UML pour décrire les phases d'analyse et de conception. C'est un diagramme d'interaction qui détaille la manière dont les opérations sont effectuées ;

- **Diagramme de communication :** Un diagramme de communication est utilisé pour décrire une communication d'objets qui interagissent pour mettre en œuvre un certain comportement dans un contexte ;

- **Diagramme de l’état :** Les diagrammes d'état vous permettent de modéliser la nature dynamique d'un système. Ils décrivent tous les états possibles d'un objet au fur et à mesure que les événements se produisent.

- **Diagramme de temps :** Les diagrammes de temps vous permettent de faire une représentation des variations d’une donnée au cours du temps.

#### **Présentation de MERISE**

##### **Définition de MERISE**

MERISE est une méthode française née dans les années 70, développée initialement par **Hubert Tardieu**. Elle fut ensuite mise en avant dans les années 80, à la demande du ministère de l'Industrie qui souhaitait une méthode de conception des SI.

MERISE est donc une méthode d'analyse et de conception des SI basée sur le principe de la Séparation des données et des traitements. Elle possède un certain nombre de **modèles** (ou **schémas**) qui sont répartis sur quatre niveaux :

- Le niveau **conceptuel.**

**-** Le niveau **organisationnel**.

- Le niveau **logique.**

- Le niveau **physique**.

##### **Les différents types de démarche en MERISE**

###### **La démarche classique**

C’est la démarche « par défaut » de la méthode. La conception se fait par étapes, afin d'aboutir à un système d'information fonctionnel reflétant une réalité physique. En outre, les données étant séparées des traitements, il faut vérifier la concordance entre données et traitements afin de s’assurer que toutes les données nécessaires aux traitements sont présentes et qu'il n'y a pas de données superflues.

###### **La démarche rapide**

Cette démarche dite « RAD » est apparue au début des années 90, en s'opposant aux démarches en cascade, jugées trop lourdes et trop contraignantes pour le développement d'applications petites et moyennes. Elle ne s’oppose pas complètement à la démarche classique mais préconise plutôt une participation active des utilisateurs, l’exigence d’une maîtrise des coûts et des délais, un cycle itératif de conception/réalisation/amélioration et un contenu fonctionnel restreint et connu du projet.

##### **Les différents modèles en Merise**

|  |  |  |
| --- | --- | --- |
| Niveau | Données | Traitements |
| Conceptuel | Modèle Conceptuel des données (MCD) | Modèle conceptuel des traitements (MCT) |
| Organisationnel | Modèle organisationnel des données (MOD) | Modèle organisationnel des traitements (MOT) |
| Logique | Modèle logique des données (MLD) | Modèle logique des traitements (MLT) |
| Physique | Modèle physique des données (MPD) | Modèle physique des traitements (MPT) |

Tableau 2 : Tableau des modèles MERISE

### **CHOIX DES METHODES**

#### **Etude comparative des méthodes**

Le tableau suivant présente l’étude comparative entre Merise et UML, que nous avons effectuée afin de choisir la méthode d’analyse et de conception ou le langage de modélisation la/le mieux adapté(e) pour notre projet.

|  |  |  |
| --- | --- | --- |
| Critères | MERISE | Unified Modeling Language |
| Traitement des données et opérations | Analyse séparée des données et des opérations. | Regroupement des traitements des données et opérations au sein des classes. Application du principe de l’encapsulation des méthodes orientées objets |
| Niveaux d’abstraction | Plusieurs niveaux : conceptuel, logique, organisationnel, physique avec plusieurs types de modèles : données, traitements, communication. Existence de règles de passage entre les différents niveaux | Niveau unique mais plusieurs types de modèles en fonction de l’aspect à décrire. Affinement des modèles lors des différentes étapes de l’analyse et de la conception. Continuité entre les différentes phases d'élaboration de l'application (traçabilité). |
| Gestion de projet | Concentré principalement sur la conception, sans prise en compte de la phase de programmation (modèle de déploiement et de composants absents). | Prise en compte de tous les stades de la conduite d'un projet. Cohérence de la phase d’analyse jusqu’au code du programme |

Tableau 3 : Tableau comparatif de la méthode MERISE ET du langage UML

#### **Méthode choisie**

L’analyse du tableau ci-dessus permet de faire ressortir plusieurs points. Certes, MERISE offre une démarche d’analyse cohérente et rigoureuse, et est mieux orientée vers la gestion des systèmes d’information avec stockage des données dans des bases de données relationnelles. Mais nous constatons que le langage de modélisation unifié UML basé sur l’approche objet, est utilisé dans l’informatique technique (temps réel) et dans les projets évolutifs, c’est-à-dire pouvant avoir différentes versions. Dans la mesure où notre système est amené à évoluer pour prendre en compte de nouvelles exigences, il nous est apparu judicieux de choisir UML à Merise pour les raisons suivantes :

- **Solution modulaire**

Possibilité de scinder notre projet en plusieurs modules qui produisent des incréments pouvant être traités en séquence pour obtenir des livrables. Ces livrables pourraient être testés, déployés et commercialisés au fur et à mesure. Dans la démarche Merise, c’est à la fin de la dernière étape de la démarche que l’on obtient les livrables.

**- Solution évolutive**

Grâce aux livrables fournis avant la fin du projet, l’architecture globale de la solution peut être ajustée à plusieurs reprises.

- **Déploiement simplifié**

Une bonne représentation du déploiement technique de la solution doit être perçue dès la conception pour faciliter le déploiement des composants et des modules. Merise ne donne pas une représentation des composants techniques sur l’architecture matérielle.

# **DEUXIEME PARTIE : ETUDE CONCEPTUELLE**

## **CHAPITRE I : ETUDE PRELIMINAIRE**

Dans ce chapitre, nous présenterons les différents besoins fonctionnels et opérationnels de notre projet.

### 

### **ANALYSE ET CONCEPTION.**

Cette partie est dédiée aux étapes fondamentales pour le développement de notre système de gestion des absences des étudiants d’UPB. Pour la conception et la réalisation de notre application, nous avons choisi de modéliser avec le langage de modélisation unifié UML pour sa flexibilité marquante qui s'exprime par l'utilisation des diagrammes. Pour la modélisation des besoins, nous utilisons les diagrammes UML suivants : diagramme de cas d'utilisation, diagramme de séquence et le diagramme de classe.

#### **Spécification des besoins**

C'est une étape importante au début de chaque démarche de développement. Son but est de veiller à développer un logiciel adéquat. Sa finalité est la description globale des fonctionnalités du système, en répondant aux questions suivantes :

#### **Besoins fonctionnels**

Notre système doit répondre aux exigences suivantes :

- Le système doit pouvoir afficher toutes les informations sur les absences de chaque étudiant.

- Enregistrer des étudiants.

- Enregistrer des enseignants.

- Enregistrer des administrateurs.

- Enregistrer des filières.

- Enregistrer des parents responsables d’un étudiant.

- Enregistrer une absence.

- Consulter une absence.

- Comptabiliser le nombre d’heures absences.

- Justifier une absence.

- Demander une autorisation d’absence.

- Autoriser une demande d’absence.

- Retirer un étudiant de la base de données

Les tâches telles que la programmation ou l'organisation des cours, l'établissement des emplois du temps des filières, la programmation des activités universitaires, l'établissement des horaires des cours ou des examens, sont donc exclus de notre projet. Les informations relatives à la gestion des professeurs et celles relatives aux inscriptions des étudiants sont aussi exclues de notre projet.

#### **Besoins non fonctionnels**

Les besoins non-fonctionnels sont des besoins optionnels qui améliorent la qualité d’une application. Dans notre cas, nous considérons la sécurité des données, les contraintes de saisie (car l’administrateur, l’étudiant, le parent ou l’enseignant peut par erreur entrer des données erronées).

### **DÉLIMITATION DU SYSTÈME**

1. **Identification des acteurs**

Dans notre application, les acteurs qui interagissent avec le système sont les administrateurs, les étudiants, les parents et les enseignants.

#### **Identification des cas d’utilisation.**

Voici les cas d'utilisation de notre application :

- **Authentifier** : l'application vérifie que l'utilisateur est bien ce qu'il prétend être et lui donne ensuite l'autorisation d’accès.

- **Demander une autorisation d’absence** : pour la demande d’autorisation d’une absence.

- **Justifier une absence** : pour la justification d’une absence enregistrée lors d’une séance.

- **Faire l’appel** : permet à l’enseignant de faire l’appel et de cocher les absents.

- **Consulter les absences** : pour consulter la liste des absents à une séance.

- **Comptabiliser les heures d’absences** : permet à l’administrateur de comptabiliser le nombre d’heures d’absences d’un étudiant.

- **Ajouter un niveau** : sert à ajouter un niveau d’étude.

- **Ajouter une filière** : sert à ajouter une filière.

- **Autoriser la demande** : permet à l’administrateur de valider ou non une demande d’absence.

**La gestion des étudiants** :

- **Ajouter :** pouvoir ajouter des nouveaux étudiants.

- **Modifier :** sert à modifier l'information dans la base de données

- **Supprimer :** sert à supprimer l’information dans la base de données

- **Recherche :** rechercher des informations sur un étudiant, pour pouvoir se renseigner ou renseigner les parents.

## **CHAPITRE II : CAPTURE DES BESOINS FONCTIONNELS**

Dans ce chapitre, il est question d'énumérer dans un premier temps le diagramme des cas d’utilisation détaillés, en second lieu décrire textuellement les principaux cas d’utilisations.

