



Rapport projet tuteuré

MARC Aurélien

JUSSEAUME Jonathan

KADILAR Michel

SOUPLET Edvans

BERRADA Slimane

2020-2021

IUT de Vélizy, DUT INFA2



IUT de Vélizy-Rambouillet
CAMPUS DE VÉLIZY-VILLACOUBLAY

Sommaire

| | | |
|-------|--|----|
| I. | Méthodologie (philosophie) | 3 |
| II. | Cahier des charges..... | 3 |
| A. | Objectif global..... | 3 |
| B. | Contexte..... | 4 |
| C. | Projet général..... | 4 |
| D. | Exigences fonctionnelles | 4 |
| E. | Exigences non fonctionnelles | 5 |
| F. | Choix techniques des technologies..... | 6 |
| G. | Logiciels utilisés pour la réalisation du projet..... | 6 |
| III. | Analyse des besoins..... | 6 |
| A. | Analyse du projet..... | 6 |
| B. | Cycle de vie du projet..... | 8 |
| C. | Diagramme de cas d'utilisation de l'application web | 10 |
| D. | Diagramme de cas d'utilisation de l'application mobile | 12 |
| IV. | Conception..... | 14 |
| A. | Introduction | 14 |
| B. | Conception très haut niveau du système | 14 |
| C. | La base de données..... | 15 |
| D. | Structure du système | 22 |
| E. | Spécification détaillé des interfaces | 35 |
| V. | Planification..... | 47 |
| A. | Tâches du projet et estimation du temps en jour.personne..... | 47 |
| B. | Elaboration d'une planification | 50 |
| VI. | Analyse de risques..... | 54 |
| A. | Identification des risques..... | 54 |
| B. | Echelle de probabilité d'occurrence d'un risque | 55 |
| C. | Echelle de l'impact d'un risque sur le projet..... | 55 |
| D. | Matrice de l'impact du risque sur le projet en fonction de la probabilité du risque | 55 |
| E. | Tableau associant un risque à sa probabilité | 56 |
| F. | Tableau associant un risque à une stratégie de réponse et un point de situation de notre projet..... | 58 |
| VII. | Plan de communication | 62 |
| VIII. | Annexes..... | 65 |
| A. | Vocabulaire | 65 |

I. Méthodologie (philosophie)

Afin de réaliser ce projet de la manière la plus efficace possible nous avons décidé sous l'impulsion du chef de projet Aurélien MARC de travailler en méthodologie Agile. Par conséquent, tout notre travail sur le projet au long de l'année sera subdivisé en plusieurs « sprints » de 1 à 2 semaines où nous définirons différents axes de travail. Chaque sprint contiendra au minimum une semaine où nous sommes présents à l'IUT (distanciel compris). Lors de ces semaines, nous avons organisé des *daily scrum meetings* (voir Vocabulaire) qui nous permettent à chaque fois de définir pour chacun des membres du groupe les *User stories* (voir Vocabulaire) sur lesquels il a travaillé la veille et ceux sur lesquels il travaillera aujourd'hui. Chaque membre peut également parler des difficultés qu'il a rencontré dans son travail et les différents empêchements qu'il a pu avoir la veille notamment à cause des contrôles par exemple. A chaque fin de sprint, une réunion plus importante est organisée afin de faire une rétrospective de ce sprint et définir les axes de travail du prochain sprint. Chaque membre possède alors l'occasion d'exprimer son ressenti sur l'organisation et sa position au sein de l'équipe.

A partir de notre analyse du sujet dont nous parlerons dans la prochaine section, nous avons identifié 3 grandes parties dans ce projet tutoré : une application mobile, une application web et un serveur. Afin de s'organiser nous avons désigné un responsable pour chacun des ces aspects selon les spécialités de chacun. Par exemple, Slimane qui a déjà travaillé sur des applications Android est devenu le responsable de l'application mobile. Un responsable a pour rôle de veiller à la propreté du code c'est-à-dire au respect des meilleures pratiques, il aide aussi ses camarades moins aguerris. Nous avons pour optique que chacun investisse du temps dans les trois grands aspects du projet tuteuré afin que chacun augmente son bagage technique.

Pour résumer, notre philosophie est avant tout de travailler en méthode agile avec la philosophie scrum et faire en sorte que chacun progresse techniquement grâce à ce projet en ayant une forte entraide entre camarades.

II. Cahier des charges

A. Objectif global

À partir du sujet, on peut dire que l'objectif de ce projet consiste à réaliser une application web ainsi qu'une application mobile faite à partir d'Android qui vont nous permettre de faire le relevé des présences (pour l'application mobile) puis ensuite de générer les différentes feuilles de présence de la classe de FA de DUT INFO à l'IUT de Vélizy. Afin de pousser ce projet plus loin, nous avons décidé que ce projet s'étendrait à d'autres formations de l'IUT de Vélizy. Ainsi, notre application sera totalement prête à gérer le fait qu'avec le futur BUT informatique il y aurait plusieurs classes d'alternants dans le département INFO. Nous avons pour objectif que ce projet soit réellement utilisé par l'IUT de Vélizy afin de simplifier sa vie.

B. Contexte

Aujourd'hui, trop de papier est consommé pour les feuilles de présence au DUT informatique de Vélizy, le but serait donc de drastiquement réduire cette consommation papetière tout en ayant un substitut ergonomique et efficace qui permette également de réduire les traitements humains et donc les délais d'attente éventuels, ce qui permettrait de fluidifier les procédures d'enregistrement des présences, des retards, et des absences. De plus, avec la COVID-19, il n'est plus réellement recommandé de laisser chaque élève toucher une même feuille pour la signer. Une numérisation de ce processus nous permettrait de mieux respecter les gestes barrières.

C. Projet général

Pour réaliser ce projet, il faudra donc créer une application web, une application web ainsi qu'un serveur web qui sera lié à une base de données et qui recevra et enverra les données aux deux applications.

D. Exigences fonctionnelles

1. Droits d'accès

Quatre niveaux de droits d'accès pour quatre types d'acteurs différents :

- Etudiant : n'a accès ni à l'application mobile, ni à l'application web mais devra rentrer sa signature dans l'application mobile
- Professeur : accès uniquement à l'application mobile afin d'émarger les cours
- Secrétaire : accès uniquement à l'application web pour modifier des informations, générer le relevé des présences
- Professeur Responsable : a les droits du professeur et de la secrétaire

2. Application mobile

- Les données saisies par le professeur à chaque cours sont envoyées sur le serveur puis sont stockées en base de données
- Un étudiant doit pouvoir rentrer sa signature dans l'application mobile
- Un étudiant doit pouvoir effacer sa signature s'il l'a mal réalisée
- Un professeur doit pouvoir se connecter avec son adresse mail et son mot de passe
- Un professeur doit pouvoir changer son mot de passe
- Un professeur doit pouvoir changer son adresse mail
- Un professeur doit pouvoir rentrer sa signature
- Un professeur doit pouvoir changer sa signature
- Un professeur doit pouvoir émarger un cours
- Un professeur doit pouvoir envoyer un cours au serveur web
- Un professeur doit pouvoir choisir l'heure de début et l'heure de fin de son cours
- Un professeur doit pouvoir sélectionner une matière
- Un professeur doit pouvoir choisir la formation (parmi celle dans lesquelles il enseigne) afin d'émarger un cours
- Un professeur doit pouvoir mettre un élève présent, en retard ou absent

- Tous les élèves doivent avoir signés pour qu'on puisse émarger un cours

3. Application web

- Un professeur responsable ou une secrétaire doit pouvoir se connecter
- L'utilisateur doit télécharger une fiche de présence pour une classe donnée à un jour donné
- L'utilisateur doit pouvoir modifier les horaires d'un cours
- L'utilisateur doit pouvoir supprimer un cours
- L'utilisateur doit pouvoir modifier la matière d'un cours parmi les matières possibles pour la classe
- L'utilisateur doit pouvoir modifier le professeur d'un cours parmi les professeurs possibles pour la classe
- L'utilisateur doit pouvoir modifier le statut d'un élève (présent, retard, absent) durant un cours
- L'utilisateur doit pouvoir consulter les informations d'une journée de cours
- L'utilisateur doit pouvoir créer une classe grâce à un fichier CSV
- L'utilisateur doit pouvoir créer un élève et l'ajouter à une classe
- L'utilisateur doit pouvoir supprimer un élève
- L'utilisateur doit pouvoir ajouter un professeur à sa formation
- L'utilisateur ne doit avoir des informations concernant uniquement la formation dont il est le responsable ou la secrétaire

4. Serveur web

- Le serveur web doit pouvoir générer une fiche de présence pour une classe donnée à un jour donné
- Le serveur web doit pouvoir valider ou non la tentative de connexion d'une personne à l'application mobile ou web
- Le serveur web doit pouvoir renvoyer les informations stockées en base de données concernant les élèves, les matières, les professeurs, les formations, les comptes.
- Le serveur web doit pouvoir envoyer un mail pour permettre à un utilisateur de changer son mot de passe lorsqu'il l'a oublié
- Le serveur web doit pouvoir ajouter des informations à la base de données

E. Exigences non fonctionnelles

- Un professeur peut changer la langue de son application mobile parmi plusieurs choix de langue : anglais, espagnol, français, roumain etc.
- Les applications doivent être aux couleurs du logo de l'université de Versailles Saint-Quentin : du bleu, du vert et du bordeaux.
- L'application doit avoir un logo original.
- La communication entre les applications et le serveur web doit être rapide en renvoyant le minimum de données possibles.

F. Choix techniques des technologies

Serveur web/ API : Java Spring Boot

Il s'agit d'un framework java qui va nous permettre de déployer une API Rest. C'est un framework assez récent qui est très simple d'utilisation et dont Jonathan a une certaine maîtrise grâce à son expérience en entreprise. De plus, ce framework nous permet d'utiliser Spring Data JPA, un ORM qui nous permettra de faire nos requêtes à la base de données de manière plus simple.

Application web : Angular

Nous allons utiliser la technologie de Google Angular qui nous permettra de faire un client qui pourra répondre à nos besoins. De plus, la synergie entre Angular et Java Spring Boot est très bonne comme nous pourrons le voir dans la conception.

Base de données : MariaDB

Pour la base de données nous avons décidé d'utiliser MariaDB car c'est l'un des Systèmes de Gestion de Bases de données que l'on voit lors de nos cours à l'IUT, il nous semblait donc logique de le choisir.

Application mobile : Android

Bien évidemment, nous avons choisi Android car c'est ce qui est préconisé par le sujet de ce projet tuteuré. De plus, la programmation d'application mobile Android est étudiée lors du semestre 4 ce qui nous permettra de réaliser cette application grâce à ces nouvelles connaissances.

G. Logiciels utilisés pour la réalisation du projet

- **Android Studio** : développement application Android
- **JIRA** : gestion du projet
- **Confluence** : création de la documentation du projet ainsi que celle du projet
- **Bitbucket** : gestion de versions du code du projet
- **Webstorm** : programmation de l'application web
- **Intellij IDEA** : programmation du serveur Spring Boot
- **DataGrip** : gestion de la base de données

III. Analyse des besoins

A. Analyse du projet

Tout d'abord, qu'est-ce qu'un besoin ?

Un besoin peut se définir comme étant une exigence de la part d'un des acteurs qui va recevoir le produit ou le service. C'est en quelque sorte les finalités que doit absolument respecter le produit ou le service.

Ce projet a pour utilisateurs tous les professeurs de la formation par alternance du département informatique de l'Institut Universitaire de Technologie de Vélizy-Villacoublay (78), ainsi que le secrétariat de ce département informatique. Le CFA Sup2000 est également indirectement concerné par ce projet, puisqu'il va recevoir des

feuilles d'émargement directement créées de manière numérique, ce qui permet de réduire les délais, de faciliter l'interaction et les modifications, ainsi que d'économiser du papier, comme énoncé dans le contexte du cahier des charges. Concernant les clients, il s'agit de deux professeurs : Monsieur Barreau, professeur responsable de la formation par apprentissage, et Monsieur Huguin, professeur de réseau et de mathématiques, principal interlocuteur technique et rédacteur du sujet de ce projet tutoré.

Ce projet n'a pour objectif de n'être utilisé qu'au sein de l'IUT de Vélizy, et plus précisément, dans un premier temps tout du moins, au sein de la formation par alternance du département informatique, bien que notre groupe ait déjà travaillé sur des possibles évolutions de ce projet à l'échelle des autres formations du département informatique, voire de tout l'IUT.

Acteurs principaux :

- Les élèves, qui n'auront en réalité qu'un seul rôle à jouer, signer une seule et unique fois sur l'application Android (mobile) au début de l'année ou dès leur arrivée à l'IUT (dès leur rajout dans la liste des élèves d'une classe), cela va permettre d'automatiser la présence des signatures dans la feuille d'émargement qui ne sera désormais possédée que par les acteurs ayant accès à l'application web, sans avoir à faire tourner une feuille d'émargement dans la salle de classe à chaque cours, d'autant plus en distanciel... ;
- Les professeurs, qui devront désormais remplir la fiche de présence soi-même, à chaque début de cours, via l'application mobile, en précisant les différents éléments qui constituent un cours (date, heure de début, heure de fin, matière, formation, élèves présents/en retard/absents), cela va permettre un envoi automatique des informations du cours courants nécessaires à la création/au remplissage de la feuille d'émargement pour le jour courant, à l'application web, sur laquelle certains acteurs pourront charger un fichier au format PDF qui sera la feuille d'émargement habituelle, que l'on retrouvera sous format numérique ;
- La secrétaire Madame Girard, ainsi que quelques professeurs dits « responsables » tels que Monsieur Barreau ou encore Madame Barbot, qui pourront accéder à l'application web, afin de consulter et modifier des feuilles d'émargement ;

Les étudiants ne possèdent aucun compte, ni sur l'application web, ni sur l'application mobile. La seule fois où ils accèdent à l'application mobile pour signer se fait donc sur le compte du professeur en charge de ces élèves pour le cours courant. Chacun des professeurs possède un compte sur l'application mobile, ce qui permet de personnaliser l'application selon le professeur afin d'éliminer le superflu et de lui permettre un remplissage efficace et rapide des informations du cours courant. Ils ne possèdent par contre pas de compte sur l'application web. En revanche, les professeurs responsables possèdent un compte sur l'application mobile et sur l'application web, ce qui leur permet à la fois de jouer le rôle d'un professeur qui va remplir la feuille de présence, mais aussi le rôle d'un responsable qui peut consulter, modifier ou supprimer les informations de chacun des cours d'un jour précis ou encore d'un élève précis

B. Cycle de vie du projet

Dans cette partie, nous allons dévier un peu du format habituel que l'on peut retrouver pour un produit commercial d'entreprise à but lucratif, puisque ce projet tutoré n'a pas pour finalité d'industrialiser et de commercialiser quoi que ce soit. Ce projet est donc développé tout simplement pour faciliter la vie des différents acteurs concernés par les feuilles d'émargement et l'application mobile est même disponible gratuitement sur le Play Store, tout comme l'application mobile est disponible gratuitement sur Internet, avec une IP publique. Cependant, des identifiants seront demandés, et la base de données qui contient ces identifiants ne sera détenue dans un premier tout du moins que par l'IUT de Vélizy.

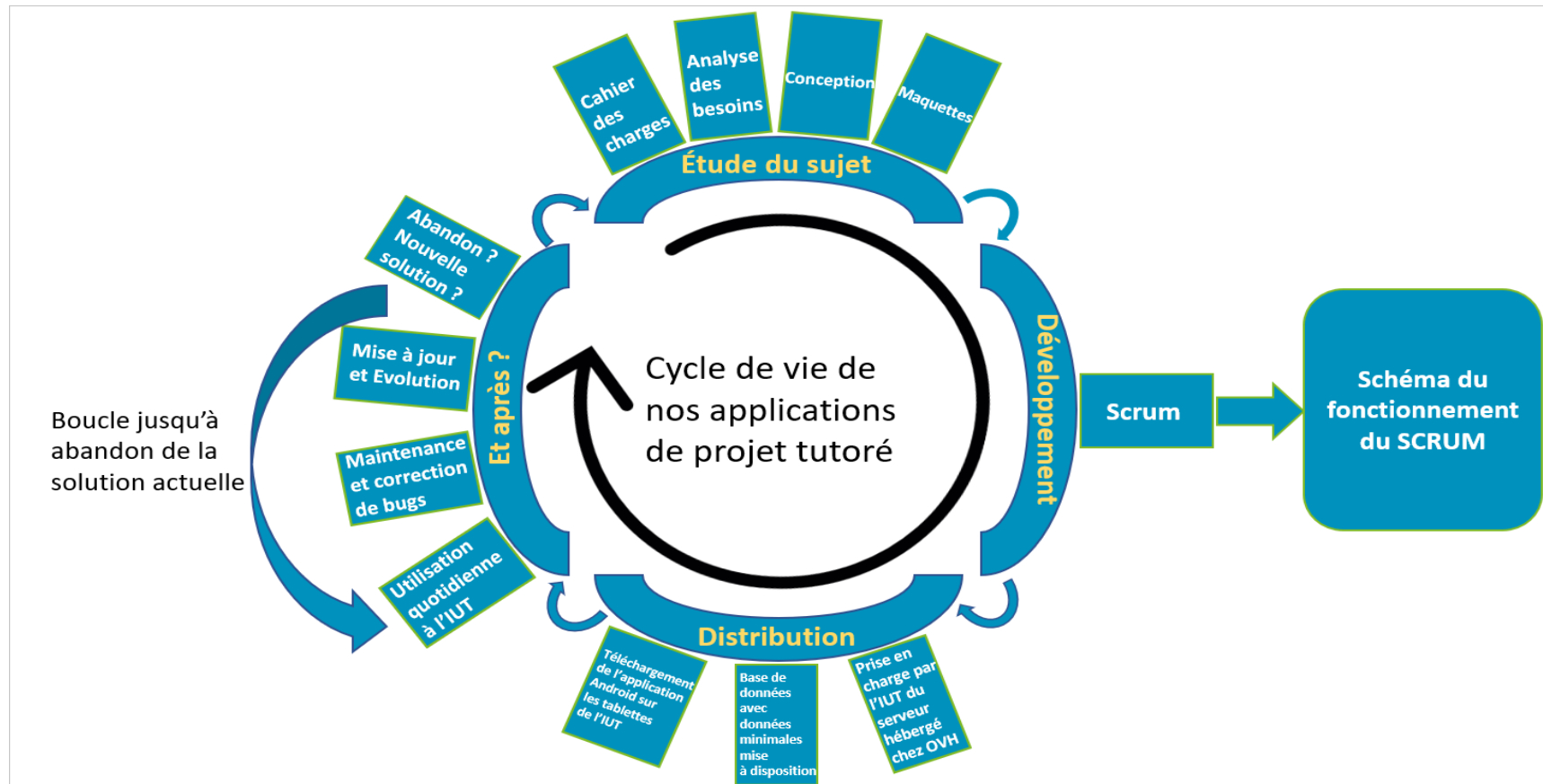


Figure 1 Cycle de vie de notre projet tuteuré

Cycle Scrum

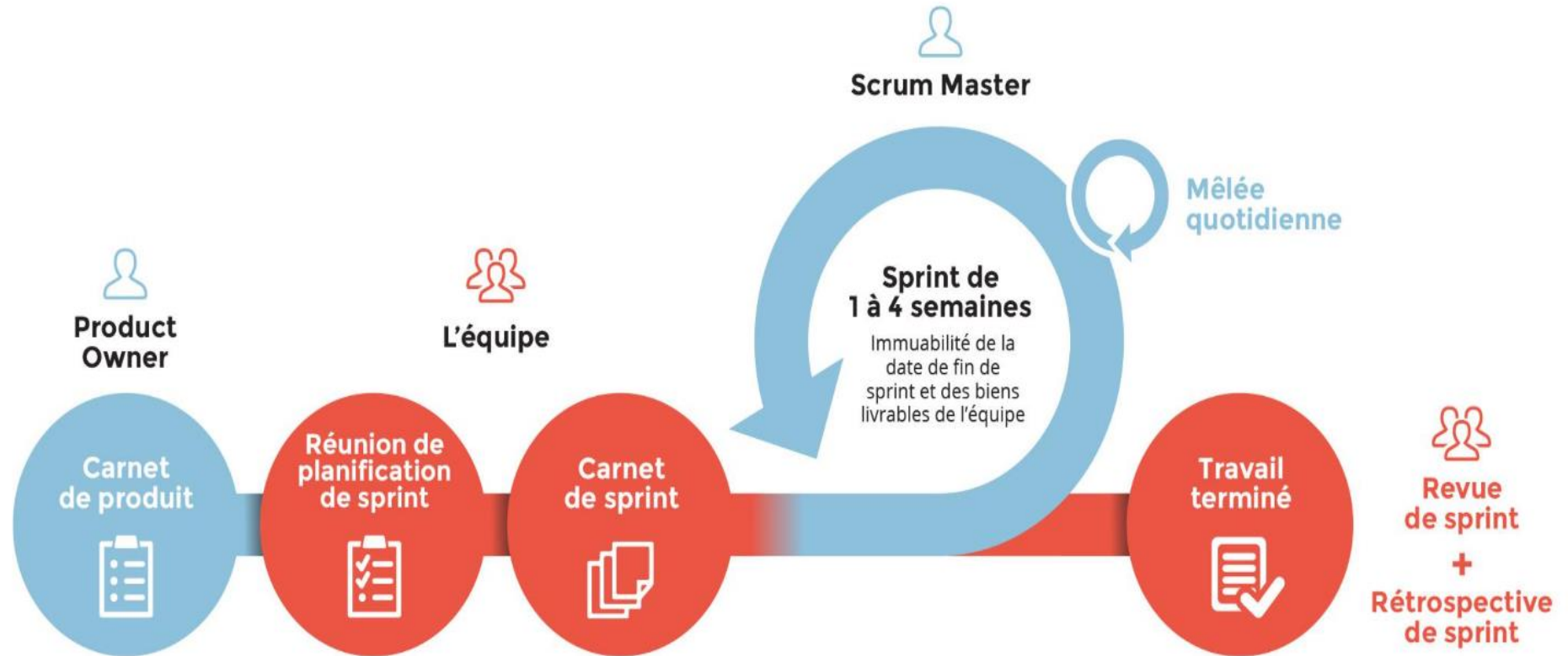


Figure 2 Support de cours de M.Riaz, professeur de MPA durant le 3e semestre pour les FA à l'IUT de Vélizy - Cours sur l'agilité, page 55

C. Diagramme de cas d'utilisation de l'application web

Tout d'abord, parlons du fait que nous ne parlons ici pas du tout de la base de données, alors que pourtant elle est au cœur des différentes actions que l'on peut voir ici. C'est parce que nous nous plaçons du point de vue de l'utilisateur, qui ne se rend donc pas compte de l'interaction qu'il y a avec la base de données lorsqu'il entreprend des actions sur l'une des deux applications. Ici ne figurent donc qu'uniquement les événements possibles que l'utilisateur peut vivre lors de son utilisation des applications.

Le/La responsable ainsi que le secrétariat pourra donc accéder à l'application web à une certaine adresse IP publique. Sur cette application web, l'utilisateur devra avant tout se connecter afin d'effectuer quelque opération, comme indiqué par les flèches « include » qui signifie « inclus », il faut comprendre : « X inclus Y », autrement dit « X nécessite Y ». La connexion est en effet le point d'entrée de l'application web, puisque sinon, n'importe qui pourrait agir sur les opérations normalement réalisées par des acteurs bien spécifiques. Suite à la connexion grâce à des identifiants qui sont stockés dans la base de données, ces acteurs peuvent donc sélectionner manuellement sur l'application web une formation, un professeur, un élève ou bien une matière à ajouter, et cet ajout va se faire dans la base de données ; ces acteurs peuvent également importer un fichier csv qui permettra d'ajouter « automatiquement » des formations, des matières, des élèves, des professeurs... Ces acteurs peuvent notamment faire des recherches calendaires de l'historique des feuilles de présence, suite à quoi ils peuvent consulter un historique des feuilles de présence, sélectionner une feuille de présence en particulier afin de la consulter, de modifier des informations ou tout simplement pour la supprimer. Ils peuvent également consulter des informations sur des élèves depuis la consultation d'une feuille de présence. On remarque que ces trois dernières actions ont une relation de type « extend » avec l'action « consulter une fiche d'absence spécifique », cette relation signifie que « X étend Y » (X est du côté non fléché et Y est du côté fléché de cette relation), autrement dit, « Y ajoute une fonctionnalité à X qui n'est pas forcément réalisée à chaque fois », l'acteur peut donc décider de ne jamais accéder à la fonctionnalité Y. Par exemple ici : les acteurs connectés peuvent décider d'accéder à une fiche de présence spécifique, mais peuvent décider de ne pas modifier cette fiche, ni de la supprimer, alors que les fonctionnalités sont présentes. Cela dépend donc du contexte d'utilisation de l'application par l'acteur concerné.

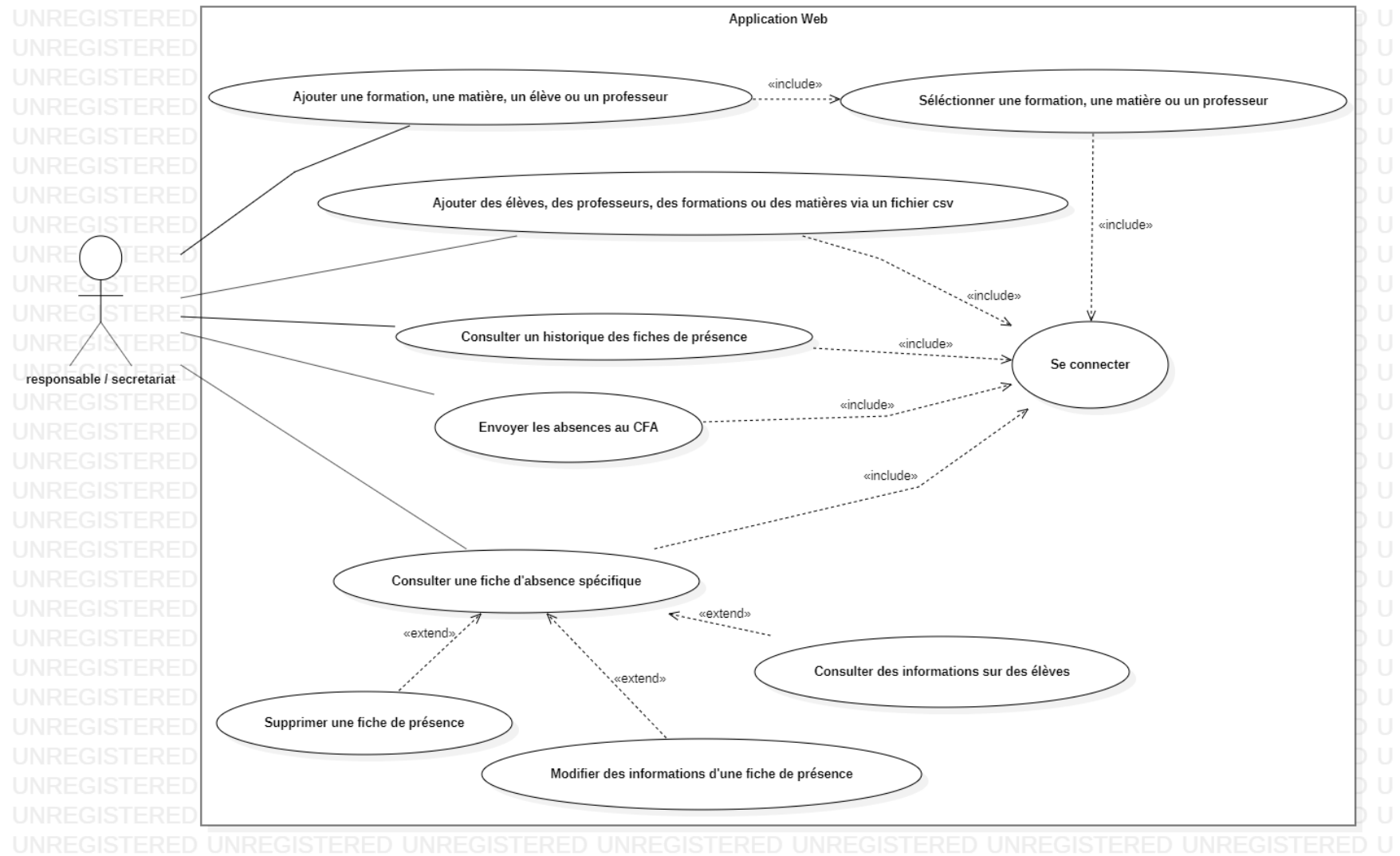


Figure 3 Diagramme de cas d'utilisation de l'application web

D. Diagramme de cas d'utilisation de l'application mobile

Ici, pareillement que pour l'application web, on ne place pas la base de données sur ce diagramme, pour les mêmes raisons que précédemment. Chose à noter, nous avons ici noté le nombre de fois qu'apparaissent certains éléments, mais nous nous sommes notamment restreints aux élèves et aux professeurs, dans leurs actions principales respectives, car c'est ce qui nous paraît important à souligner ici.