### **DIAGRAMME DES CAS D’UTILISATION**

1. **Diagramme des cas d’utilisation**

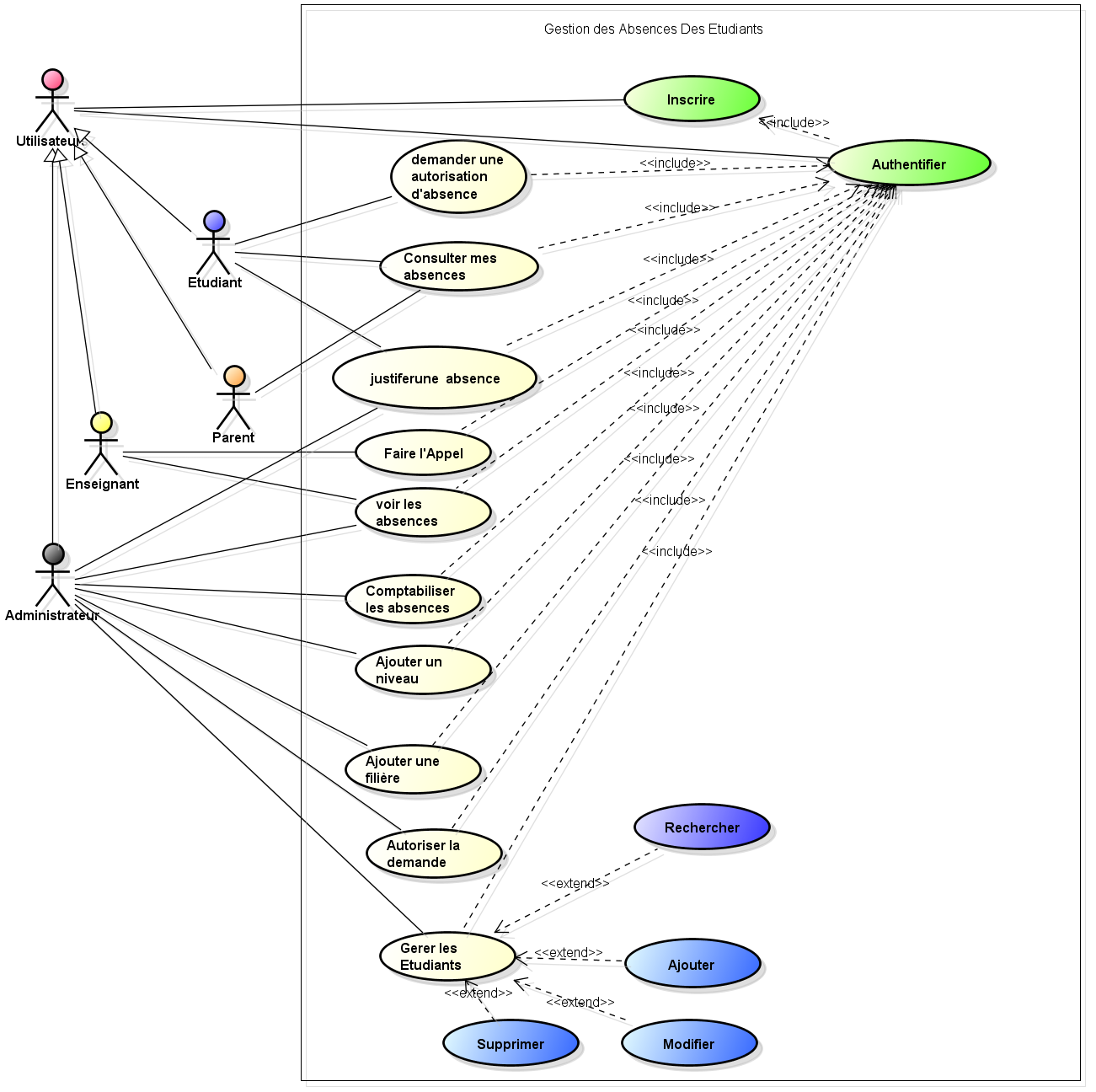


Figure 4 : Diagramme de cas d’utilisation

Pour documenter les cas d’utilisation, la description textuelle est très utile, car elle permet de communiquer aisément avec les utilisateurs. Pour chacun des cas d’utilisation identifiés, nous présentons une fiche de description.

### **DESCRIPTION TEXTUELLE.**

###### **Cas d’utilisation « Authentifier »**

- **Cas d’utilisation** : authentifier

- **Utilisateurs** : administrateur, Étudiant, Parent, Enseignant

- **Objectif** : il permet à l’utilisateur de s’identifier en saisissant son nom d’utilisateur et son mot de passe.

- **Précondition** : l’administrateur, l’étudiant, l’enseignant ou le parent doit être enregistré dans la base de données.

- **Postcondition** :

• Utilisateur authentifié

• La page d’accueil s’affiche.

- **Scénario nominal** :

1. L’utilisateur ouvre l’application.

2. Le système affiche la page d’authentification.

3. L’utilisateur saisit son nom d’utilisateur et son mot de passe.

4. L’utilisateur valide ses coordonnées.

5. Le système vérifie l’existence des données.

6. Le système affiche la page d’accueil.

- **Scénario alternatif** :

A. Erreur d’authentification : nom d’utilisateur ou mot de passe non valide.

Cet enchaînement démarre au point 4.

5. Le système affiche un message d’erreur.

• B. Les champs obligatoires vident.

Le scénario reprend au point 2.

* **Cas d’utilisation « Ajouter »**

- **Cas d’utilisation** : ajouter

- **Acteur** : administrateur

- **Objectif** : ajouter un étudiant dans le système.

- **Précondition** : succès d’authentification.

- **Postcondition** : un nouvel étudiant a été ajouté.

- **Scénario nominal** :

1. L'administrateur clique sur « Ajouter ».

2. Le système affiche le formulaire à remplir.

3. L’administrateur saisit les informations à remplir sur un nouvel étudiant.

4. L’administrateur valide les informations.

5. Le système vérifie les données.

6. Le système enregistre l’étudiant dans la base de données.

7. Le système affiche un message.

- **Scénario alternatif :**

A. Champs obligatoires non-valident ou vident.

Cet enchaînement démarre au point 4.

5. Le système affiche un message d’erreur.

Le scénario reprend au point 2.

* **Cas d’utilisation « Modifier » :**

- **Cas d’utilisation** : modifier les informations relatives à l’étudiant.

- **Acteur** : administrateur

- **Objectif** : il permet à l’administrateur de modifier les informations relatives à l’étudiant.

- **Précondition** :

• Succès d’authentification,

• L’étudiant doit être présent dans le système.

- Postcondition : les informations sont modifiées.

- **Scénario nominal :**

1. L’administrateur choisit d’afficher la liste des étudiants.

2. Le système affiche la liste.

3. L’administrateur choisit l’étudiant pour lequel il veut modifier les informations.

4. Le système affiche le formulaire de modification contenant les informations de l’étudiant à modifier.

5. L’administrateur modifie les informations sur l’étudiant.

6. L’administrateur valide les informations.

7. Le système demande une confirmation de la modification.

8. L’administrateur confirme la modification.

9. Le système vérifie les données.

10. Le système enregistre la modification dans la base de données.

11. Le système affiche un message de modification.

- **Scénario alternatif :**

A. Champs obligatoires non valides ou vide.

Cet enchaînement démarre au point 8.

9. Le système affiche un message d’erreur.

Le scénario reprend au point 4.

* **Cas d’utilisation « Supprimer »**

- **Cas d’utilisation** : supprimer

- **Acteurs** : administrateur

- **Objectif** : il permet à l’administrateur de supprimer un étudiant.

- **Précondition** : l’étudiant doit être présent dans la base de données.

- **Postcondition** : succès authentification.

- **Scénario nominal :**

1. L’administrateur choisit d’afficher la liste des étudiants.

2. Le système affiche la liste.

3. L’administrateur choisit l’étudiant qu’il veut supprimer.

4. L’administrateur demande une suppression.

5. Le système demande une confirmation de suppression.

6. L’administrateur valide la confirmation de suppression,

7. Le système supprime l’étudiant de la base de données.

8. Le système affiche un message de suppression.

- **Scénario alternatif :** aucun.

* **Cas d’utilisation « rechercher »**

- **Cas d’utilisation** : rechercher

- **Acteurs** : administrateur

- **Objectif** : il permet à l’administrateur de rechercher un étudiant dans la base de données.

- **Précondition** : l’étudiant doit être présent dans la base de données.

- **Postcondition** : aucun

**- Scénario nominal :**

1. L’administrateur choisit d’afficher la liste des classes.

2. Le système affiche la liste des classes.

3. L’administrateur choisit la classe de l’étudiant.

4. Le système affiche la liste de la classe.

5. L’administrateur choisit l’étudiant.

6. Le système affiche les informations sur l’étudiant.

- **Scénario alternatif** : aucun.

* **Cas d’utilisation « autoriser la demande »**

- **Cas d’utilisation** : autoriser la demande

- **Acteurs :** administrateur

- **Objectif :** il permet à l’administrateur de valider une demande d’absence d’un étudiant à une séance.

- **Précondition** :

• L’étudiant doit être présent dans le système.

• L’étudiant doit effectuer une demande d’absence.

- **Postcondition :**

• Demande autorisée.

• Demande refusée.

- **Scénario nominal :**

1. L’administrateur choisit d’afficher la liste des demandes.

2. Le système affiche la liste des demandes.

3. L’administrateur choisit une demande.

4. Le système affiche la demande.

5. L’administrateur vérifie et examine les informations relatives à la demande.

6. Le système enregistre la décision de l’administrateur.

- **Scénario alternatif :** aucun.

* **Cas d’utilisation : « ajouter une filière »**

- **Acteurs :** administrateur

- **Objectif** : il permet à l’administrateur d’ajouter une filière dans le système.

- **Précondition** : succès d’authentification

- **Postcondition** : une nouvelle filière a été ajoutée.

- **Scénario nominal :**

1. L'administrateur clique sur « Ajouter une nouvelle filière »

2. Le système affiche le formulaire à remplir.

3. L’administrateur saisit les informations à remplir sur la nouvelle filière.

4. L’administrateur valide les informations.

5. Le système vérifie les données.

6. Le système enregistre la filière dans la base de données.

7. Le système affiche un message d’enregistrement.

- **Scénario alternatif :**

A. Champs obligatoires non-valident ou vident.

Cet enchaînement démarre au point 4.

5. Le système affiche un message d’erreur.

Le scénario reprend au point 2.

* **Cas d’utilisation « ajouter un niveau »**

- **Cas d’utilisation** : ajouter un niveau.

- **Acteurs** : administrateur

- **Objectif :** il permet à l’administrateur d’ajouter un niveau d’étude dans le système.

- **Précondition** : succès d’authentification

**- Postcondition** : un niveau a été ajouté.

**- Scénario nominal :**

1. L'administrateur clique sur « Ajouter un niveau. »

2. Le système affiche le formulaire à remplir.

3. L’administrateur saisit les informations à remplir sur le niveau.

4. L’administrateur valide les informations.

5. Le système vérifie les données.

6. Le système enregistre le niveau dans la base de données.

7. Le système affiche un message d’enregistrement.

- **Scénario alternatif :**

A. Champs obligatoires non-valident ou vident.

Cet enchaînement démarre au point 4.

5. Le système affiche un message d’erreur.

Le scénario reprend au point 2.

* **Cas d’utilisation « comptabiliser les absences »**

- **Cas d’utilisation** : comptabiliser les absences

- **Acteurs** : administrateur

- **Objectif** : il permet à l’administrateur de comptabiliser le nombre d’heures d’absence d’un étudiant à des séances de cours, TD ou de TP.

- **Précondition :**

• Succès d’authentification.

• L’étudiant doit être présent dans le système.

- **Postcondition** : aucun

- **Scénario nominal** :

1. L’administrateur choisit d’afficher la liste des étudiants.

2. Le système affiche la liste des étudiants.

3. L’administrateur choisit un étudiant.

4. Le système affiche l’étudiant.

5. L’administrateur choisit d’afficher la liste des absences de l’étudiant.

6. Le système affiche la liste des absences de l’étudiant.

7. L’administrateur compte les heures d’absence de l’étudiant.

8. Le système enregistre la décision de l’administrateur.

- **Scénario alternatif** : aucun.

* **Cas d’utilisation « Consulter les absences »**

- **Cas d’utilisation** : Consulter les absences

- **Acteurs** : administrateur, Enseignant

- **Objectif** : il permet à l’administrateur ou à l’enseignant de consulter la liste des étudiants absents lors d’une séance.

- **Précondition :**

• Succès d’authentification.

- **Postcondition** : aucun

- **Scénario nominal :**

1. L’administrateur ou l’enseignant choisit d’afficher la liste d’absences.

2. Le système affiche la liste d’absences.

- **Scénario alternatif : aucun.**

* **Cas d’utilisation « Faire l’appel »**

- **Cas d’utilisation** : faire l’appel

- **Acteurs** : enseignant

- **Objectif** : il permet à l’enseignant de faire l’appel et de cocher les absents à son cours.