On voit qu'il y a ici deux acteurs, le professeur, et l'élève, ceux-ci ont des actions communes, la signature, à réaliser une seule fois dans l'application mobile. Concernant les n élèves (présents) d'une même classe, ils interviennent lors de leur premier cours uniquement (s'ils sont tous présents), en entrant à tour de rôle leur signature, qui sera par la suite automatiquement entrée pour les prochains cours de l'année, puisque cette dernière est enregistrée dans la base de données, lorsque l'élève est présent. Il y a donc n signatures qui sont entrées lors du premier cours d'une certaine classe. Les élèves ne peuvent pas modifier leur signature au cours de l'année, contrairement aux professeurs. Concernant le professeur, il doit tout d'abord se connecter avant d'effectuer n'importe quelle action, à part la réinitialisation de son mot de passe grâce à son email. Paramétrer l'application (localement) et émarger un cours requièrent donc logiquement une connexion au compte d'un professeur. Afin de pouvoir émarger un cours, il faut impérativement que le professeur ait signé sur l'application mobile, et sur le même principe, tous les élèves présents et en retard doivent avoir signé, alors ceux qui sont absents ne le doivent pas. Abordons un petit peu les paramètres de l'application ; on peut voir qu'un certain nombre de fonctionnalités sont à la disposition du professeur, comme le changement d'email par exemple, si jamais le professeur préfère se connecter avec un mail différent (un mail personnel par exemple, sur lequel il serait davantage attentif à d'éventuelles notifications). Comme nous avons pu le voir dans les explications du diagramme précédent, les fonctionnalités ainsi proposées ne sont qu'optionnelles, mais sont pourtant bien existantes dans le cas où elles seraient nécessaires, cela se traduit comme on a pu le constater par une relation « extend », contrairement à d'autres tâches qui sont, elles, nécessaires à atteindre l'objectif premier de l'application, qui est avant tout de pouvoir émarger un cours. C'est pourquoi choisir une formation, un groupe, une matière, un horaire de début et de fin, ainsi que la liste des élèves présents/absents/en retard est obligatoire afin d'émarger un cours, et se traduit par plusieurs relations « include ».

IV. Conception

A. Introduction

Dans cette partie, nous allons présenter la conception de l'application. Nous verrons d'abord la conception très haut niveau du système, puis la conception de notre base de données, la conception de notre système en général puis les spécificités de nos applications. Comme nous travaillons en méthodologie Agile alors cela impacte la façon dont le groupe travaille sur la conception. Nous ne sommes pas face à un modèle en cascade ce qui signifie que finir l'entièreté de la conception n'est pas une condition sine qua non pour commencer la programmation. En effet, avec une méthodologie Agile, nous sommes avec un modèle itératif et incrémental, par conséquent, dans cette partie dédiée à la conception, certains éléments n'auront pas encore une conception précise notamment pour certaines maquettes et fonctionnalités du site web. Cependant, la conception de notre base de données a bel et bien dû être faite avant le début de la partie programmation même si cette base de données est vouée à être modifiée au semestre 4 puisque Monsieur BARREAU a demandé des ajouts aux fonctionnalités prévues de base dans le sujet ce qui nous forcera à repenser le modèle de données pour l'adapter à ces nouvelles contraintes.

B. Conception très haut niveau du système

Notre système sera donc composé comme nous l'avons déjà dit d'un serveur réalisé en Java Spring Boot, d'une application mobile Android et d'une application web Angular. Le serveur sera connecté à une base de données MariaDB. A travers cette partie, nous allons explorer les différents aspects de la conception de ce système.

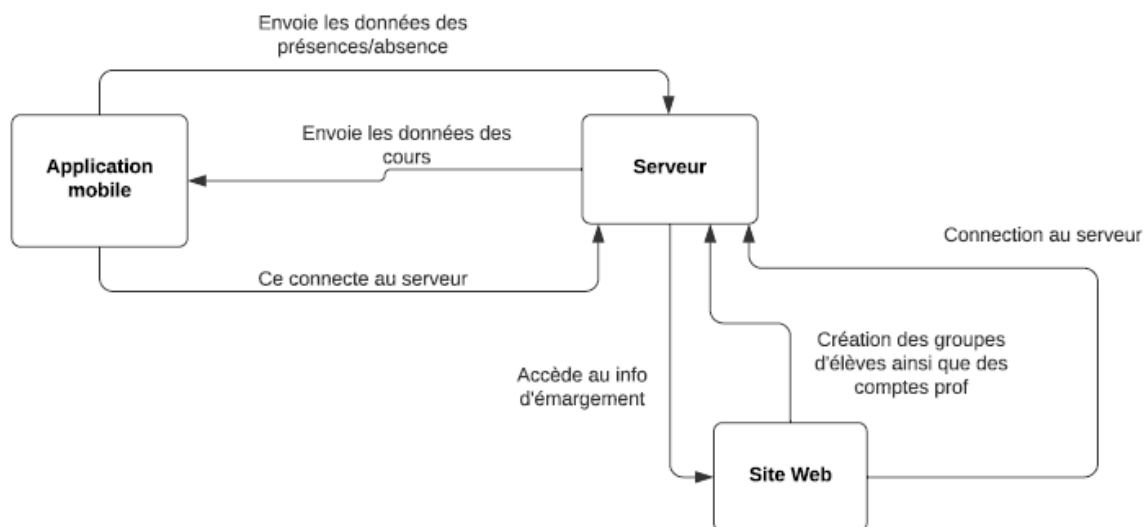


Figure 5 Schéma haut niveau du système

C. La base de données

Avant de parler de la conception du serveur, de l'application web ou de l'application mobile, nous allons d'abord présenter la manière dont nous avons conçu notre base de données. Ce choix découle du fait que la structure du serveur et des applications va se baser sur la structure de notre base de données. C'est notamment du au fait que notre serveur qui est réalisé avec Java Spring Boot utilise également un framework appelé Spring Data JPA qui est un ORM qui va donc nous permettre de faire des requêtes sur notre base de données. Avec cet ORM, nous créons des classes Java qui sont des entités et qui correspondent à une table dans la base de données. Les objets que nous ferons transiter dans nos applications seront donc des répliques exactes de nos tables en base de données, voilà pourquoi nous présentons cette partie avant les autres.

1. Identification des entités et des attributs

Attention : A partir de maintenant nous allons utiliser le terme de « formation », une formation désigne une classe d'une formation dans la suite.

A l'aide de notre analyse du sujet, nous avons déterminé différentes entités qui avaient des attributs propres.

On distingue tout d'abord trois types d'acteurs du système : les professeurs, les secrétaires et les étudiants.

On peut dire que ces acteurs sont des personnes. Une personne est caractérisée par son nom et son prénom. Le professeur, en plus d'être une personne, a la particularité de pouvoir enseigner pour plusieurs formations. De plus, un professeur possède une signature puisqu'il remplit la feuille d'émargement en la signant. Tout comme le professeur, la secrétaire peut être liée à plusieurs formations. En effet, comme nous l'avons déjà dit avec la réforme du BUT, la secrétaire sera amenée à gérer plusieurs classes de FA (en deuxième année et en troisième année). La secrétaire et le professeur doivent pouvoir se connecter sur l'application donc cela implique qu'ils possèdent un compte et donc une adresse mail qui sert de login et un mot de passe.

Concernant les formations, celles-ci sont caractérisées par les différentes matières qu'elles proposent. Une matière est caractérisée par son intitulé. On peut également dire qu'une formation possède un professeur responsable.

Pour revenir aux étudiants, ils sont caractérisés par une signature puisque que comme le professeur ils remplissent la feuille d'émargement avec leur signature. Les étudiants sont liés à une formation et ils peuvent appartenir à un groupe dans leur classe car à partir du semestre 4, la classe des FA se divise en deux puisque les élèves ont des spécialités différentes (algorithmique, maths approfondie VS réseaux et création d'entreprises).

Enfin, un cours correspond à une date, une heure de début, une heure de fin, a été fait par un professeur, correspond à une matière et a été envoyé au CFA ou non.

Chaque cours est associé à plusieurs élèves qui peuvent soit être présents, absents ou en retards à ce cours.

On résume la liste des entités et de leurs attributs :

- **Personne :**
 - A un nom
 - A un prénom
- **Professeur :**
 - Est une personne
 - Enseigne une à plusieurs formations
 - A un compte
 - Enseigne plusieurs cours
 - A une signature
- **Secrétaire**
 - Est une personne
 - Est associé à une ou plusieurs formations
 - A un compte
- **Etudiant**
 - Est une personne
 - Est dans une formation
 - A un numéro de groupe
 - A une signature
 - Est lié à plusieurs cours par le fait d'être soit présent, soit en retard, soit absent (un état de présence)
- **Formation**
 - A un ou plusieurs professeurs
 - Peut avoir plusieurs secrétaires même si en théorie il n'y en a qu'une, on a voulu laisser la porte ouverte à plusieurs secrétaires pour une même formation (exemple : cause d'arrêt maladie, congé maternité)
 - A plusieurs matières
 - A un intitulé
 - A une liste d'étudiants
- **Cours**
 - Est lié à un professeur
 - Est lié à une matière
 - A une heure de début
 - A une heure de fin
 - A une date
 - Est lié à plusieurs étudiants qui sont soit présent, soit en retard soit absent à ce cours

- A été envoyé au CFA ou non (un état)
- Matière
 - A un intitulé
 - Est lié à plusieurs cours
 - Est lié à une formation
- Compte
 - A une adresse mail
 - Peut être rattaché à soit un professeur, soit une secrétaire
 - A un mot de passe

2. Diagramme entité association

A partir de l'identification des entités et des associations que nous avons faites juste avant, on peut désormais s'atteler à la réalisation d'un diagramme entité association.

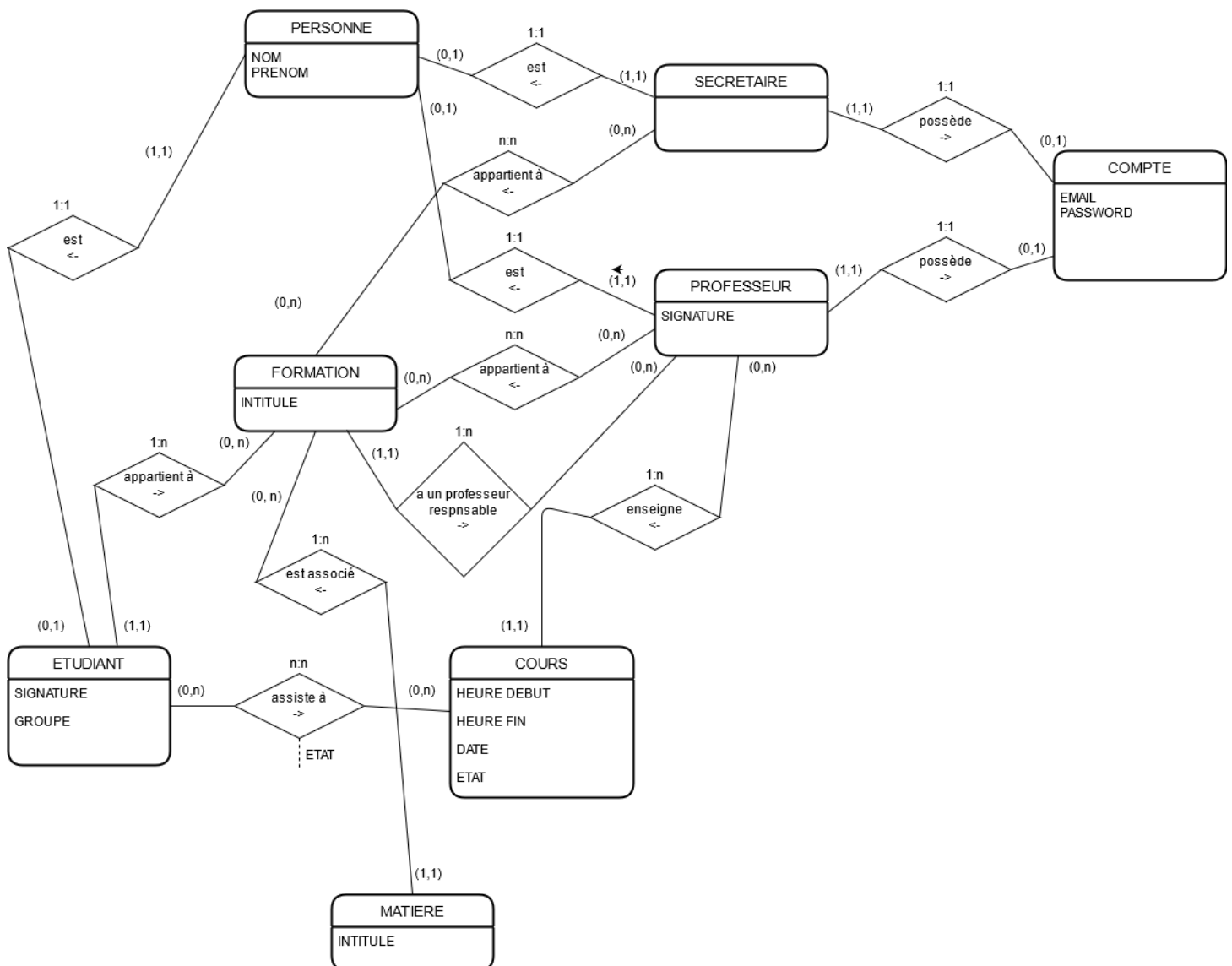


Figure 6 Diagramme entité association de l'application

3. Passage d'un diagramme entité association à un schéma de bases de données

Nous allons désormais expliquer le processus qui nous a permis de passer d'un diagramme entité association au schéma que nous utilisons dans notre base de données.

Tout d'abord comme nous l'avons vu en cours, chaque entité doit avoir sa propre clé primaire. Ensuite se pose la question du placement des clés étrangères qui dépend des associations que nous avons identifié entre chaque entité.

Lorsque nous avons une association de cardinal 1 : 1, avec d'un côté les cardinaux (1,1) et de l'autre les cardinaux (0,1) alors cela signifie que l'entité qui a les cardinaux (1,1) doit posséder la clé primaire de l'autre entité comme le montre l'exemple ci-dessous.