- **Précondition** :

• Succès d’authentification.

• L’étudiant doit être présent dans le système.

- **Postcondition** : absence enregistrée.

- **Scénario nominal** :

1. L’enseignant choisit d’afficher la liste des niveaux.

2. Le système affiche la liste des niveaux.

3. L’administrateur choisit un niveau.

4. Le système affiche le niveau.

5. L’enseignant choisit d’afficher la liste des filières appartenant au niveau.

6. Le système affiche la liste des filières appartenant à ce niveau.

7. L’administrateur choisit une filière.

8. Le système affiche la filière.

9. L’administrateur choisit d’afficher la liste des étudiants de cette filière.

10. Le système affiche la liste des étudiants.

11. L’enseignant faire l’appel, coche les absents et enregistre,

12. Le système enregistre les informations sur les absences dans la base de données.

- **Scénario alternatif :** aucun.

* **Cas d’utilisation « Justifier une absence »**

- **Cas d’utilisation :** justifier une absence

- **Acteurs** : étudiant, Administrateur

- **Objectif** : il permet à l’étudiant ou à l’administrateur d’apporter une justification à une absence.

- **Précondition** :

• Succès d’authentification.

• L’étudiant doit être présent dans le système.

- **Postcondition** : justification enregistrée.

- **Scénario nominal :**

1. L’administrateur ou l’étudiant choisit d’afficher la liste d’absences.

2. Le système affiche la liste d’absences.

3. L’administrateur ou l’étudiant choisit l’absence à justifier.

4. Le système affiche l’absence.

5. L’administrateur ou l’étudiant clique sur <<Justifier>>

6. Le système affiche le formulaire à remplir.

7. L’administrateur ou l’étudiant saisit les informations à remplir sur la justification.

8. L’administrateur ou l’étudiant valide les informations.

9. Le système vérifie les données.

10. Le système enregistre la justification dans la base de données.

11. Le système affiche un message de confirmation.

**- Scénario alternatif :**

A. Champs obligatoires non-valident ou vident.

Cet enchaînement démarre au point 4.

9. Le système affiche un message d’erreur.

Le scénario reprend au point 2.

* **Cas d’utilisation « consulter mes absences »**

- **Cas d’utilisation** : consulter mes absences

- **Acteurs** : étudiant, parent

- **Objectif** : il permet à l’étudiant ou aux parents de consulter les absents enregistrés lors d’une séance.

- **Précondition** :

• Succès d’authentification.

• L’étudiant doit être présent dans le système.

- **Postcondition** : aucun

- **Scénario nominal** :

1. L’étudiant ou le parent choisit d’afficher la liste d’absences.

2. Le système affiche la liste d’absences.

- **Scénario alternatif** : aucun.

* **Cas d’utilisation « Demander une autorisation d’absence »**

- **Cas d’utilisation** : demander une autorisation d’absence

- **Acteurs** : étudiant

- **Objectif** : il permet à l’étudiant de faire une demande d’autorisation d’absence à une séance.

- **Précondition** :

• Succès d’authentification.

- **Postcondition** :

• Demande enregistrée.

- **Scénario nominal** :

1. l’étudiant clique sur « effectuer une demande »,

2. Le système affiche le formulaire à remplir.

3. L’étudiant saisit les informations à remplir sur la demande.

4. L’étudiant valide les informations.

5. Le système vérifie les données.

6. Le système enregistre la demande dans la base de données.

7. Le système affiche un message de confirmation.

**- Scénario alternatif :**

A. Champs obligatoires non-valident ou vident.

Cet enchaînement démarre au point 4.

5. Le système affiche un message d’erreur.

Le scénario reprend au point 2.

## **CHAPITRE III : ANALYSE DU SYSTÈME**

### **ELABORATION DES DIAGRAMMES DE SÉQUENCE**

#### **Diagramme de séquence du processus d’authentification**

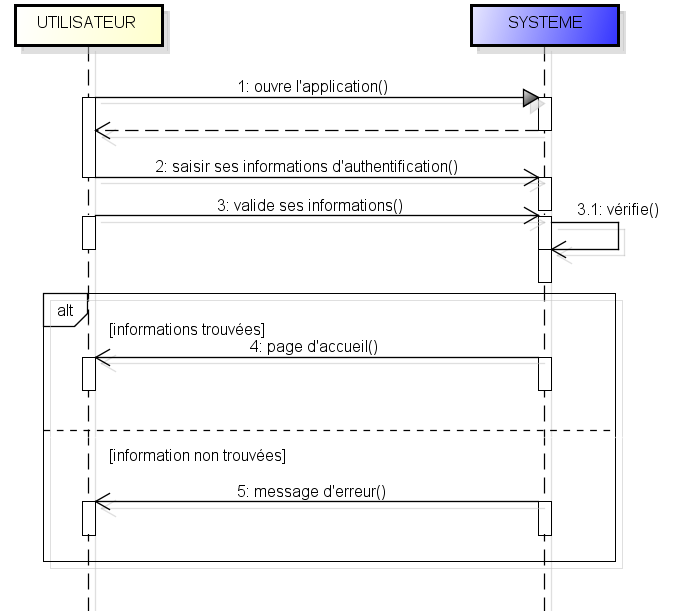


Figure 5 : Diagramme de séquence d’authentification

Ce processus est déclenché par l'administrateur, l’étudiant, le parent ou l’enseignant. Un formulaire s'affiche en premier où il doit faire rentrer son nom d’utilisateur et son mot de passe après validation de la requête le système vérifie ses informations dans la base de données, si elles sont conformes le système ouvre la page d’accueil sinon une interface indiquant un message d’erreur s’affiche.

* + - 1. **Diagramme de séquence du processus d’enregistrement d’une absence**

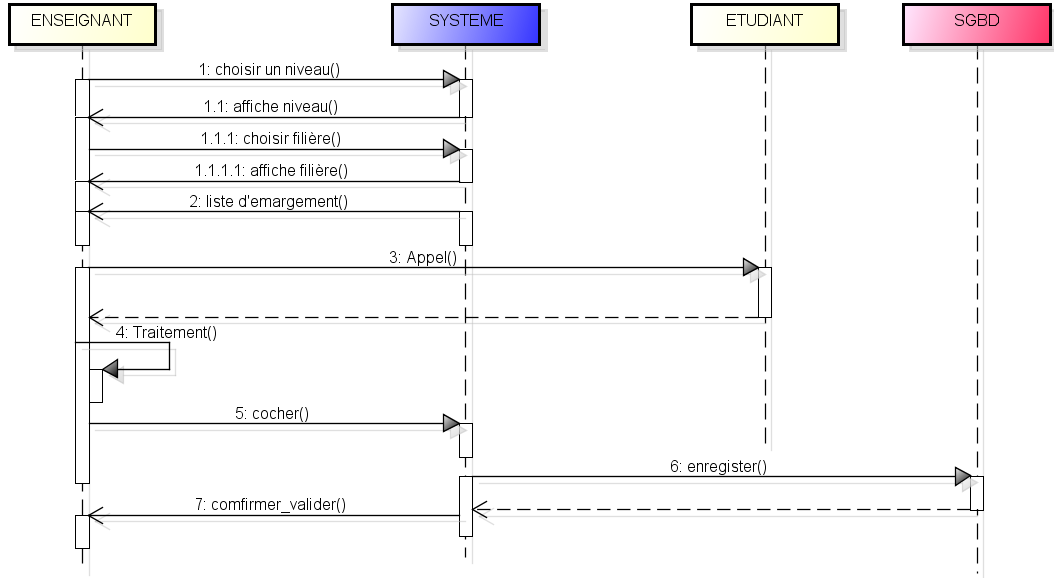


Figure 6 : Diagramme de séquence d’absence

Ce processus est déclenché par l’enseignant. Après l’authentification, l’enseignant choisir un niveau d’étude, le système affiche le niveau d’étude demander et la liste des filières associée, l’enseignant choisir une filière, le système affiche la filière et la liste d’émargement associée à cette filière, ensuite l’enseignant faire l’appel et une réponse est retournée à l’enseignant qui effectue un traitement et enregistrer l’absence si l’étudiant est absent.

* + - 1. **Diagramme de séquence du processus d’enregistrement d’une justification pour une absence**

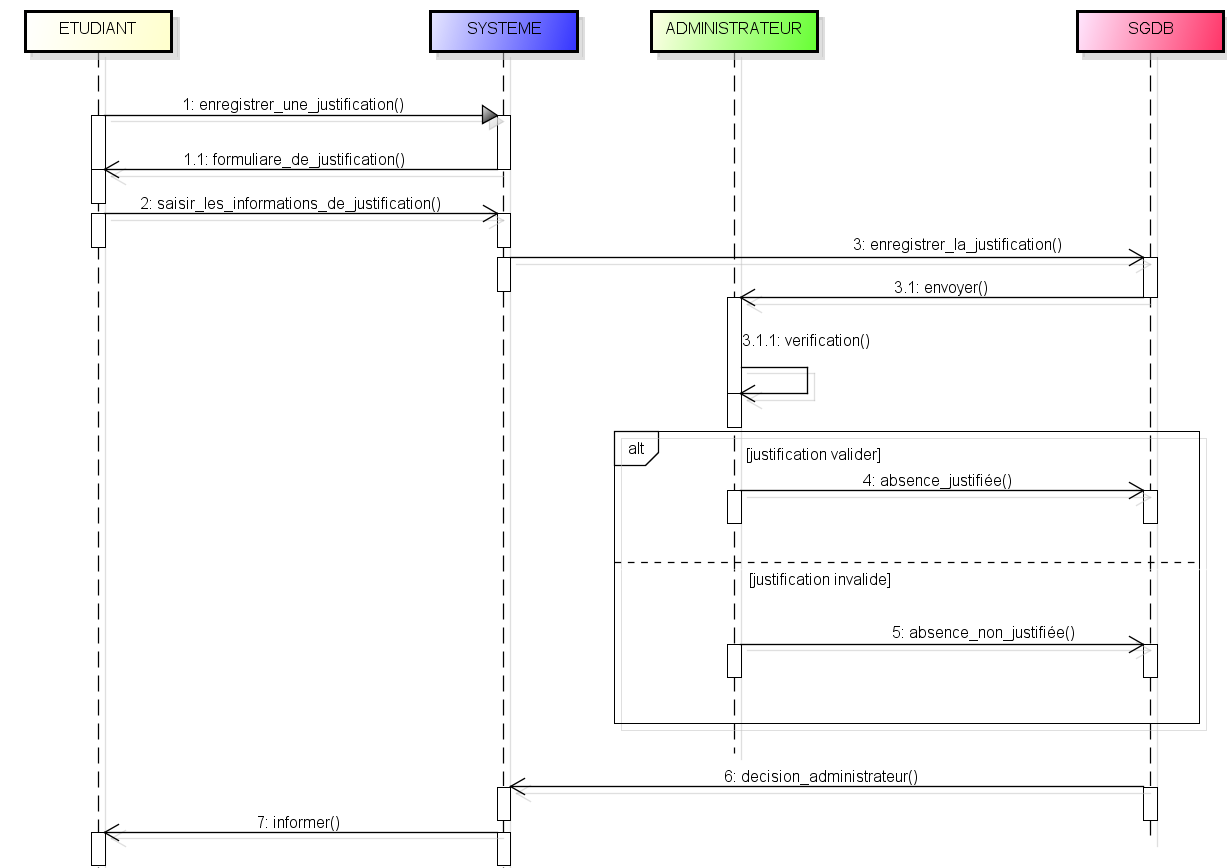


Figure 7 : Diagramme de séquence de justification

Ce processus est déclenché par l’étudiant. Après l’authentification, l’étudiant ouvre l’application et choisir d’enregistrer une justification, le système affiche un formulaire à remplir, il saisir les informations sur la justification et envoie au système qui l’enregistre dans la base de données. La base de données envoie les informations à l’administrateur qui vérifie si la justification est valide ou non. Si la justification est valide, l’absence est justifiée sinon l’absence n’est pas justifiée.

* + - 1. **Diagramme de séquence du processus de demande d’une autorisation d’absence**

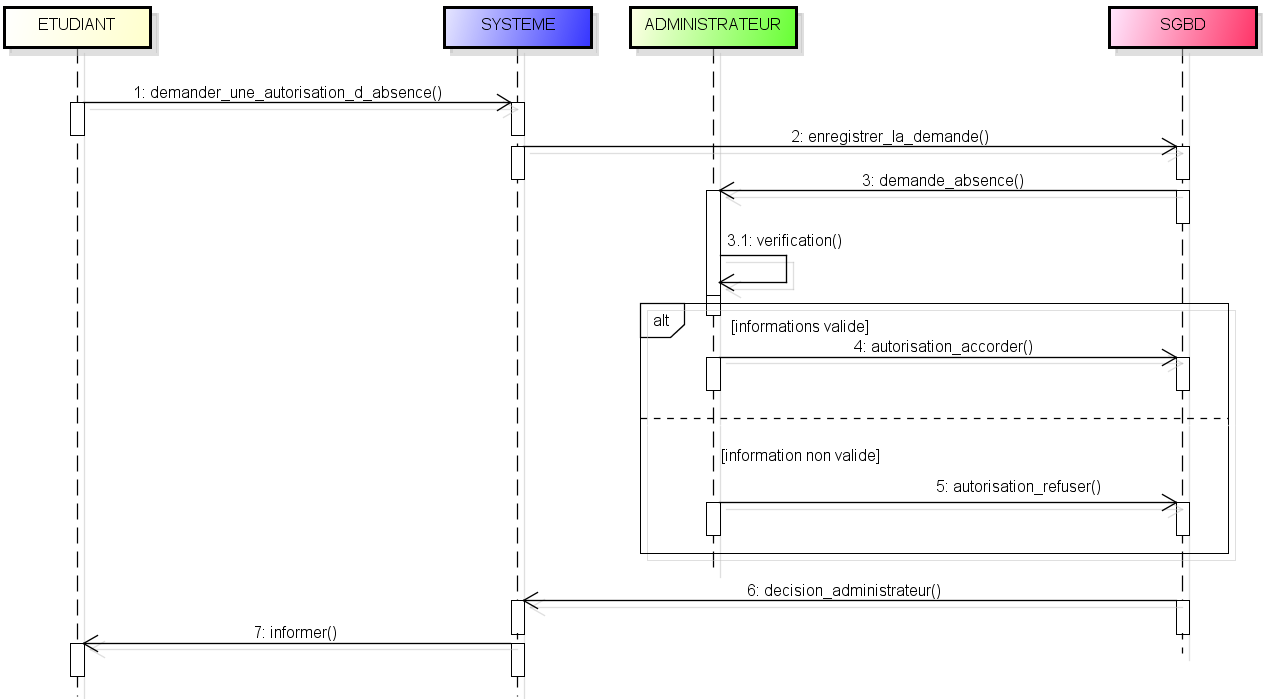


Figure 8 : Diagramme de séquence de demande d’absence

Ce processus est déclenché par l’étudiant. Après l’authentification, l’étudiant fait une demande d’autorisation d’absence, le système enregistre la demande dans la base de données. La demande est ensuite soumise à l’administrateur qui procède à des vérifications. Si les informations données par l’étudiant sont valides, l’administrateur autorise l’absence de l’étudiant sinon la demande est refusée.

### **ELABORATION DU DIAGRAMME DE CLASSE**

#### **Règle de gestion**

Les règles de gestions sont des déclarations qui aident à la création d’un modèle d’une base de données. Pour le système, nous avons relevé les règles de gestion suivantes :

RG 1 : une filière contient un ou plusieurs étudiants.

RG 2 : un étudiant est dans une et une seule filière pendant son parcours

RG 3 : Un étudiant est inscrit dans un ou plusieurs niveaux,

RG 4 : un administrateur est responsable d’un et un seul niveau.

RG 5 : un niveau a un et un seul responsable.

RG 6 : un étudiant peut être absent une ou plusieurs fois

RG7 : une absence est pour un et un seul étudiant.

RG 8 : un étudiant a un ou plusieurs parents

RG 9 : un parent a un ou plusieurs étudiants sous sa responsabilité.

RG 10 : une absence concerne un et un seul ECUE

RG 11 : dans l’ECUE on a un ou plusieurs absences

RG 12 : une justification concerne une et une seule absence.

RG 13 : à une absence, on a une et une seule justification.

RG 14 : dans un semestre, on a un ou plusieurs ECUE

RG 15 : un ECUE apparait dans un et un seul semestre

RG 16 : dans une année académique, on a un ou plusieurs semestres.

RG 17 : un semestre est pour une ou plusieurs années académiques.

RG 18 : un utilisateur a un et un seul rôle dans l’application.

RG 19 : Un rôle est attribué à un ou plusieurs utilisateurs

#### **Diagramme de classe**

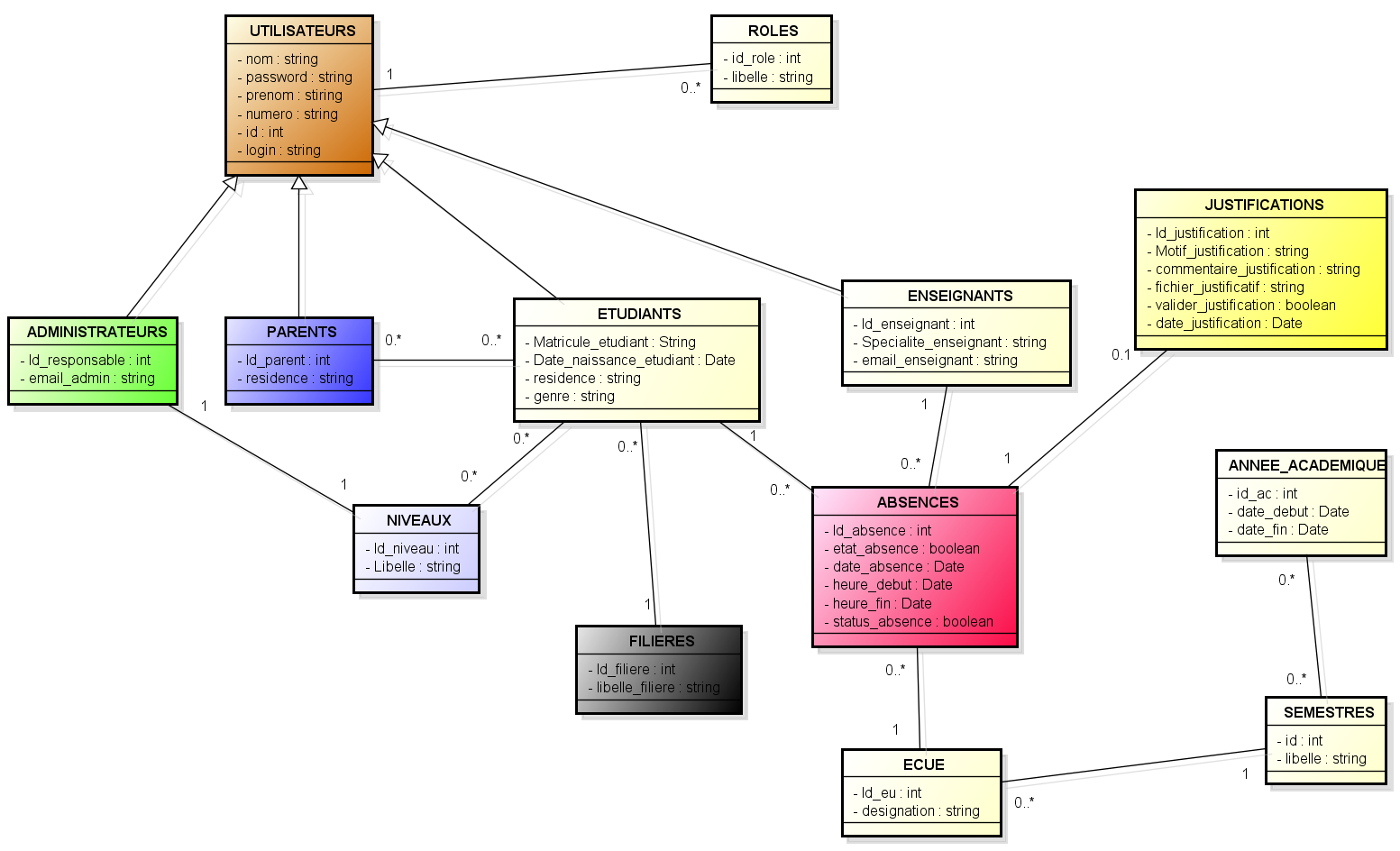


Figure 9 : Diagramme de classe

# **TROISIEME PARTIE : IMPLEMENTATION ET RESULTAT**

## **CHAPITRE I : NOTION D’API REST**

Dans ce chapitre, nous présenterons dans un premier temps les **API REST** et en second lieu l’implémentation de l’API REST.