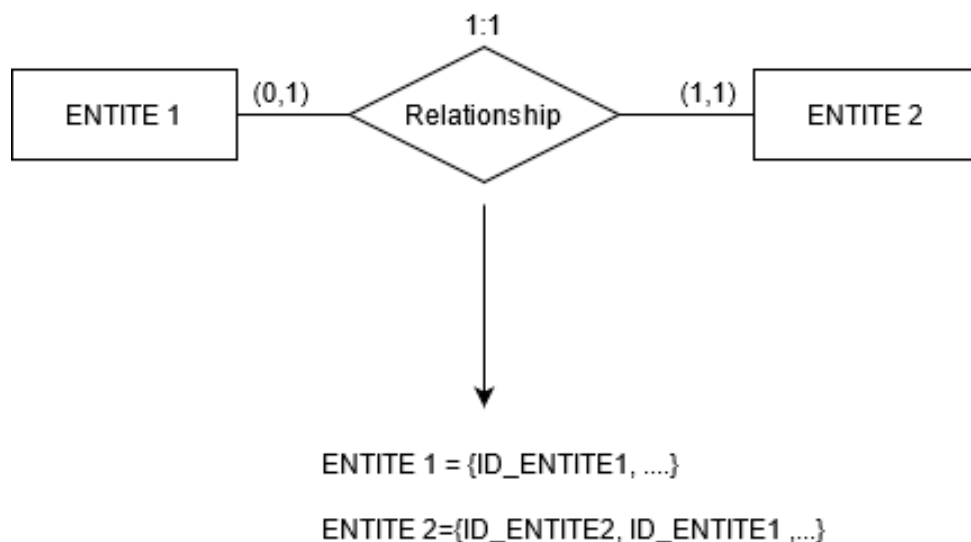


Figure 7 Relation 1:1 et répartition des clés

Lorsque nous avons une association de cardinal 1 : n, alors cela implique l'entité qui a les cardinaux (1,1) de son côté devra posséder la clé primaire de l'autre entité comme le montre l'exemple ci-dessous.

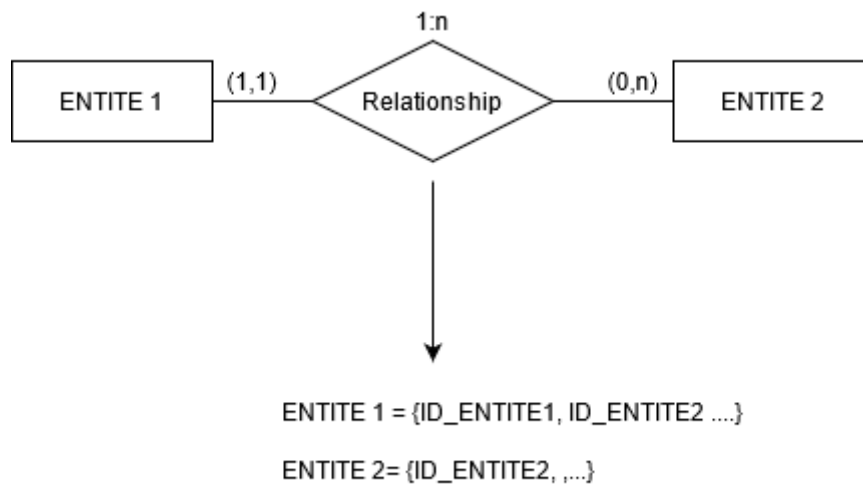


Figure 8 Relation 1:n et répartition des clés

Enfin, lorsque nous avons une association de cardinal n : n, une nouvelle table est créée et qui contient la clé primaire de chaque attribut en plus de sa propre clé primaire. Si cette association possédait des attributs alors les attributs sont stockées dans cette nouvelle table. Voici un exemple ci-dessous.

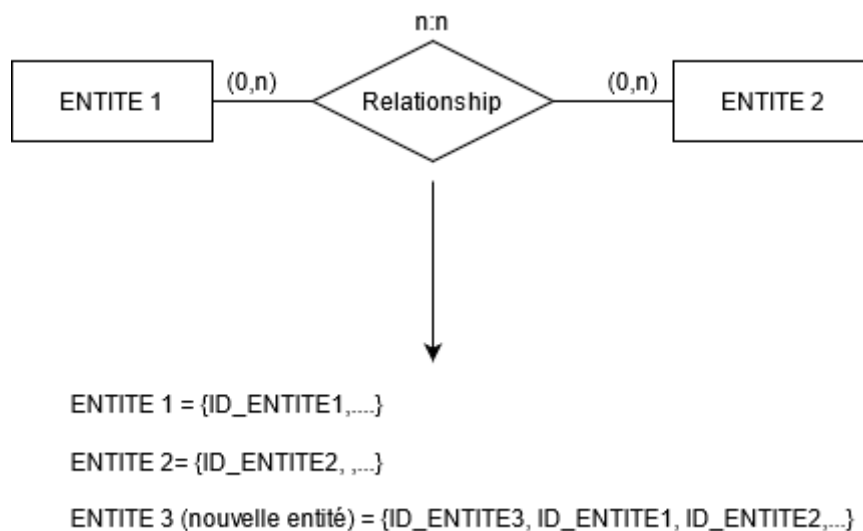


Figure 9 Relation n:n et répartition des clés

En nous servant de ces différentes règles et de notre diagramme entité association, nous avons pu bâtir notre schéma relationnel dont vous pouvez voir le détail ci-dessous.

- PERSONNE = {ID_PERSONNE, NOM, PRENOM}
- SECRETAIRE = {ID_SECRETAIRE, ID_CONTACT, ID_PERSONNE}
- PROFESSEUR = {ID_PROFESSEUR, ID_PERSONNE, ID_CONTACT, SIGNATURE}
- CONTACT = {ID_CONTACT, ID_COMPTE, ADRESSE_MAIL}
- COMPTE = {ID_COMPTE, PASSWORD}
- COURS = {ID_COURS, ID_MATIERE, ID_PROFESSEUR, ID_DATA, BEGIN, END }
- MATIERE = {ID_MATIERE, ID_FORMATION, INTITULE}
- FORMATION = {ID_FORMATION, ID_RESPONSABLE, INTITULE}
- ETUDIANT = {ID_ETUDIANT, ID_PERSONNE, ID_FORMATION, GROUPE, SIGNATURE}
- DATA = {ID_DATA, DONNES, CODE}
- PRESENCE = {ID_PRESENCE, ID_ETUDIANT, ID_COURS, ID_DATA}
- FORMATION_PROFESSEUR = {ID_FORMATION_PROFESSEUR, ID_FORMATION, ID_PROFESSEUR}
- FORMATION_SECRETAIRE = {ID_FORMATION_SECRETAIRE, ID_SECRETAIRE, ID_FORMATION}

On voit ici qu'on a dû créer des tables qui représentent l'association entre une Formation et un Professeur, et entre une Formation et une Secrétaire.

On a également créé une table Présence qui correspond à la relation entre un étudiant et un cours. On voit également le rajout d'une table DATA qui va stocker les différentes constantes de notre application par exemple : on a Présent, En retard, Absent (pour l'état de la présence) et Envoyé et non Envoyé (pour l'état du cours).

Nous avons également séparé l'entité Compte, en deux tables : une table Contact qui stocke l'adresse mail et une autre Compte qui stocke le mot de passe. On a fait cela pour isoler les mots de passe dans une table à part.

Dans le but de pouvoir permettre la modification du mot de passe quand un utilisateur l'a oublié, on a du rajouter une table qui nous permettra de faire cela.

- PASSWORD_RESET_TOKEN = {ID, TOKEN, ID_CONTACT, EXPIRY_DATE}
- Ajout d'une colonne dans COMPTE : RESET

4. Diagramme de notre base de données

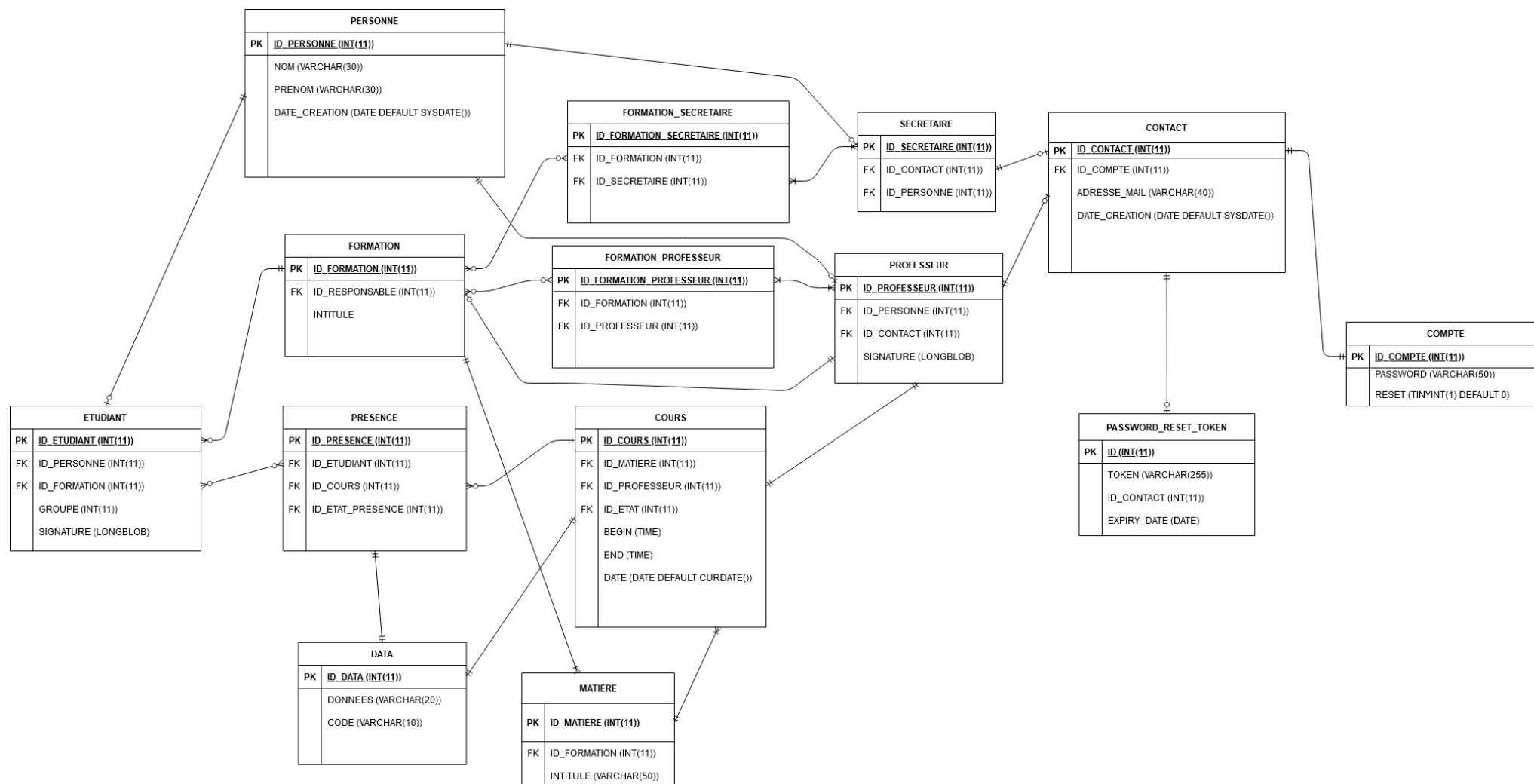


Figure 10 Diagramme des tables de notre base de données

Après avoir réalisé cette conception, nous avons réalisé des tests avec différentes requêtes et la base de données. Celle-ci sera amené à être remanié en vue du semestre 4. En effet, notre volonté de rendre cette application utilisée par l'IUT fait que les exigences des professeurs sont poussées plus loin pour coller de plus en plus avec la réalité du terrain.

D. Structure du système

Maintenant que nous avons une base de données assez bien conceptualisée, nous pouvons nous attaquer à la conception de la structure de nos différents systèmes.

1. Structure du serveur : Java Spring Boot

Comme nous l'avions dit dans l'introduction de la présentation de notre conception de notre base de données, la structure d'un serveur Java Spring Boot qui utilise Spring Data JPA est fortement influencée par la structure même de la base de données puisque les différentes classes qui composent le modèle du serveur font directement écho aux tables en base de données. En réalité le terme d'API serait plus approprié que de « serveur » puisque Java Spring Boot nous permet de créer des API Rest.

a) *Modèles*

Chaque classe présente dans le modèle correspond donc à une table dans la base de données. Seule les tables qui représentent uniquement une association « n : n » entre deux tables et qui n'ont pas d'attribut propre ne sont pas présentes dans le modèle. Cela signifie donc FORMATION_SECRETAIRE et FORMATION_PROFESSEUR n'ont pas de classes associées dans notre modèle mais que la table PRESENCE en a une puisqu'elle a son propre attribut. On a donc la liste des classes suivantes dans notre modèle :

- Compte (équivalent de la table COMPTE)
- Contact (équivalent de la table CONTACT)
- Cours (équivalent de la table COURS)
- Etudiant (équivalent de la table ETUDIANT)
- Formation (équivalent de la table FORMATION)
- Matière (équivalent de la table MATIERE)
- PasswordResetToken (équivalent de PASSWORD_RESET_TOKEN)
- Personne (équivalent de PERSONNE)
- Presence (équivalent de PRESENCE)
- Professeur (équivalent de PROFESSEUR)
- Secrétaire (équivalent de SECRETAIRE)
- TableData (équivalent de DATA)