### **PRESENTATION DES API REST**

#### **Définition d’ API**

**API est une abréviation et signifie Application Programming Interface : c’est un moyen de communication entre deux logiciels, que ce soit entre différents composants d’une application ou entre deux applications différentes.**

##### **Fonctionnement des API**

**Pour mieux expliquer le fonctionnement d’une api, nous allons voir ensemble les bases d’une communication serveur et client. Prenons l’exemple d’Air France, une compagnie aérienne française. Quelque part dans le monde, les serveurs d’Air France ont accès à toutes les données que vous voulez voir pour un trajet Paris-Abidjan : les différents avions, les tarifs, les statuts des vols, etc… Pour que vous puissiez y avoir accès, votre navigateur (que l’on appelle le client) doit recevoir ces informations de quelqu’un. Ce quelqu’un, c’est le serveur. L’application doit avoir une conversation avec le serveur.**

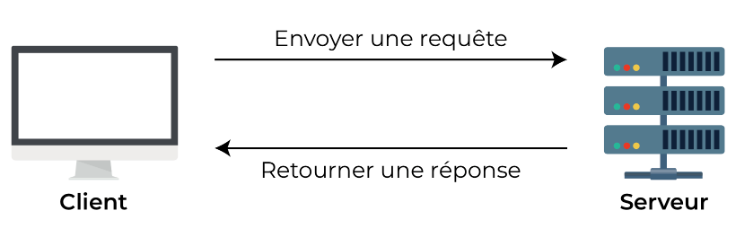


Figure 10 : Architecture client/serveur 1

**Cela ressemble à ça :**

* **Client : <<Salut serveur, est-ce que je pourrais avoir un avion pour Abidjan le 10 décembre ?>>**
* **Serveur : <<Voilà, tous les transports disponibles vers Abidjan le 10 décembre ! >>**

**Ou alors, si le serveur ne parvient pas à trouver les données, il pourrait répondre comme ceci :**

* **Serveur : <<Désolé, en fait il n’y a pas de vols disponibles le 10 décembre.>>**

**C’est ce qu’on appelle la communication entre client et serveur : le client formule une requête (ou une demande) pour obtenir une information et le serveur envoie une**

**Réponse contenant les données demandées si cela est possible. Du coup, l’API se place où dans ce schéma ? Et quel est le rapport avec Air France ?**

**Une API facilite l’interaction entre deux applications différentes afin qu’elles puissent communiquer entre elles : elle sert d’intermédiaire. Le client va demander à l’API une information, celle-ci va aller chercher cette information dans la base de données puis la renvoyer au client dans un second temps.**

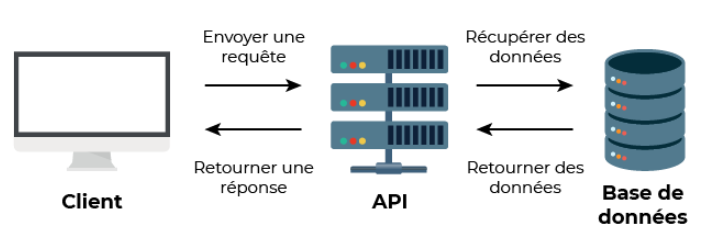


Figure 11 : Notion d’API dans l’architecture client/serveur

**Les API permettent la communication entre de nombreux composants différents de votre application, mais aussi entre des composants de votre application et d’autres développeurs. Elles agissent ainsi comme un intermédiaire qui transmet des messages à travers un système de requêtes et de réponses. Reprenons notre exemple avec Air France.**

**On crée une application de comparateur de vols que l’on va appeler MyVol. Celle-ci ne peut pas accéder directement aux informations d’Air France ou de toute autre compagnie aérienne. En effet, l’application n’a pas accès à leurs bases de données. Mais si Air France à une API à qui on peut demander des informations et qui partage certaines données de la base de données avec d’autre**

**Applications, alors MyVol peut demander des informations à l’API d’Air France. L’API lui renvoie alors des données que MyVol peut partager. Ainsi, MyVol peut comparer les prix entre les différentes compagnies qui ont mis en place un vol le 10 décembre pour Abidjan.**

**Les API peuvent communiquer :**

* **D’un logiciel à un autre ;**
* **D’un client à un serveur ;**
* **Ou d’un logiciel à des développeurs.**

#### **Les types d’API**

###### **Les API privées**

**Les API privées garantissent que les personnes en dehors de votre entreprise ou de votre application n’ont pas accès aux données de votre base de données. Par exemple, si les développeurs d’UPB voulaient construire une application interne pour que les responsables de la scolarité puissent gérer et analyser des informations d’inscription, il y aurait de nombreuses données que les responsables voudraient voir, crée, et modifier. Pour que les utilisateurs puissent interagir avec les données, les développeurs d’UPB pourraient créer une API pour que les responsables puissent accéder aux données d’inscription à travers leur application, sans pour autant donner ces accès aux utilisateurs de la plateforme comme vous et moi. Une API privée permet uniquement aux utilisateurs autorisés au sein de votre entreprise ou de votre application d’utiliser l’API qui peut accéder à la base de données.**

###### **Les API publiques**

**Contrairement aux API privées, les API que l’on appelle publiques sont utilisables par d’autres personnes, qu’elles soient sur votre application ou non. Elles permettent aux développeurs de récolter les données d’une autre application pour améliorer ou enrichir leurs propres projets sans autorisation stricte.**

#### **Présentation de REST**

**REST signifie Représentationnel State Transfer (ou transfert d’état de représentation, en français), et constitue un ensemble de normes, ou de lignes directrices architecturales qui structurent la façon de communiquer les données entre votre application et le reste du monde, ou entre différents composants de votre application. Il y a six lignes directrices architecturales clés pour les API REST.**

* **Client-serveur séparation : L’une des normes de REST est la séparation du client et du serveur. Un client est celui qui va utiliser l’API. Cela peut être une application, un navigateur ou un logiciel.**

**Un serveur est un ordinateur distant capable de récupérer des données depuis la base de données, de les manipuler si besoin et de les renvoyer à l’API, comme ce gros ordinateur au milieu :**

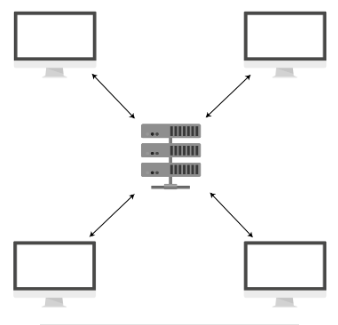


Figure 12 : Architecture client/serveur 2

* **Stateless : L’un des autres aspects uniques des API REST est qu’elles sont sans état, ce qui signifie que le serveur ne sauvegarde aucune des requêtes ou réponses précédentes.**
* **Cacheable (ou sauve gardable) : La réponse doit contenir l’information sur la capacité ou non du client de mettre les données en cache, ou de les sauvegarder. Si les données peuvent être mises en cache, la réponse doit être accompagnée d’un numéro de version. Ainsi, si votre utilisateur formule deux fois la même requête (c’est-à-dire veut revoir une page) et que les informations n’ont pas changé, alors votre serveur n’a pas besoin de rechercher les informations une deuxième fois. A la place, le client peut simplement mettre en cache les données la première fois, puis charger à nouveau les mêmes données la seconde fois.**
* **Uniforme Interface (interface uniforme) : Lors de la création d’une API REST, les développeurs acceptent d’utiliser les mêmes normes. Ainsi, chaque API a une interface uniforme. L’interface constitue un contrat entre le client et le service, que partagent toutes les API REST. C’est utile, car lorsque les développeurs utilisent des API, cela leur permet d’être sûrs qu’ils se comprennent entre eux.**
* **Layered system (système de couches) : Chaque composant qui utilise REST n’a pas accès aux composants au-delà du composant précis avec lequel il interagit.**
* **Code on demand (code à la demande) : Le code à la demande signifie que le serveur peut étendre sa fonctionnalité en envoyant le code au client pour téléchargement. C’est facultatif, car tous les clients ne seront pas capables de télécharger et d’exécuter le même code – donc ce n’est pas utilisé habituellement, mais au moins, nous savons que ça existe.**

**Les API REST se basent sur le protocole HTTP pour transférer les informations. Le même protocole sur lequel la communication web est fondée.**

#### **Présentation du protocole http**

##### **Le protocole HTTP**

* **HTTP : HyperTest Transfert Protocol**
* **Protocole qui permet au client de récupérer des documents du serveur**
* **Ces documents peuvent être statiques (contenu qui ne change pas :HTML, PDF, Image, etc…) ou dynamiques (contenu généré dynamiquement au moment de la requête : PHP, JSP, ASP, etc.)**
* **Ce protocole permet également de soumettre les formulaires**
* **Fonctionnement**
* **Le client se connecte au serveur (Créer un socket)**
* **Le client demande au serveur un document : Requête HTTP**
* **Le serveur renvoi au client le document (status=200) ou d’une erreur (status=404 quand le document n’existe pas)**
* **Déconnexion**

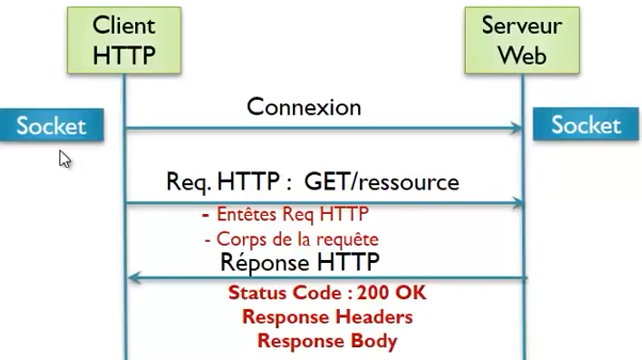


Figure 13 : Communication client/serveur

#### **Utilisation des méthodes HTTP dans le cas d’une API REST**

* **Une requête HTTP peut être envoyée en utilisant les méthodes suivantes :**
* **GET : Pour consulter des ressources**
* **POST : Pour soumettre et ajouter de nouvelles ressources**
* **PUT : Pour mettre à jour des ressources existantes**
* **DELETE : Pour supprimer des ressources existantes**
* **OPTIONS : Permet au client HTTP d’interroger le serveur pour fournir les options de communications à utiliser pour une ressource ciblée ou un ensemble de ressources (Méthodes autorisées, Entêtes autorisées, Origines autorisés, etc..). Utilisé souvent comme requête de pré-vérification Cross-Origine CORS (Cross Origin Resource Sharing).**
* **HEAD : Permet de consulter les métadonnées d’une ressource (Type, Capacité, Date de dernière modification etc …)**

##### **Les codes status**

* **Lorsque le serveur renvoie un document, il lui associe un code de status renseignant ainsi le client sur le résultat de la requête (requête invalide, document non trouvé …).**
* **Les principales valeurs des codes de status HTTP sont détaillées dans le tableau ci-après.**
* **Information 1xx :**
* **100 (contenu) : Utiliser dans le cas où la requête possède un corps.**
* **101(switching protocol) : Demander au client de changer de protocole. Par exemple de HTTP 1.0 vers HTTP 1.1**
* **Succès 2xx :**
* **200(ok) : Le document a été trouvé et son contenu suit**
* **201(created) : Le document a été créé en réponse à un PUT**
* **202(Accepted) : Requête acceptée, mais traitement non terminé**
* **204 (No response) : Le serveur n’a aucune information à renvoyer**
* **206 (Partial content) : Une partie du document suit**
* **Redirection 3xx :**
* **301 (Moved) : Le document a changé d’adresse de façon permanente**
* **302 (Found) : Le document a changé d’adresse temporairement**
* **304 (Not modified) Le document demandé n’a pas été modifié**
* **Erreurs du client 4xx :**
* **400 (Bad request) : La syntaxe de la requête est incorrecte**
* **401 (Unauthorized) : Le client n’a pas les privilèges d’accès au document**
* **403 (Forbidden) : L’accès au document est interdit**
* **404 (Not found) : Le document demandé n’a pu être trouvé**
* **405 (Method not allowed) : La méthode de la requête n’est pas autorisée**
* **Erreur du serveur 5xx :**
* **500 (Internal error) : Une erreur inattendue est survenue au niveau du serveur**
* **501 (Not implemented) : La méthode utilisée n’est pas implémentée**
* **502 (Bad gateway) : Erreur de serveur distant lors d’une requête proxy**

##### **Les entêtes HTTP**

* **Entêtes HTTP génériques :**
* **Content-length : Longueur en pctets des données suivant les en-têtes**
* **Content-type : Type MIME des données qui suivent**
* **Connection : Indique si la connexion TCP doit rester ouverte (Keep-Alive) ou être fermée(close)**
* **Entêtes de la requête :**
* **Accept : Type MIME que le client accepte**
* **Accept-encoding : Méthodes de compression supportées par le client**
* **Accept-language : Langues préférées par le client (pondérées)**
* **Cookie : Données de cookie mémorisées par le client**
* **Host : Hôte virtuel demandé**
* **If-modified-since : Ne retourne le document que si modifié depuis la date indiquée**
* **If-none-match : Ne retourne le document que s’il a changé**
* **Referer : URL de la page à partir de laquelle le document est demandé**
* **User-agent : Nom et version du logiciel client**

##### **Les entêtes de réponse HTTP**

* **Allowed : méthodes HTTP autorisées pour cette URL (comme POST)**
* **Content-encoding : méthode de compression des données qui suivent**
* **Content-language : langue dans laquelle le document retourné est écrit**
* **Date : date et heure UTC courante**
* **Expires : date à laquelle le document expire**
* **Last-modified : date de dernière modification du document**
* **Location : adresse du document lors d’une redirection**
* **Etag : numéro de version du document**
* **Pragma : données annexes pour le navigateur (par exemple, no.cache)**
* **Server : nom et version du logiciel serveur**
* **Set-cookie : permet au serveur d’écrire un cookie sur le disque du client**

1. **OUTILS D’IMPLEMENTATION**

Cette partie est dédié à la mise en œuvre de chacune des fonctionnalités décrites dans le chapitre de l’étude préliminaire.

Nous commençons par la description des environnements matériels et logiciels qui nous ont permis de réaliser notre projet ainsi que l'architecture physique de notre système à travers le diagramme de déploiement. Nous passons ensuite à la phase d'implémentation dans laquelle nous allons présenter les différentes techniques que nous avons utilisées pour réaliser notre **API REST.**

#### **Environnement de travail**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **NOM** | **NOMBRE DE POUCES** | **SYSTÈME HÔTE** | **MÉMOIRE RAM** | **CAPACITÉ DU DISQUE DUR** | **PROCESSEUR** |
| HP | 15 | Windows 10 | 4 Go | 500 Go | AMD, CORES 2C+2G |

Tableau 4 : Tableau résumant les caractéristiques de notre machine

#### **Environnement logiciel**

- Un environnement de développement : Eclipse

**Eclipse** est un projet, décliné et organisé en un ensemble de sous-projets de développements logiciels, de la [fondation Eclipse](https://fr.wikipedia.org/wiki/Fondation_Eclipse) visant à développer un environnement de production de logiciels [libre](https://fr.wikipedia.org/wiki/Logiciel_libre) qui soit extensible, universel et polyvalent, en s'appuyant principalement sur [Java](https://fr.wikipedia.org/wiki/Java_(technique)).

Son objectif est de produire et fournir des outils pour la réalisation de logiciels, englobant les activités de programmation (notamment [environnement de développement intégré](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9) et [Framework](https://fr.wikipedia.org/wiki/Framework)) mais aussi d'[AGL](https://fr.wikipedia.org/wiki/Atelier_de_g%C3%A9nie_logiciel) recouvrant [modélisation](https://fr.wikipedia.org/wiki/Mod%C3%A9lisation_des_donn%C3%A9es), [conception](https://fr.wikipedia.org/wiki/Conception_de_logiciel), [test](https://fr.wikipedia.org/wiki/Test_(informatique)), [gestion de configuration](https://fr.wikipedia.org/wiki/Gestion_de_configuration), [Reporting](https://fr.wikipedia.org/wiki/Reporting" \o "Reporting)… Son [EDI](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement), partie intégrante du projet, vise notamment à supporter tout [langage de programmation](https://fr.wikipedia.org/wiki/Langage_de_programmation) à l'instar de [Microsoft Visual Studio](https://fr.wikipedia.org/wiki/Microsoft_Visual_Studio). En fait Eclipse est à la fois considéré comme un [EDI](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9), un [Framework](https://fr.wikipedia.org/wiki/Framework) ou une [plateforme](https://fr.wikipedia.org/wiki/Plate-forme_(informatique)).

- Framework : Spring Framework

Spring est un **Framework**, c’est-à-dire un **cadre de travail existant** que les développeurs peuvent utiliser pour des projet JAVA EE. Il est open source et facilite le développement en JAVA EE.

- Serveur de base de données : MySQL

MySQL est un système de gestion de base de données relationnelle (SGBDR). Il fait partie des logiciels de gestion de base de données les plus utilisé au monde autant par le grand public (application web) que par des professionnels. Il est distribué sous licence GPL et propriétaire. MySQL est open source et gratuits appartenant à Oracle Corporation et fonctionnant sur des systèmes de type UNIX ou WINDOWS. Son architecture est de type client/serveur.

## **CHAPITRE II : IMPLEMENTTION**

Dans cette partie, il s'agit ici de vérifier si le système offre véritablement les services exigés par les utilisateurs, de détecter les défaillances, de combler les manques et d’adapter le produit à l’environnement de travail des utilisateurs.

### **RESULTATS**

* **Les ressources de l’API REST.**

L'API REST expose plusieurs ressources permettant aux développeurs de construire une application de gestion des absences des étudiants d’UPB.

##### **Ressource absence**

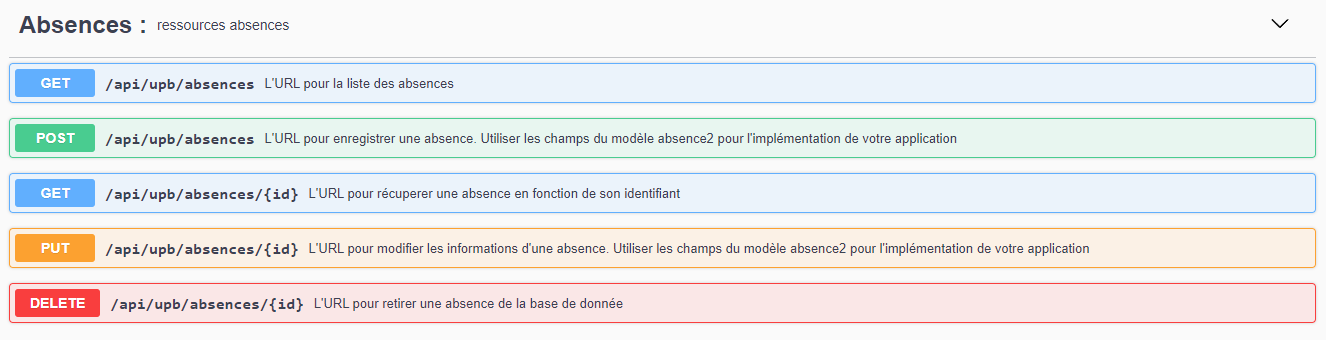


Figure 14 : Ressource absence

Cette ressource permet à un utilisateur authentifier d’enregistrer une absence, de sélectionner toutes les absences, de sélectionner une absence spécifique en fonction de son identifiant, de mettre à jour une absence et de supprimer une absence du système.

##### **Ressource administrateur**

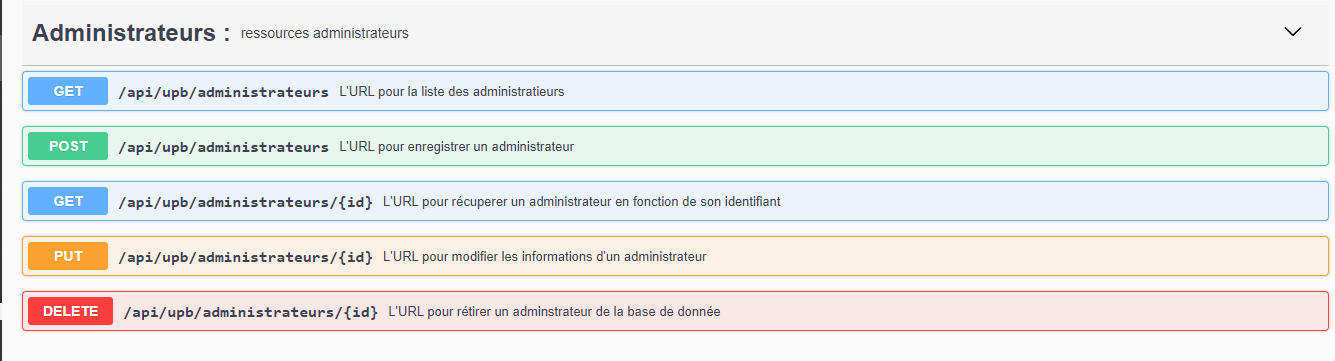


Figure 15 : Ressource administrateur

Cette ressource permet à un utilisateur authentifier d’enregistrer un administrateur, de sélectionner tous les administrateurs, de sélectionner un administrateur spécifique en fonction de son identifiant, de mettre à jour les informations d’un administrateur et de supprimer un administrateur du système.

##### **Ressource niveau**



Figure 16 : Ressource niveau

Cette ressource permet à un utilisateur authentifier d’enregistrer un niveau d’étude, de sélectionner tous les niveaux d’étude, de sélectionner un niveau spécifique en fonction de son identifiant, de mettre à jour un niveau et de retirer un niveau du système.

##### **d Ressource justification**

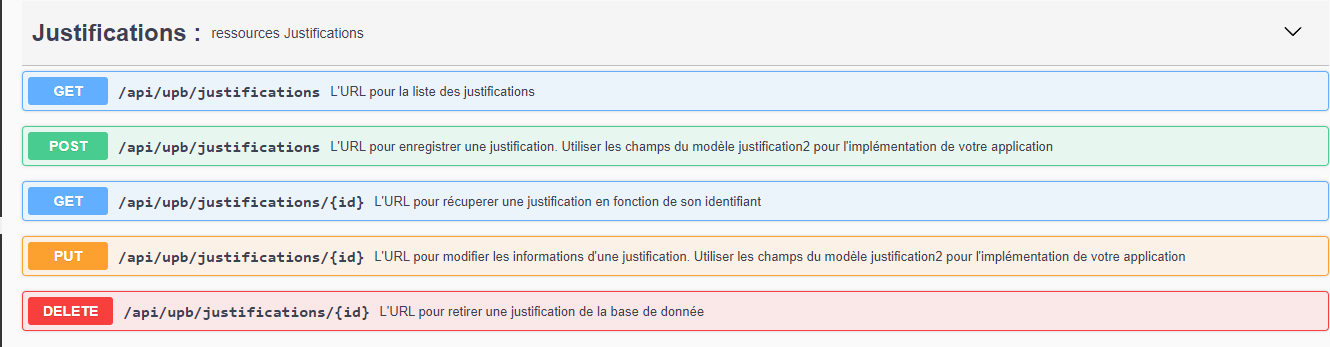


Figure 17 : Ressource justification

Cette ressource permet à un utilisateur authentifié d’enregistrer une justification concernant une absence, de sélectionner toutes les justifications concernant les absences, de sélectionner une justification spécifique en fonction de son identifiant, de mettre à jour une justification et de supprimer une justification du système.