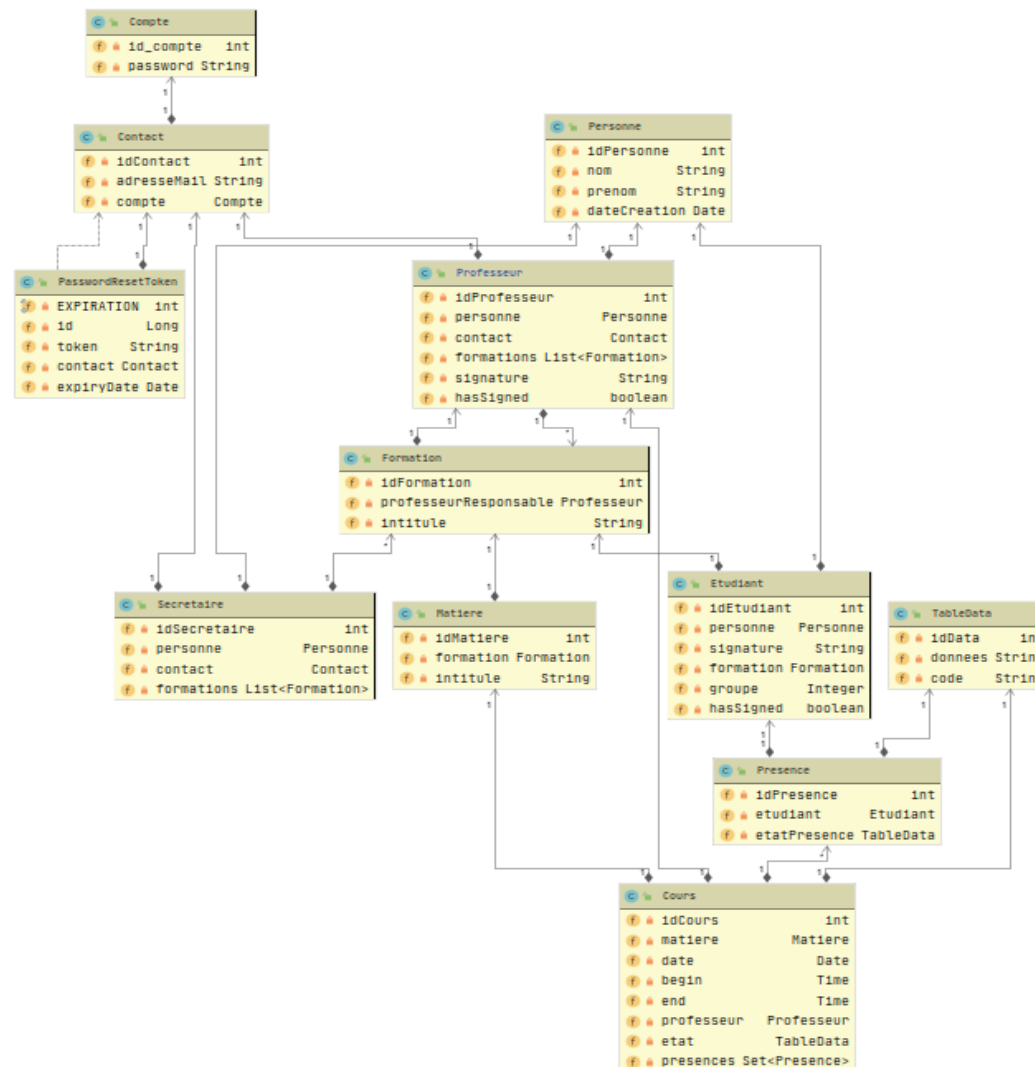


Figure 11 Diagramme de classes du modèle du serveur

Il est intéressant de noter que l'on peut rajouter des champs qui ne sont pas des colonnes en base de données. Par exemple, on voit que la classe Etudiant et la classe Professeur ont un champ « hasSigned » qui est un booléen. Il s'agit d'un champ `@Transient` qui n'est donc pas une colonne en BDD. Ce champ « hasSigned » sera rempli à true si la signature n'est pas NULL en base de données, et à false dans le cas contraire. Ce champ permettra aux applications mobiles et web si le professeur ou l'étudiant a déjà signé sans qu'on ait besoin d'envoyer la signature qui est un document assez lourd et ralentirait le réseau. A part cela notre structure des classes est vraiment similaire à la base de données.

b) Controllers

Les controllers sont les classes qui vont recevoir les requêtes HTTP des autres applications. Ce sont les end points de notre API et chaque méthode de chaque controller est lié à une requête http bien particulière (url particulier, paramètres particuliers...). Il existe un controller pour chaque classe du modèle. Chaque controller va traiter des informations relatives à sa classe. Par exemple, il existe une classe EtudiantRestController qui va recevoir des objets Etudiants, en renvoyer en fonction des requêtes HTTP que ce controller reçoit. Nous essayons de respecter le principe LCRUD pour nos controllers. LCRUD est l'acronyme de List, Create, Read, Update et Delete.

- List : renvoie une liste d'objets selon potentiellement différents critères (appelé grâce à une méthode GET)
- Create : réception d'un objet qui sera ajouté en base de données (appelé grâce à une méthode POST)
- Read : renvoie un objet précis souvent grâce à son id en base de données (appelé grâce à une méthode GET)
- Update : modifie la valeur d'un objet en base de données avec celui envoyé par la requête http (appelé grâce à une méthode PATCH)
- Delete : supprimer un objet de la base de données (appelé grâce à une méthode DELETE)

Les controllers sont donc des API qui serviront de pont de communication entre le serveur et les autres applications. Une fois que le controller reçoit la requête il va appeler une méthode du service qui lui est associé. Il est donc temps d'aborder les services.

c) Services

Les services sont les ponts de communication entre les controllers et les DAO que nous aborderons après. Tout comme les controllers, il y a un service pour chaque objet du modèle. Il possède dans ses champs une interface qui va être notre DAO dont nous parlerons dans notre prochaine partie.

d) DAO

Les DAO JPA sont des interfaces qui héritent de l'interface JpaRepository qui va nous permettre de faire des requêtes SQL dans la base de données via des noms de méthodes ou des annotations (@Query). Il y a une DAO pour chaque objet du modèle et il va se concentrer sur des requêtes pour la table qui correspond à sa classe. Par exemple la DAOEtudiant fera principalement des requêtes à la table ETUDIANT et ajoutera, supprimera, modifiera des Etudiants.

e) *Diagramme de packages du serveur*

Nous avons donc vu, les éléments essentiels qui composent notre serveur et il est intéressant de résumer tout cela avec un diagramme de packages. Nous sommes donc avec un code qui est subdivisé en quatre packages. Nos controllers ont accès aux services et les services ont accès aux dao. Ces trois packages ont accès au package models. Au dessus de tout ces packages se trouvent notre « main » qui va lancer notre programme. Ci-dessous vous pourrez trouver le diagramme de packages du serveur.

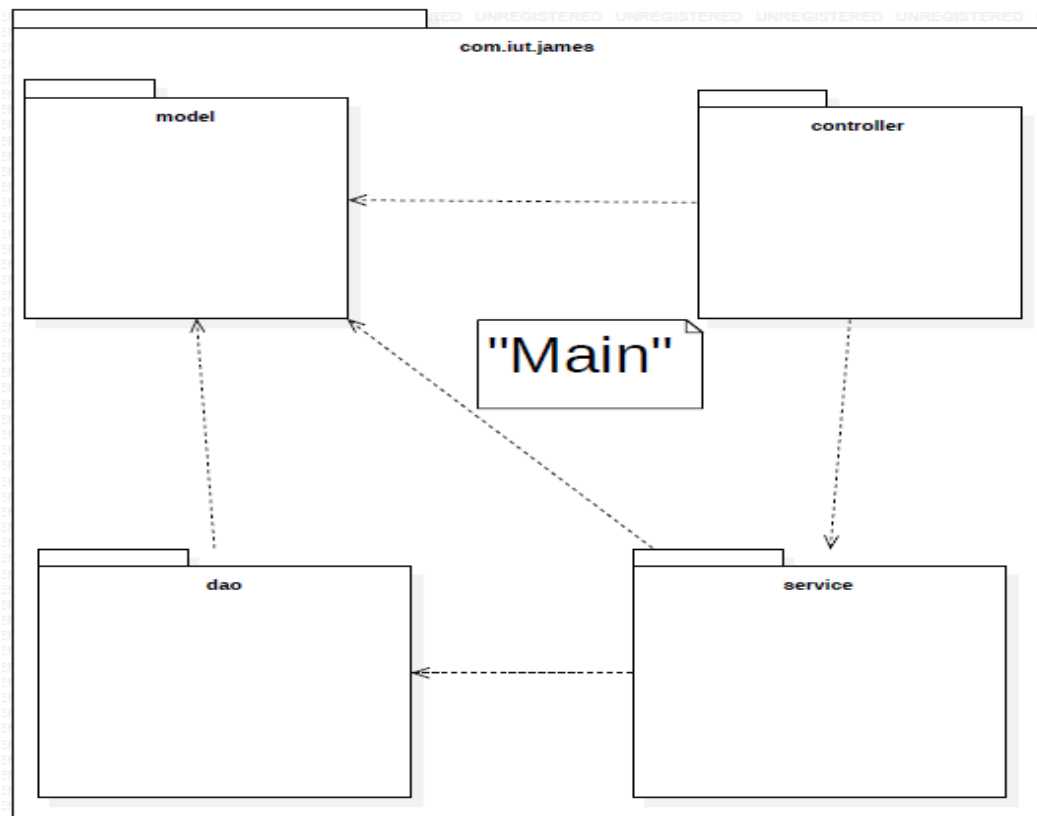


Figure 12 Diagramme de packages de notre API

f) *Communication interne au serveur*

Comme nous venons de le voir, pour chaque classe du modèle, il y a un controller, un service et une dao. Regardons maintenant comment se déroule la communication interne entre ses éléments avec l'exemple d'une requête GET sur l'`EtudiantRestController` qui a en paramètre un id d'étudiant en base de données et qui est donc censé renvoyer l'`Etudiant` qui a cet ID.

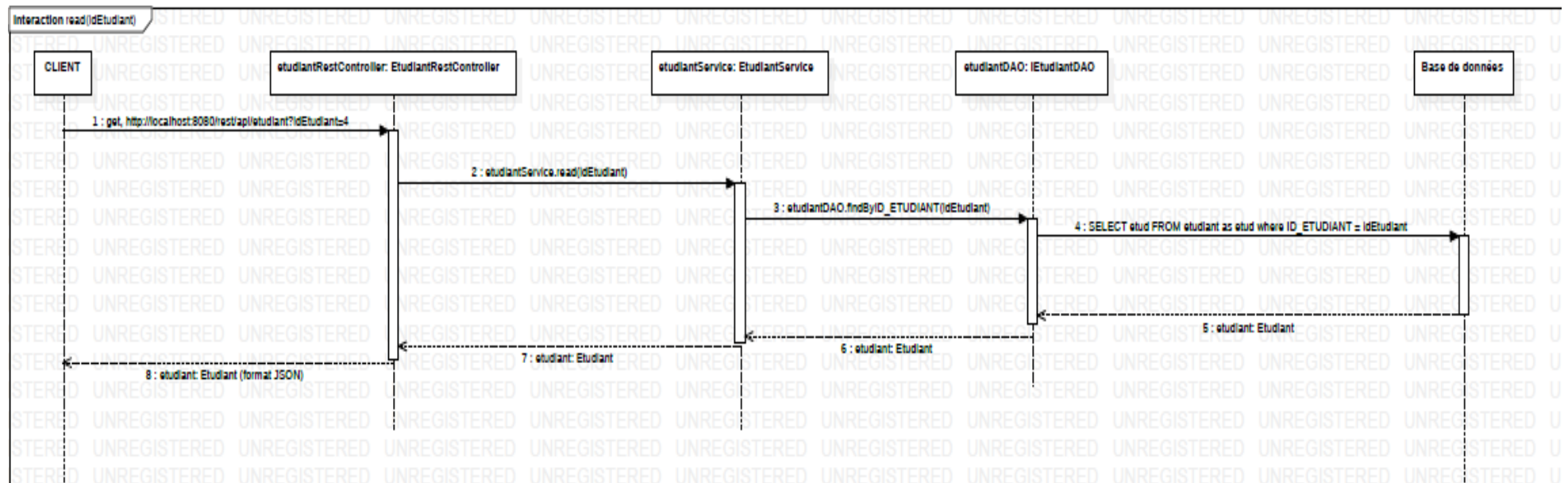


Figure 13 Diagramme d'interaction du système lors de la requête `http://localhost:8080/rest/api/etudiant?idEtudiant=4`

2. Structure de l'application mobile

Etudions maintenant la structure de notre application mobile qui sera réalisé en Android et donc en Java/XML.

a) Modèles

Concernant les classes de notre modèle de notre application mobile, elles sont forcément très similaires à celle de notre serveur et donc de notre base de données. La seule différence c'est que l'application mobile ne possède que les classes qui sont nécessaires à son fonctionnement. Par exemple, la classe Secrétaire n'est pas présente car les secrétaires n'ont pas accès à l'application mobile donc nous n'avons pas besoin de cette classe. De plus, la classe Compte n'est pas non plus ici car nous n'avons pas besoin des mots de passe dans l'application mobile, ceux-ci sont seulement gérés par le serveur. Enfin, nos classes Professeur et Etudiant n'ont pas de champ « signature » car comme nous l'avons vu plus tôt, le serveur n'a pas besoin de renvoyer la signature aux applications car il s'agit d'une information assez lourde (image en base 64). L'application saura si l'étudiant ou le professeur a signé grâce à un booléen.

Notre classe Etudiant aura un champ positionSpinner qui nous permettra de savoir le numéro du statut de présence sélectionné par le professeur pour cet étudiant.

Voici le diagramme de classes de notre modèle :

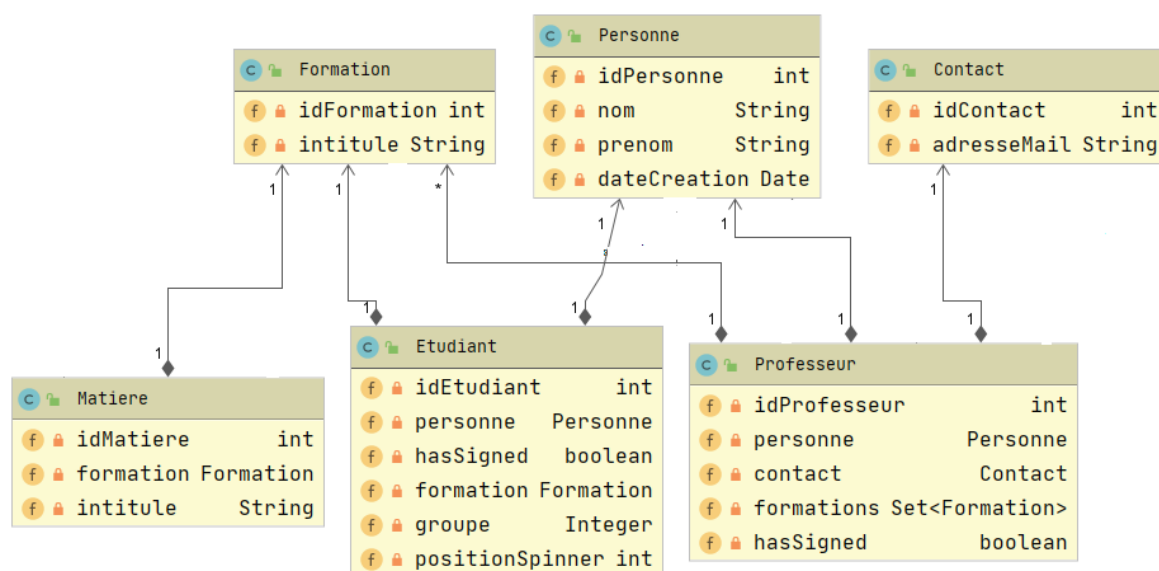


Figure 14 Diagramme de classes du modèle de l'application mobile

b) Activités

Les activités sont les différentes « pages » de notre application mobile. Une activité Android comporte une classe Java qui hérite de AppCompatActivity ainsi qu'un code XML qui permet de mettre en forme son apparence. Nous avons identifié plusieurs activités qui seront nécessaires à la bonne réalisation du projet.