##### **e Ressource filière**

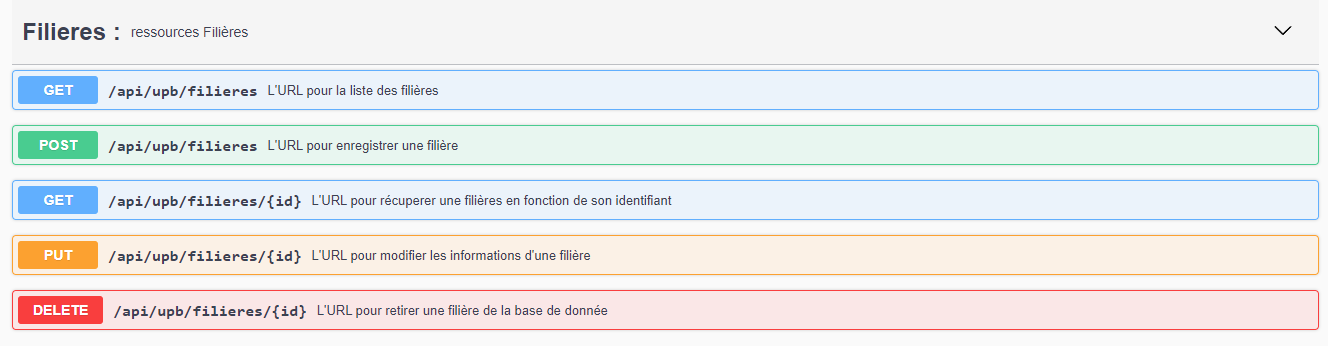


Figure 18 : Ressource filière

Cette ressource permet à un utilisateur authentifié d’enregistrer une filière, de sélectionner toutes les filières, de sélectionner une filière spécifique en fonction de son identifiant, de mettre à jour une filière et de supprimer une filière du système.

##### **f Ressource enseignant**

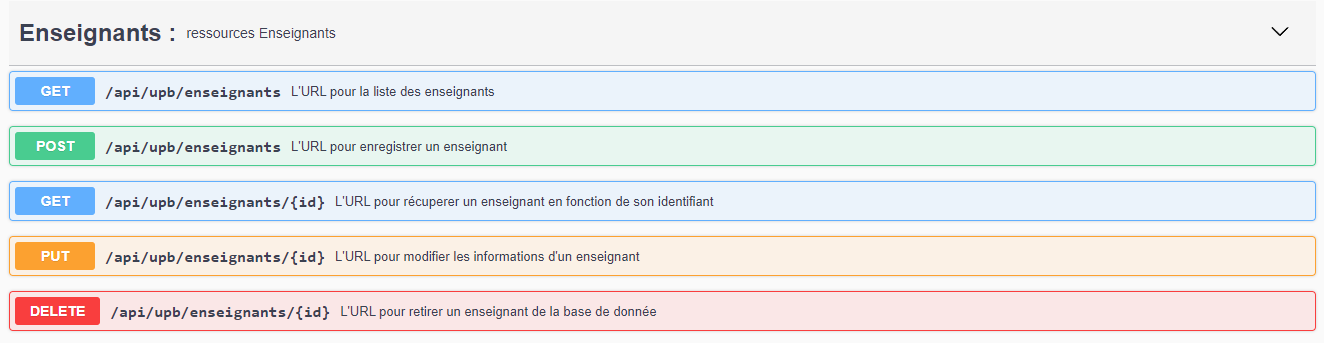


Figure 19 : Ressource enseignant

Cette ressource permet à un utilisateur authentifié d’enregistrer un enseignant, de sélectionner tous les enseignants, de sélectionner un enseignant spécifique en fonction de son identifiant, de mettre à jour les informations d’un enseignant et de supprimer un enseignant du système.

##### **g Ressource parent**

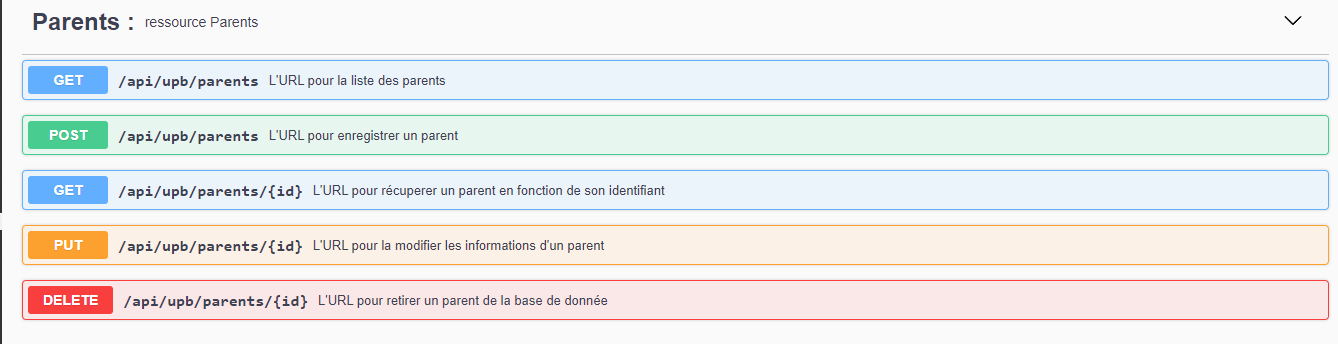


Figure 20 : Ressource parent

Cette ressource permet à un utilisateur authentifié d’enregistrer un parent responsable d’un étudiant, de sélectionner tous les parents, de sélectionner un parent spécifique en fonction de son identifiant, de mettre à jour les informations sur un parent et de retirer un parent du système.

##### **h Ressource étudiant**

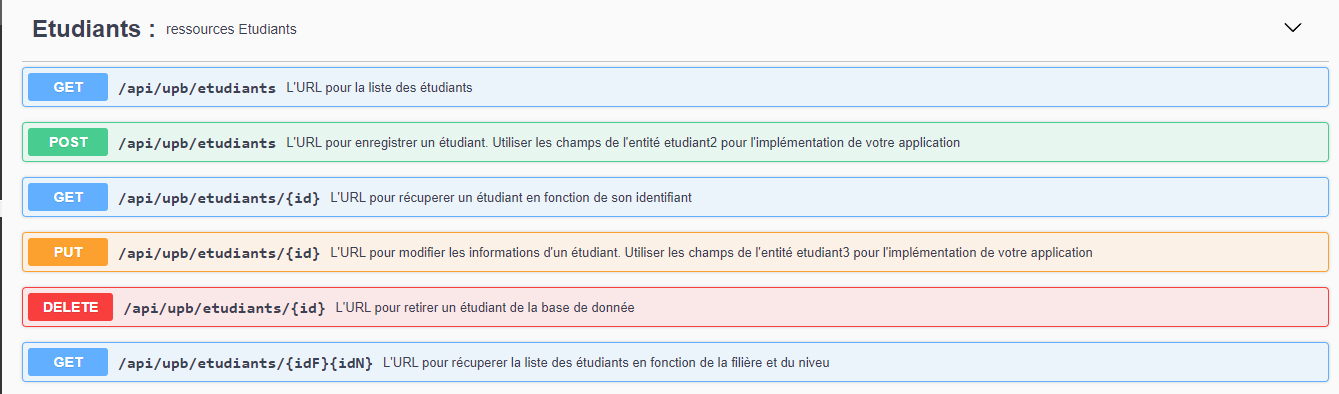


Figure 21 : Ressource étudiant

Cette ressource permet à un utilisateur authentifié d’enregistrer un étudiant dans le système, de sélectionner tous les étudiants, de sélectionner un étudiant spécifique en fonction de son identifiant, de mettre à jour les informations sur un étudiant et de retirer un étudiant du système.

##### **i Ressource utilisateur**

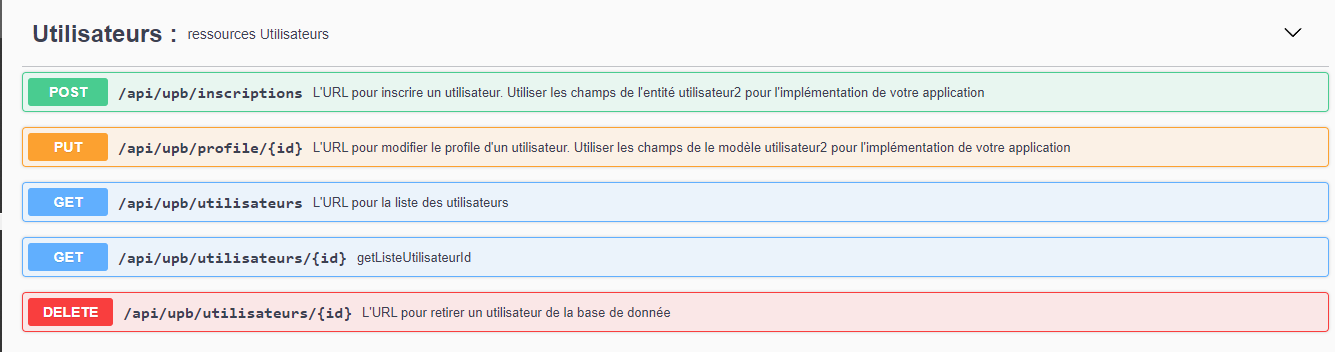


Figure 22 : Ressource utilisateur

Cette ressource permet d’enregistrer un utilisateur dans le système, de sélectionner tous les utilisateurs, de sélectionner un utilisateur spécifique en fonction de son identifiant, de mettre à jour les informations sur un utilisateur et de retirer un utilisateur du système.

### **COÛT DU PROJET**

1. **Plateforme d’hébergement du projet**

Pour la mise en production du projet, nous avons utilisé la plateforme **Heroku**. **Heroku est** une entreprise créant des logiciels pour serveur qui permettent le déploiement d’application web. Créé en 2007, elle est rachetée en 2010 par l’éditeur Salesforce.com. Le logiciel **Heroku** supporte les bases de données relationnelles et NoSQL. Il possède une architecture modulaire qui permet de supporter plusieurs environnements d’exécution grâce à la virtualisation logicielle par des containers nommés dynos. Heroku facture en fonction de l'utilisation. Chaque mois, vous recevez la facture du mois d'utilisation précédent, pas celui en cours. Le prix par dyno/mois indiqué dans le tableau est le **maximum** qui vous est facturé si vous utilisez le dyno 24 heures sur 24, 7 jours sur 7 depuis votre dernier cycle de facturation.

|  |  |
| --- | --- |
| **Type de dynamomètre** | **Prix ​​par dyno/mois** |
| Libre | 0 $ |
| Loisir | 7 $ |
| Standard-1x | 25 $ |
| Standard-2x | 50 $ |
| Performance-m | 250 $ |
| Performance-l | 500 $ |

Tableau 5 : Tableau récapitulatif des différentes modalités de facturation sur la plateforme Heroku

# **CONCLUSION**

Au cours de ce mémoire, nous avons présenté les différentes étapes de la conception et la réalisation de notre API REST pour la gestion des absences des étudiants d’UPB. Afin de satisfaire les besoins des utilisateurs nous avons commencé la conception en utilisant le langage de modélisation UML et la mise en œuvre des bases de données avec le système de gestion de bases de données MySQL ensuite l'implémentation des requêtes SQL pour la manipulation des données et enfin la concrétisation de l’API REST sous l'environnement de programmation avec le Framework Spring Boot.