- LoginActivity : l'activité qui se trouve au lancement de l'application et qui permet de se connecter à l'application.
- WelcomeActivity : l'activité à laquelle on accède après s'être connecté. C'est une interface d'accueil qui permet d'accéder à d'autres activités
- ForgotPasswordActivity : l'activité dans laquelle on peut entrer son adresse mail pour recevoir un email pour changer son mot de passe. On peut y accéder depuis LoginActivity.
- ParametreActivty : activité dans laquelle on peut changer certains paramètres de notre application
- EmailChangeActivity : activité qui permet de changer l'email de notre compte
- PasswordChangeActivity : activité qui permet de changer le mot de passe de notre compte
- SignatureActivity : activité qui permet de rentrer son mot de passe pour un professeur ou un élève
- AppelActivity : activité qui permet d'émarger un cours

Les activités peuvent utiliser des Views et peuvent faire appel à des services que nous allons présenter dans les prochaines sections.

c) Views

Les Views vont être des composants Android qui nous sont propres et que l'on peut réutiliser dans d'autres activités. Elles possèdent une classe Java et un fichier XML qui leur sont propres. Nous aurons notamment besoin d'une View pour que l'on puisse dessiner la signature dans l'application.

d) Les services

Les services sont les classes Java qui vont faire les requêtes http au serveur et qui vont récupérer les réponses de ce dernier. Dans un souci de découpage et pour une meilleure lisibilité, il y a un service par classe dans le modèle. Chaque service essaye de respecter le principe LCRUD que nous avons déjà abordé dans la partie sur les controllers dans la structure du serveur. Les services s'occupent également de manipuler les JSON renvoyés par le serveur en les convertissant en objets des bonnes classes.

e) *Diagramme de packages*

Finalement, notre projet Android par deux grands packages à savoir la partie xml dans le package res et la partie java dans le package java. Chaque package étant lui-même subdivisé en d'autres packages.

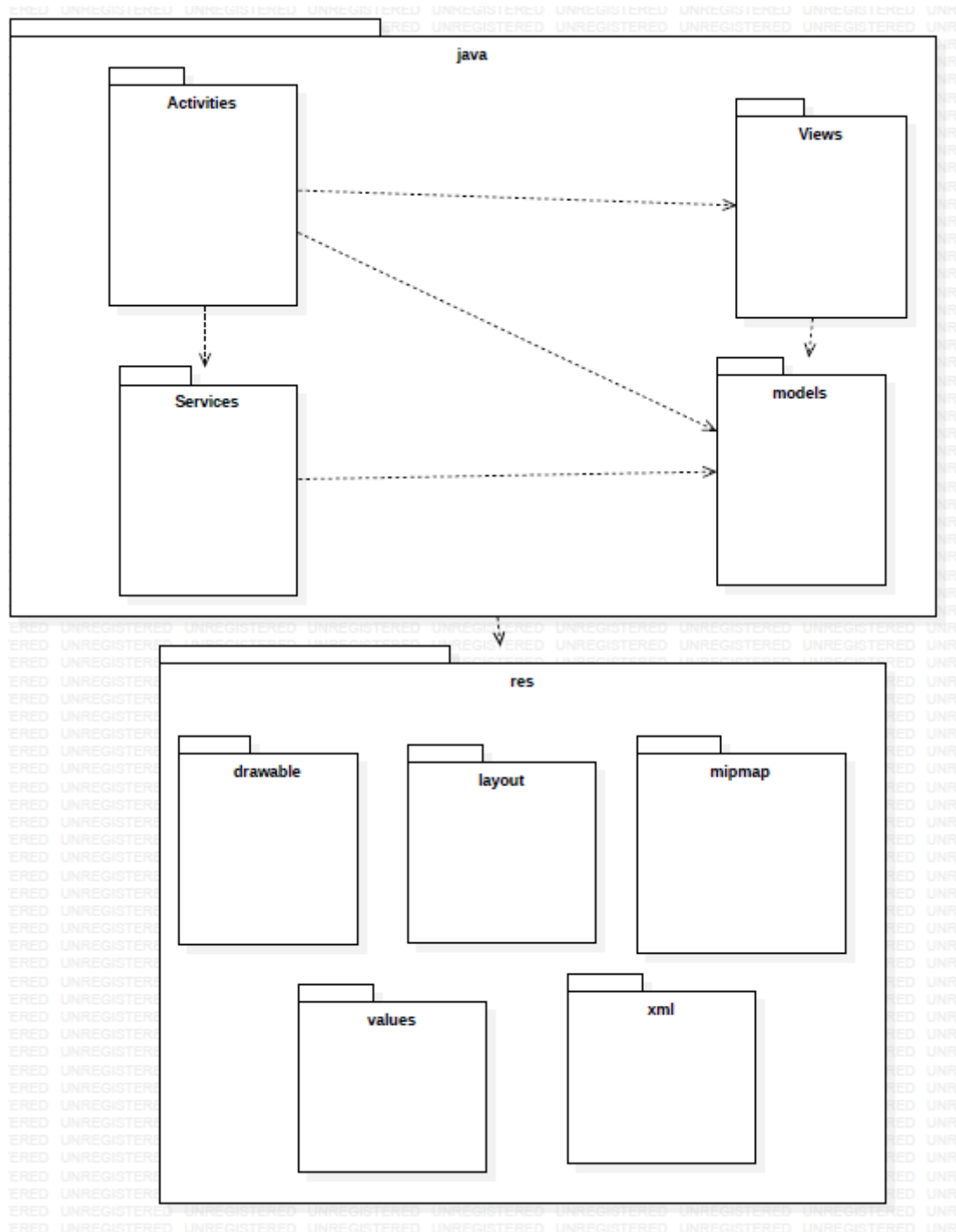


Figure 15 Diagramme de packages de l'application mobile

3. Structure de l'application web

Lançons-nous maintenant dans la conception de notre application web qui sera un frontend réalisé grâce au Framework de Google : Angular et donc avec le langage TypeScript qui est une surcouche de JavaScript. La manière dont notre application sera conçue sera donc fortement impactée par la façon dont Angular fonctionne.

a) Modèles

Encore une fois on va avoir un modèle très proche de celui du serveur et donc de la base de données. Nos classes du modèle reprendront donc celles du serveur avec la particularité que les champs aient exactement les mêmes noms. En effet, en faisant cela, alors au moment où notre Serveur Java Spring Boot nous renverra notre JSON avec les informations alors le client Angular pourra directement « mapper » ce JSON afin de créer un objet avec directement la bonne classe et les champs bien remplis.

Voici le diagramme de nos classes TypeScript dans l'application mobile :

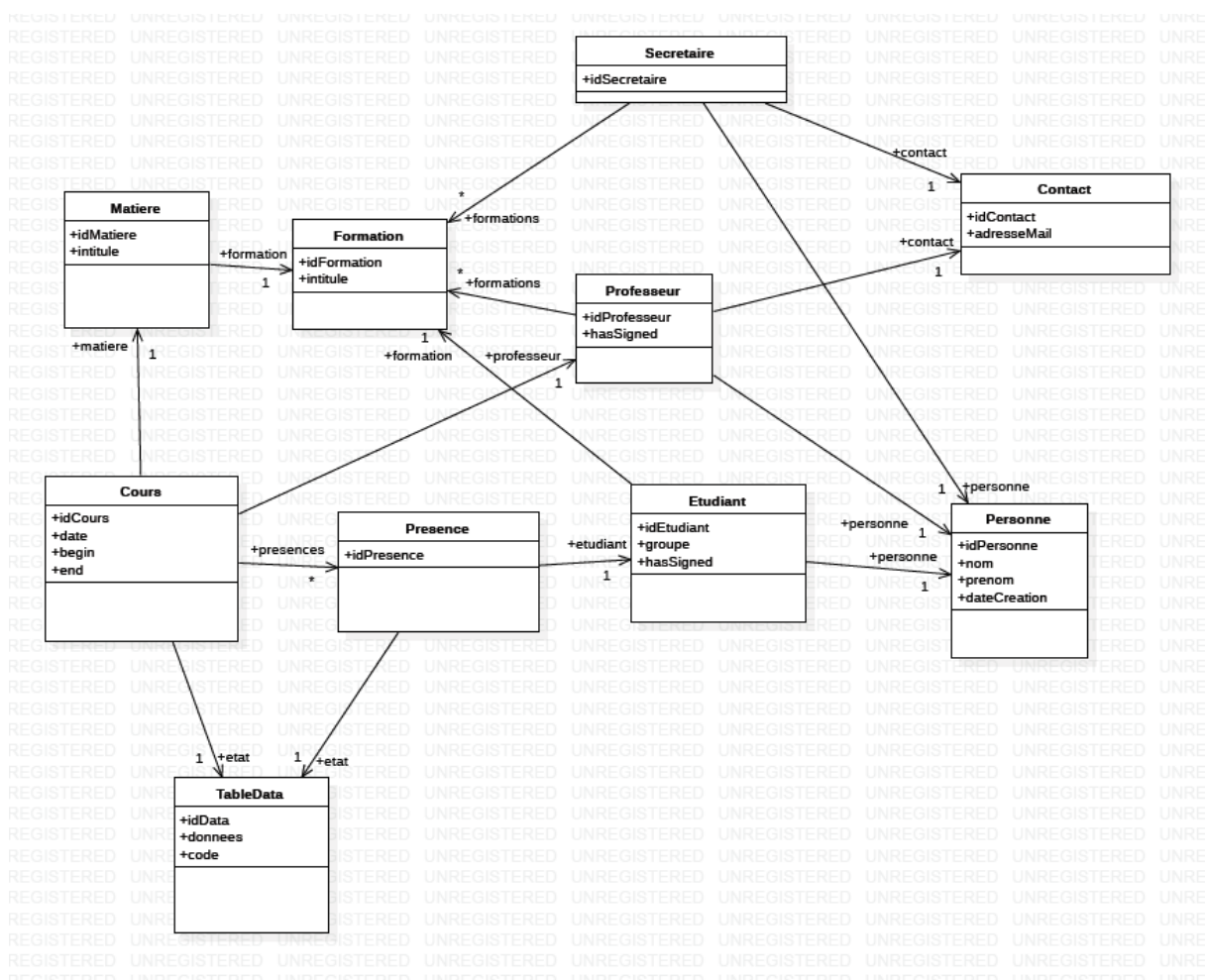


Figure 16 Diagramme de classes du modèle de l'application web

b) Composants Pages

Il y a ensuite nos composants Angular. Un composant Angular correspond au regroupement de 4 fichiers liés entre eux : un fichier HTML (le template de notre composant), un fichier CSS (le style de notre composant), et deux fichiers Typescript : un qui correspond à celui dédié pour les tests et l'autre qui va gérer les données de notre template, les interactions de l'utilisateur etc.

On peut utiliser nos composants comme des pages HTML à part entière. Forcément beaucoup de composants comme ceux-ci seront présents et ils utiliseront le modèle pour afficher des données. Parmi les composants de type « pages » on peut notamment mentionner :

- Composant home -> page d'accueil du site qui présente le projet
 - Composant login -> page avec un formulaire pour rentrer notre login et notre mot de passe pour se connecter
 - Composant reset -> page avec un formulaire qui nous permet de changer de mot de passe quand on y accède avec un token de réinitialisation spécifique
 - Composant dashboard-> page après la connexion qui nous montre des informations pertinentes comme le résumé d'une journée de cours, et qui nous permet de naviguer vers d'autres pages
 - Composant fiche d'absence -> page qui montre les informations d'une journée de cours pour une formation, et qui permet de générer une fiche de présence.
 - Composant modification cours->page qui permet de modifier un cours
 - Composant historique-> page qui nous permet de voir l'historique des journées de cours
 - Composant création formation-> page qui permet de créer une formation
 - Composant ajoutProfesseurFormation->page qui permet d'ajouter des professeurs à des formations
 - Composant AjoutMatiere-> page qui permet de créer une matière et de l'ajouter à une formation.
 - Composant AjoutEleve-> page qui permet d'ajouter des élèves
- Et encore bien d'autres composants de type « pages » qui viendront s'ajouter avec les modifications demandées par Monsieur Barreau au semestre 4. Les composants « pages » peuvent inclure d'autres composants en eux et utilisent les services pour obtenir les données nécessaires à leur affichage.

c) Les composants réutilisables

On peut également créer des composants assez génériques que nous pouvons réutiliser dans toute l'application ce qui évite de faire de la duplication de code et d'avoir un style global uniforme. Nous avons

notamment créé un composant table qui permet d’afficher dans une table les informations sur les journées de cours dans plusieurs endroits de l’application. Nous avons également créé des modales (pop up) notamment pour les cookies, la demande de reset de mot de passe etc.

d) Les services

Comme dans l’application mobile, nous avons des services qui vont nous permettre de communiquer avec le serveur. Ils sont bien évidemment découpés en fonction des différentes classes de notre modèle pour plus de lisibilité et tente de respecter au maximum le principe LCRUD que nous avons déjà expliqué dans la partie Structure du serveur sur les controllers.