Ce projet a fait l'objet d'une expérience intéressante, qui nous a permis d'améliorer nos connaissances et nos compétences dans le domaine du génie logiciel.

Cependant, des perspectives d'améliorations de notre API REST restent envisageables telles que la programmation ou l’organisation des cours, l’établissement des emplois du temps des filières, la programmation des activités universitaire, l’établissement des horaires des cours ou des examens.

# **BIBLIOGRAPHIE**

**JEAN-PIERRE** **FRAGNIÈRE**, Comment réussir un mémoire, Edition 5, Dunod, 2016, 142P

**JOHN WILEY & SONS, INC,** Les API pour les nuls, Edition limitée IBM, 111 River St. Hoboken, NJ 07030-5774

# **WEBOGRAPHIE**

**Présentation de l’Université Polytechnique de Bingerville [En ligne] visité le : 04/09/2022**

<https://upb.ci/en/formations>

**Approche UML [En ligne] visité le : 04/09/2022**

<http://www.iro.umontreal.ca/~dift6825/UML.htm>

**Documentation sur les API REST [En ligne] visité le : 04/08/2022**

[**https://openclassrooms.com/fr/courses/6573181-adoptez-les-api-rest-pour-vos-projets-web/6817216-identifiez-les-avantages-d-une-api-rest**](https://openclassrooms.com/fr/courses/6573181-adoptez-les-api-rest-pour-vos-projets-web/6817216-identifiez-les-avantages-d-une-api-rest)

[**https://aws.amazon.com/fr/what-is/restful-api/**](https://aws.amazon.com/fr/what-is/restful-api/)

[**https://www.ibm.com/fr-fr/cloud/learn/rest-apis**](https://www.ibm.com/fr-fr/cloud/learn/rest-apis)

**Formations sur le Framework Spring Boot [En ligne] visité le : 04/08/2022**

<https://spring.io/projects/spring-boot>

**Merise [En ligne] visité le 05/08/2022** <https://www.memoireonline.com/02/10/3158/m_Conception-et-realisation-dune-application-de-webmapping-danalyse-territoriale-sur-des-SIG-et-bas31.html>

**Table des matières**

[**Dédicace** III](#_Toc116516978)

[**Remerciement** IV](#_Toc116516979)

[**Avant-propos** V](#_Toc116516980)

[**Sommaire** VI](#_Toc116516981)

[**Sigles et abréviations** VII](#_Toc116516982)

[**Liste des Tableaux** VIII](#_Toc116516983)

[**Introduction** 1](#_Toc116516984)

[**PREMIERE PARTIE : GENERALITE** 2](#_Toc116516985)

[**CHAPITRE I : CONTEXT DE L’ETUDE** 3](#_Toc116516986)

[**I.** **PRESENTATION D’UPB** 3](#_Toc116516987)

[**1.** **Présentation** 3](#_Toc116516988)

[**2.** **Organisation de l’UPB** 3](#_Toc116516989)

[**a** **Les principales Directions et instances** 3](#_Toc116516990)

[**b** **Organigramme de l’UPB** 4](#_Toc116516991)

[**3.** **Enseignements** 5](#_Toc116516992)

[**a** **Structure des études** 5](#_Toc116516993)

[ **Schéma général du cursus universitaire à l’UPB** 5](#_Toc116516994)

[ **Organisation des enseignements** 5](#_Toc116516995)

[ **Programme des études** 5](#_Toc116516996)

[**b** **Durée et crédits pour chaque cycle** 6](#_Toc116516997)

[**4.** **Inscription pédagogique** 6](#_Toc116516998)

[**5.** **Nos filières** 7](#_Toc116516999)

[**II.** **CAHIER DES CHARGES** 7](#_Toc116517000)

[**1.** **Contexte du projet** 7](#_Toc116517001)

[**2.** **Objectif général** 8](#_Toc116517002)

[**3.** **Objectifs spécifiques** 8](#_Toc116517003)

[**4.** **Étude de l’existant** 8](#_Toc116517004)

[**a** **Analyse de l’existant** 8](#_Toc116517005)

[**b** **Description du système actuel de gestion des absences.** 8](#_Toc116517006)

[**c** **Critique de l’existant** 9](#_Toc116517007)

[**CHAPITRE II : MÉTHODES DE DÉVELOPPEMENT** 10](#_Toc116517008)

[**I.** **PRESENTATION DES METHODES** 10](#_Toc116517009)

[**1.** **Présentation de UML** 10](#_Toc116517010)

[**a** **Définition de UML** 10](#_Toc116517011)

[**b** **Les vues en UML** 11](#_Toc116517012)

[ **Présentation des vues** 11](#_Toc116517013)

[ **Les différents types de vues** 11](#_Toc116517014)

[ **Les diagrammes en UML** 11](#_Toc116517015)

[ **Les diagrammes statiques** 11](#_Toc116517016)

[ **Les diagrammes dynamiques** 12](#_Toc116517017)

[**2.** **Présentation de MERISE** 13](#_Toc116517018)

[**a** **Définition de Merise** 13](#_Toc116517019)

[**b** **Les différents types de démarche en Merise** 13](#_Toc116517020)

[ **La démarche classique** 13](#_Toc116517021)

[ **La démarche rapide** 14](#_Toc116517022)

[**c** **Les différents modèles en Merise** 14](#_Toc116517023)

[**II.** **CHOIX DES METODES** 14](#_Toc116517024)

[**1.** **Etude comparative des méthodes** 14](#_Toc116517025)

[**1.** **Méthode choisie** 15](#_Toc116517026)

[**DEUXIEME PARTIE : ETUDE CONCEPTUELLE** 17](#_Toc116517027)

[**CHAPITRE I : ETUDE PRELIMINAIRE** 18](#_Toc116517028)

[I. **ANALYSE ET CONCEPTION.** 18](#_Toc116517029)

[**1.** **Spécification des besoins** 18](#_Toc116517030)

[**2.** **Besoins fonctionnels** 18](#_Toc116517031)

[**3.** **Besoins non fonctionnels** 19](#_Toc116517032)

[**II.** **DÉLIMITATION DU SYSTÈME** 19](#_Toc116517033)

[**1.** **Identification des cas d’utilisation.** 19](#_Toc116517034)

[**CHAPITRE II : CAPTURE DES BESOINS FONCTIONNELS** 21](#_Toc116517035)

[**I.** **DIAGRAMME DES CAS D’UTILISATION** 21](#_Toc116517036)

[**II.** **DESCRIPTION TEXTUELLE.** 22](#_Toc116517037)

[**CHAPITRE III : ANALYSE DU SYSTÈME** 31](#_Toc116517040)

[**I.** **ELABORATION DES DIAGRAMMES DE SÉQUENCES** 31](#_Toc116517041)

[**1.** **Diagramme de séquence du processus d’authentification** 31](#_Toc116517042)

[**II.** **ELABORATION DU DIAGRAMME DE CLASSE** 35](#_Toc116517043)

[**1.** **Règle de gestion** 35](#_Toc116517044)

[**2.** **Diagramme de class** 36](#_Toc116517045)

[**TROISIEME PARTIE : IMPLEMENTATION ET RESULTAT** 38](#_Toc116517046)

[**CHAPITRE I : NOTION D’API REST** 39](#_Toc116517047)

[**I.** **PRESENTATION DES API REST** 39](#_Toc116517048)

[**1.** **Présentation des API** 39](#_Toc116517049)

[**a** **Fonctionnement des API** 39](#_Toc116517050)

[**b** **Les types d’API** 41](#_Toc116517051)

[ **Les API privées** 41](#_Toc116517052)

[ **Les API publiques** 41](#_Toc116517053)

[**2.** **Présentation de REST** 42](#_Toc116517054)

[**3.** **Présentation du protocole http** 43](#_Toc116517055)

[**a** **Le protocole HTTP** 43](#_Toc116517056)

[**b** **Utilisation des méthodes HTTP dans le cas d’une API REST** 44](#_Toc116517057)

[**c** **Les codes status** 45](#_Toc116517058)

[**d** **Les entêtes HTTP** 46](#_Toc116517059)

[**e** **Les entêtes de réponse HTTP** 46](#_Toc116517060)

[**1.** **Environnement de travail** 47](#_Toc116517061)

[**2.** **Environnement logiciel** 47](#_Toc116517062)

[**CHAPITRE II : IMPLEMENTATION** 49](#_Toc116517063)

[I. **RESULTAT** 49](#_Toc116517064)

[**a** **Ressource absence** 49](#_Toc116517065)

[**b** **Ressource administrateur** 50](#_Toc116517066)

[**c** **Ressource niveau** 50](#_Toc116517067)

[**d** **Ressource filière** 51](#_Toc116517068)

[**e** **Ressource des enseignants** 52](#_Toc116517069)

f [**Ressource parent** 52](#_Toc116517070)

[**g** **Ressource des étudiants** 53](#_Toc116517071)

[**h** **Ressource utilisateur** 53](#_Toc116517072)

[II **COUT DU PROJET** 54](#_Toc116517073)

[**CONCLUSION** 56](#_Toc116517074)

[**BIBLIOGRAPHIE** X](#_Toc116517075)

[**WEBOGRAPHIE** XI](#_Toc116517076)

**TABLE DES MATIERES** [XII](#_Toc116517076)

**RESUME**

L’informatiqueestune science qui étudie les technologies de traitement automatique de l'information. Elle est devenue l’outil principal du développement social et économique. Vu l’importance qu’elle joue dans une organisation, il est donc nécessaire de consacrer des investissements importants et une attention considérable.

C’est également ce constat qui nous a amenés à concevoir et réaliser une **API REST** de gestion des absences des étudiants pour l’Université Polytechnique de Bingerville afin d’optimiser la gestion des absences de ses étudiants.

Pour ce faire, nous avons présenté d’abord l’Université Polytechnique de Bingerville et identifié les besoins de l’Université. Nous sommes passé ensuite à l’analyse et la conception du projet en utilisant le langage de modélisation unifié UML. Et finalement, pour l’implémentation de ce projet, nous avons utilisé les outils et les langages du développement des **API REST**.

Mots clés : UML, JAVA, MySQL, API REST, Gestion d’absence, Spring Boot.

**ABSTRACT**

Computer science is a science that studies automatic information processing technologies. It has become the main tool for social and economic development. Given the importance it plays in an organization, it is therefore necessary to devote significant investments and considerable attention. It is also this observation that leads us to design and create a REST API for managing student absences for the Polytechnic University of Bingerville in order to optimize the management of is students' absences.

To do this, we first presented the Polytechnic University of Bingerville and identified the needs of the University. We then moved on to the analysis and design of the project using the unified modeling language UML. And finally, for the implementation of this project, we used the tools and languages of REST API development.

Keywords: UML, JAVA, MySQL, REST API, absence management, Spring Boot