e) Diagramme de packages

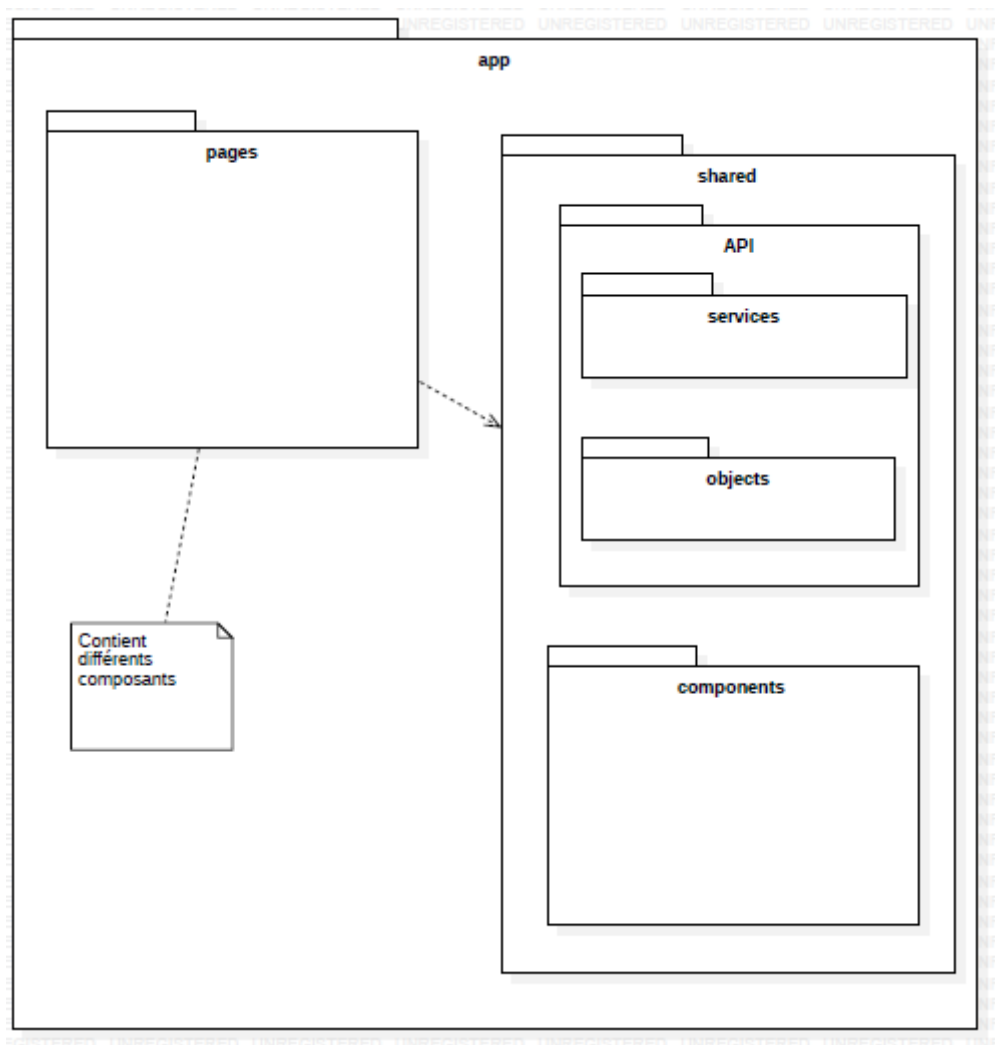


Figure 17 Diagramme de packages de l'application web

4. Schéma des communications entre les applications et l'API

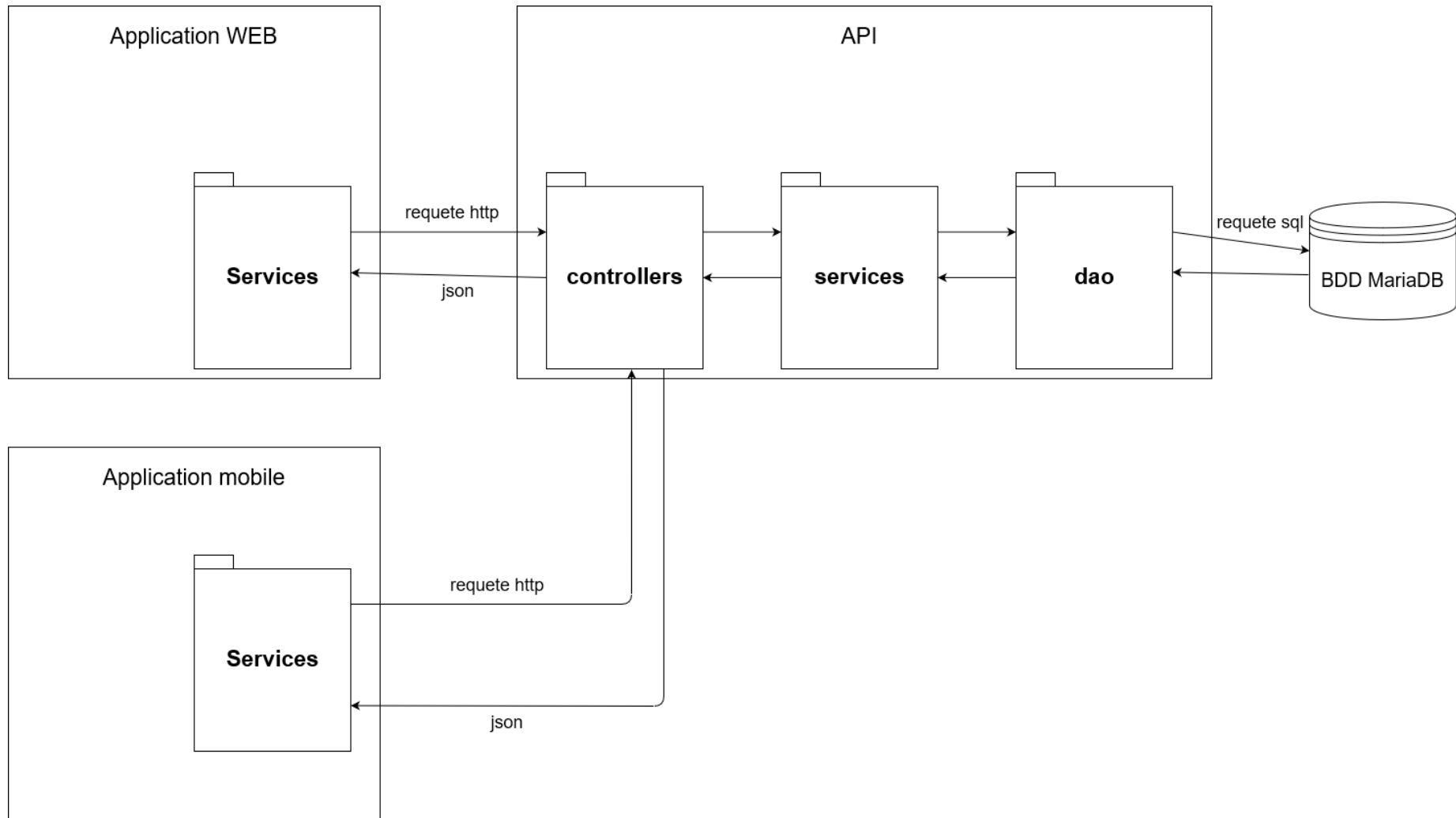


Figure 18 Schéma des communications entre les applications et l'API

E. Spécification détaillé des interfaces

1. Charte graphique

Dans cette partie, nous allons expliquer de quoi est composé notre charte graphique

a) *Logo*

Tout d'abord afin de donner une réelle identité visuelle à notre projet nous avons contacté un ami d'Edvans afin que ce dernier nous concocte un logo pour l'application et pour notre groupe. Nous voulions que ces logos montrent un manchot en référence au fameux manchot de Linux : Tux. Pour l'application mobile, nous voulions que le logo évoque le fait que ce soit une application qui permette de remplir une feuille d'émargement. Pour cela, le nez du manchot a une forme proche de la plume d'un stylo. Pour le logo de notre groupe, nous voulions que ce manchot fasse le mouvement du « DAB » qui est passé de mode depuis 2017 mais qui est comme une blague interne au groupe.

Nous avons donc obtenu les deux logos que vous pouvez voir ci-dessous.



Figure 19 Logo des applications web et mobile: JAMES






Figure 20 Logo de notre groupe





b) Couleurs

Concernant les couleurs des applications, nous étions plutôt partis sur des couleurs assez sombres notamment pour le site web comme vous pourrez le voir dans les maquettes web. Concernant l'application mobile nous n'avions pas d'idée visuelle précise en tête. Après une réunion avec Monsieur BARREAU et Monsieur HOGUIN, il a été décidé que les couleurs du logo de l'UVSQ seraient utilisées par les applications. On a donc la charte graphique suivante :

- **Application Mobile**

-  : fond de l'application (rvb : 0, 144, 188)
-  : couleur du texte (rvb : 255, 255, 255)
-  : couleur des boutons (rvb : 128, 186, 39)

- **Application WEB**

-  : couleurs des boutons, du header et du footer (rvb : 0, 144, 188)
-  : couleur de fond (rvb : 255, 255, 255)
-  : couleur au survol des boutons (rvb : 105, 4, 60)
-  : couleur du texte (rvb : 0, 0, 0)

c) Typographie

Concernant la typographie nous avons pris les polices par défaut pour l'application mobile et l'application web.

2. Interfaces application mobile

a) Interface de connexion

- Champ de texte 'Login'
- Champ de texte 'Mot de passe'
- Bouton 'Connexion', si le login et le mot de passe sont bons on va à l'interface de menu

- Bouton ‘Mot de passe oublié ?’, emmène vers l’interface mot de passe oublié



Figure 21 L'interface de connexion

b) Interface de menu

- Bouton ‘Emarger un cours’ : emmène vers l’interface pour émarger un cours
- Bouton ‘Paramètres’ : emmène vers l’interface de paramètres
- Message de bienvenue



Figure 22 Interface de menu

c) *Interface pour émarger un cours*

- Menu déroulant (Spinner) pour choisir sa formation
 - Les sous groupes sont proposés, par exemple INFA2, INFA2-1, INFA2-2
- Menu déroulant pour la matière (affichage s'adapte à la formation choisie)
- Deux composants permettant de régler des heures
- Liste des élèves
 - Nom, Prénom
 - Menu déroulant ('Présent', 'En retard', 'Absent'), à 'Présent' par défaut
 - Bouton 'Signature' activé si l'élève n'a pas signé et qui conduit vers l'écran de signature
- Bouton Valider
 - Ouvre une fenêtre qui récapitule le cours et propose de valider ou non

| Formation | |
|---------------------|---------|
| James | |
| Matière | |
| Cours de SCRUM | |
| Heure de début | |
| 08 | 59 |
| 09 | 00 |
| 10 | 01 |
| Heure de fin | |
| 09 | 59 |
| 10 | 00 |
| 11 | 01 |
| John-JR Doe | Présent |
| Ludvine Doe | Présent |
| Martin Doe | Présent |
| Philippe Doe | Présent |
| VALIDATION DU COURS | |

Figure 23 Interface d'émargement d'un cours

d) *Interface des paramètres*

- Bouton ‘Modifier son mot de passe’ : emmène vers l’interface de modification du mot de passe
- Bouton ‘Modifier son adresse mail’ : emmène vers l’interface de modification de l’adresse mail
- Menu déroulant des langues parmi quatre choix ‘Anglais’, ‘Espagnol’, ‘Roumain’, ‘Français’.
- Bouton ‘Changer signature’ : emmène vers l’interface de changement de signature pour un professeur

On peut voir-ci-dessous, une première maquette de l’interface de paramètres. En effet, celle-ci devait permettre de changer les couleurs de l’application mais l’idée a été abandonnée après qu’il a été décidé que les couleurs de l’application reprendraient celles du logo de l’UVSQ (bleu et vert). De plus, la maquette ne contient pas le changement de signature pour le professeur qui n’était pas prévu au

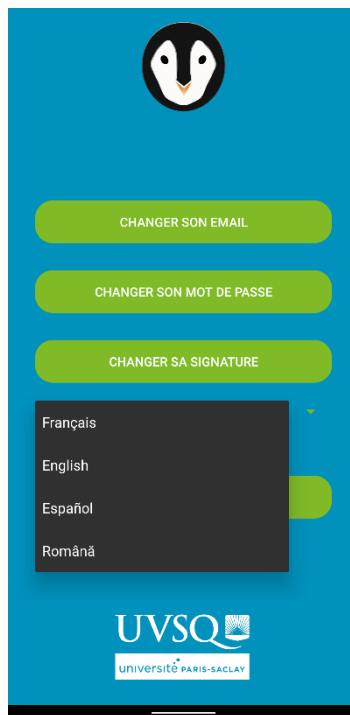


Figure 24 Interface de paramètres

début.

e) *Interface de modification de l’adresse mail*

- Champ textuel ‘nouvelle adresse mail’
- Champ textuel ‘retaper l’adresse mail’

- Bouton 'valider' : enregistre l'adresse mail et renvoie vers l'interface des paramètres, si les deux champs textuels au-dessus sont égaux

f) *Interface de modification du mot de passe*

- Champ textuel 'nouveau mot de passe'
- Champ textuel 'retaper le mot de passe'
- Bouton 'valider' : enregistre le mot de passe et renvoie vers l'interface des paramètres, si les deux champs textuels au-dessus sont égaux

g) *Interface 'mot de passe oublié'*

- Champ textuel 'adresse mail'
- Bouton 'Envoyer' : si l'adresse mail est dans la base de données alors on envoie un mail à avec un token qui va nous permettre de modifier son mot de passe sur l'application web

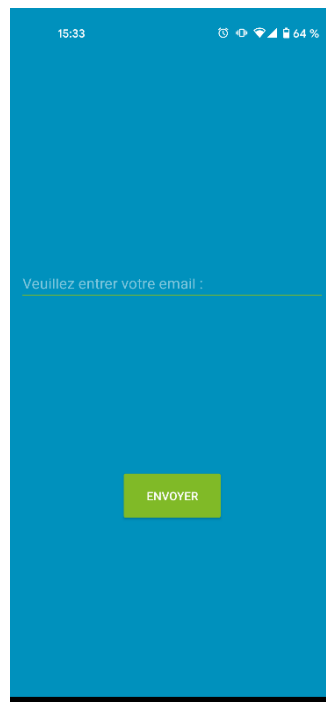


Figure 25 Interface pour demander le reset du mot de passe

h) *Interface de signature*

- Une zone de dessin assez grande
- Un bouton reset qui permet d'effacer la zone de dessin

- Un bouton valider qui permet d'envoyer sa signature quand on la juge bonne



Figure 26 Interface de signature

3. Interfaces application web

Les maquettes que nous vous proposerons dans cette partie ont été réalisées avec Bootstrap Studio qui a donc la particularité de produire du code HTML avec des classes Bootstrap au moment de la création de la maquette. Ainsi, nous gagnons du temps sur nos développements grâce à ce code HTML produit automatiquement. Nous vous proposerons les toutes dernières maquettes que nous avons réalisées pour la nouvelle interface avec les couleurs de l'UVSQ.

a) *Interface de connexion*

- Message de bienvenue
- Un champ login
- Un champ mot de passe
- Un bouton 'Connexion' qui nous permet d'aller à l'interface d'accueil s'il le login et le mot de passe sont bons
- Un bouton 'Mot de passe oublié', qui permet d'accéder à l'interface permettant de demander la réinitialisation de son mot de passe :
 - Sur cette interface on peut entrer notre adresse mail. Ainsi, un mail sera envoyé à cette adresse si elle est présente en base de données. Ce mail contiendra un token unique nous permettant d'aller sur l'interface de réinitialisation du mot de passe.



Figure 27 Interface de connexion

b) *Interface du tableau de bord*

- Un lien vers la gestion d'une classe et des professeurs
- Un lien pour envoyer des fiches de présence au CFA
- Un lien pour consulter l'historique des fiches de présence
- Un tableau qui montre les journées de cours dont la fiche de présence n'a pas encore été générée et envoyée au CFA.
 - Les lignes de ce tableau sont cliquables et permettent d'accéder à l'interface montrant le récapitulatif d'une journée de cours



Figure 28 Interface du tableau de bord

c) *Interface d'une journée de cours*

- Un bouton permettant de générer la fiche de présence
- Un diaporama qui permet via des flèches de passer d'un cours à l'autre
- Pour chaque cours :
 - On affiche la matière
 - On affiche le professeur
 - On affiche l'heure de début et de fin
 - On affiche une liste des élèves et de leur statut dans ce cours ('Présent', 'En retard', 'Absent')
 - La police est de couleur rouge si l'élève est 'Absent', de couleur orange si l'élève est 'En retard'
 - Pour chaque cours, un bouton 'Modifier' est présent, nous permettant d'accéder à l'interface de modification. Ce bouton est présent uniquement si la fiche de présence du cours en question n'a pas été générée
 - Une table avec les matières de la journée de cours. Chaque ligne de cette table est un lien cliquable qui permet d'afficher directement les informations du cours en question.

The screenshot displays a web interface for managing course attendance. At the top, a teal header bar contains the text 'James - Gestion de feuille d'émargement' on the left and a 'Menu' link on the right. Below the header is a blue banner with a penguin logo. The main content area has a breadcrumb trail 'home/accueilgestion-abs-fiche-presence'. The course details for 'James' on '24/12/2020' are shown, including the course code 'DSM', professor 'John Doe', and hours '08:00 à 09:30'. A 'MODIFIER' button is present. Below this is a table of students and their attendance status:

| ETUDIANTS | STATUT |
|--------------|---------|
| MARTIN Doe | Absent |
| PHILIPPE Doe | Présent |

Navigation arrows (left and right) are located on either side of the student list. Below the table is a 'GENERER FICHE DE PRESENCE' button. The footer contains logos for 'STARKUP DARTON', 'CGU - Le groupe PUT', and 'UVSQ UNIVERSITÉ PARIS-SACLAY'.

Figure 29 Interface d'une journée de cours

d) *Interface d'une modification d'un cours*

- Deux composants permettant de régler des horaires (valeur par défaut remplie avec les horaires du cours)
- Un bouton permettant d'enregistrer nos modifications
- Un menu déroulant permettant de choisir un professeur parmi les professeurs de la formation du cours et qui ont leur signature rempli (valeur par défaut remplie avec le professeur du cours)
- Un menu déroulant permettant de choisir une matière parmi les matières de la formation (valeur par défaut remplie avec la matière du cours)
- Une table qui sur chaque ligne associe un étudiant à un menu déroulant avec les valeurs 'Présent', 'En retard', et 'Absent', la valeur par défaut de chaque menu déroulant est celle du statut de présence de l'élève durant ce cours

James - Gestion de feuille d'arrangement

Menu

home/accueil/gestion-absence-presence/modification

James, le 24/12/2020

Professeur: John Doe

Matiere: DSM

Heure début: 08 09

Heure fin: 08 09

| ETUDIANTS | STATUT |
|--------------|-----------|
| MARTIN Doe | Absent |
| PHILIPPE Doe | Présent |
| | En retard |
| | Absent |

ENREGISTRER

STARKUP DARTION

UVSQ université PARIS-SACLAY

CGU Le groupe ITUT

Figure 30 Interface de modification d'un cours

e) *Interface d'accueil du site*

- Un lien permettant d'avoir des informations sur le groupe du projet
- Un lien permettant d'accéder au rapport du projet
- Un lien permettant d'accéder au code du projet
- Un lien permettant d'accéder à une présentation de l'application
- Un lien permettant d'accéder à la partie du site qui permet l'administration

- Un bouton qui permet d'accéder à l'interface de connexion

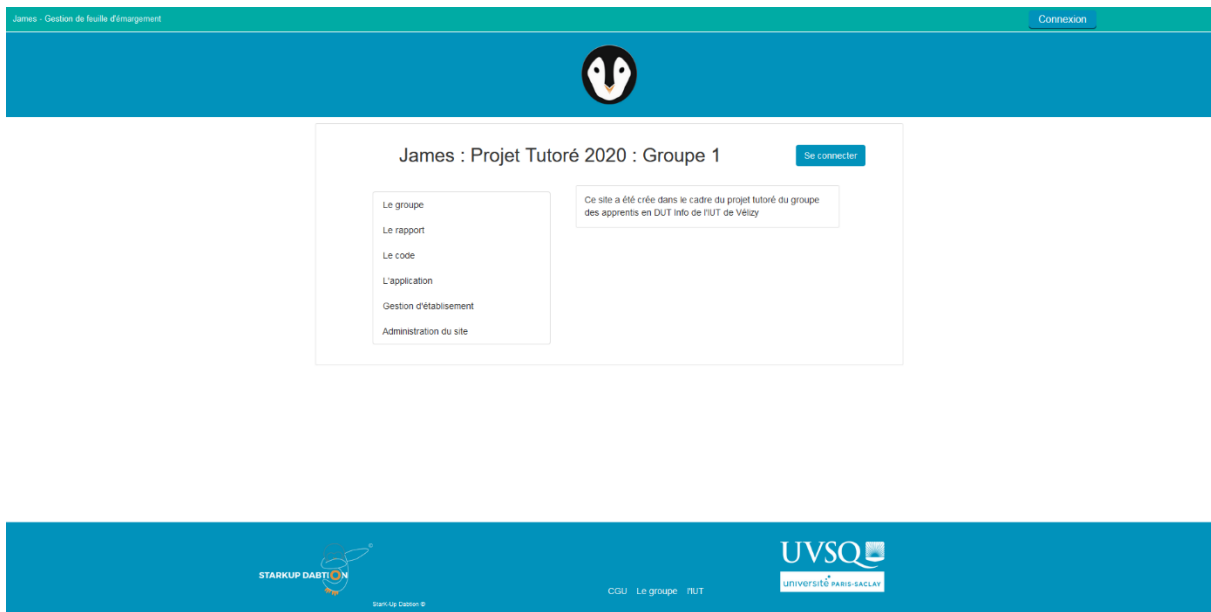


Figure 31 Interface d'accueil

f) *Interface de création d'une formation*

- Champ permettant de rentrer le nom de la formation
- Menu déroulant avec tous les professeurs et qui permet de choisir le professeur responsable de cette formation
- Bouton permettant de valider la création de cette formation

Maquette à venir

Les spécificités peuvent encore changer

g) *Interface de gestion des professeurs dans une formation*

- Liste de professeurs qui ne sont pas dans la formation d'un côté
- Liste de professeurs qui sont dans la formation de l'autre côté
- Système où on peut glisser chaque professeur d'une liste à une autre selon si on veut ajouter un professeur à une formation ou le retirer
- Bouton 'Enregistrer' pour valider les modifications

Maquette à venir

Les spécificités peuvent encore changer

h) Interface d'ajout d'un élève à une formation

- Menu déroulant permettant de choisir sa formation
- Champ 'Nom'
- Champ 'Prenom'
- Menu déroulant groupe, valeur 1 et 2, champ optionnel
- Bouton 'Valider' pour confirmer l'ajout de cet élève

Maquette à venir

Les spécificités peuvent encore changer

i) POPUP ajoutant des étudiants en masse

- Menu déroulant permettant de choisir la formation de tous les étudiants de ce fichier CSV
- Champ permettant d'upload un fichier CSV
- Vérification du respect du format de ce fichier CSV
- Bouton 'Valider' -> si le format est respecté alors on ajoute les étudiants à la base de données

Maquette à venir

Les spécificités peuvent encore changer

Il reste d'autres écrans qui ont été demandées lors de réunion avec Monsieur BARREAU dont les spécificités ne sont pas encore précisées.

4. Cartographie du site web (à la date du 27 février)

La cartographie du site web peut potentiellement être modifiée avec l'ajout de nouvelles pages dont le chemin n'a pas forcément été décidé.

Notre site est accessible à l'adresse : <https://www.stark-up-dabtion.fr/accueil>

| Chemin | Paramètres | Authentification nécessaire | Page associé |
|-----------------------------|--|------------------------------------|---|
| /reset | token (UUID qui doit être présent en base de données dans la table PASSWORD_RESET_TOKEN) | NON | Interface de reset de mot de passe |
| /login | | NON | Interface de connexion |
| /accueil | | OUI | Tableau de bord du site |
| /accueil/historique | | OUI | Interface d'historique des fiches de présences |
| /accueil/gestion-abs | | OUI | Interface montrant les fiches de présences non envoyées |

| | | | |
|--|--|-----|---|
| /accueil/gestion-abs/fiche-presence ou /accueil/historique/fiche-presence | idFormation (l'identifiant de la formation dont on veut les informations) date (date au format américain YYYY-MM-DD, date à laquelle on veut les informations sur la formation) | OUI | Interface de consultation d'une journée de cours pour une formation |
| /accueil/gestion-abs/fiche-presence/modification | idCours (l'id du cours que l'on veut modifier) | OUI | Interface permettant de modifier un cours |

V. Planification

Afin d'essayer d'établir une certaine planification du projet malgré le fait que nous travaillons en SCRUM, nous avons tout d'abord répertorié les différentes tâches du projet que nous avons identifié au début.

A. Tâches du projet et estimation du temps en jour.personne

| Groupement | Tâches | Difficulté US | Temps en jour.personne |
|---------------------------------------|--|---------------|------------------------|
| Planification du projet | Choix des technologies | 7 | 2.5 |
| Planification du projet | Attribution des rôles | 3 | 0.5 |
| Analyse et conception global | Conception très haut niveau du système | 7 | 2 |
| Analyse et conception global | Création d'un premier cahier des charges | 7 | 3 |
| Analyse et conception Android | Création du diagramme d'utilisation | 5 | 1 |
| Analyse et conception Android | Création des spécificités des maquettes | 7 | 3 |
| Analyse et conception WEB Haut niveau | Création du diagramme d'utilisation | 5 | 1 |
| Analyse et conception WEB Haut niveau | Création de la maquette du tableau de bord | 5 | 0.5 |
| Analyse et conception WEB Haut niveau | Création des spécificités des maquettes | 7 | 1 |
| Analyse et conception de la BDD | Conception de la BDD haut niveau | 7 | 3 |

| | | | |
|-----------------------------------|---|----|-----|
| Analyse et conception de la BDD | Analyse des besoins de la BDD | 11 | 3 |
| Gestion du rapport | Ecriture du rapport de projet | 7 | 12 |
| Création de la BDD | Création de la base de données de Tests | 3 | 1 |
| Création de la BDD | Création de la base de données définitive | 3 | 1 |
| Query SQL | Création des cas de tests pour la BDD | 2 | 1 |
| Query SQL | Création des requetes SQL pour les cas de tests | 3 | 1 |
| Infrastructure | Mise en place d'un serveur ALPHA | 17 | 2 |
| Infrastructure | Mise en place d'un environnement de test pour la BDD | 5 | 1 |
| Infrastructure | Préparation de l'environnement de DEV Web | 3 | 1 |
| Module paramètre Android | Création des maquettes des paramètres | 5 | 2 |
| Module paramètre Android | ETQ Prof, je souhaite pouvoir changer les paramètres de l'application | 7 | 4 |
| Module paramètre Android | ETQ Prof, je souhaite changer la durée d'un cours par défaut | ? | 2 |
| Module paramètre Android | ETQ Prof, je souhaite changer mon adresse mail | ? | 1 |
| Module d'authentification Android | ETQ PO, je souhaite qu'un cryptage du mot de passe soit crée afin de protéger les données | 2 | 0.5 |
| Module d'authentification Android | Création d'une interface de connexion | 11 | 3 |
| Module d'authentification Android | ETQ Prof, je souhaite pouvoir demander une réinitialisation de mon MDP | 11 | 2 |
| Module d'authentification Android | ETQ PO, je souhaite que l'API renvoie les informations du professeur | 2 | 1 |
| Module d'authentification Android | ETQ Prof, je souhaite pouvoir changer mon MDP | 13 | 1 |
| Module d'authentification Android | ETQ PO, je souhaite la modernisation de l'interface de connexion | 5 | 1 |
| Module d'authentification Web | Création d'une interface de demande de réinitialisation du MDP | 13 | 2 |

| | | | |
|-------------------------------|---|----|-----|
| Module d'authentification Web | ETQ PO, je souhaite qu'un cryptage du mot de passe soit crée afin de protéger les données | 2 | 0.5 |
| Module d'authentification Web | Création d'une interface de connexion WEB | 7 | 3 |
| Module d'authentification Web | ETQ PO, je souhaite que l'API de connexion renvoie l'ID du secrétaire | 2 | 1 |
| Module d'authentification Web | Création de l'interface de réinitialisation du MDP | 5 | 2 |
| Module générique Android | ETQ Tuteur-Projet, je souhaite que les couleurs de l'UVSQ soit appliquées sur l'application | 2 | 0.5 |
| Module générique Android | ETQ Professeur je souhaite pouvoir accéder à un tableau de bord | 5 | 1 |
| Module générique Android | ETQ Professeur, je souhaite pouvoir rentrer ma signature | 5 | 1 |
| Module d'absences Android | ETQ Prof, je souhaite pouvoir sélectionner un groupe et un cours afin de l'émarger | 20 | 3 |
| Module d'absences Android | ETQ Elève, je souhaite pouvoir rentrer ma signature | 5 | 3 |
| Module d'absences Android | ETQ Prof, je souhaite pouvoir envoyer mon cours émargé à la secrétaire | 5 | 1 |
| Module d'absences Android | ETQ Professeur, je souhaite qu'un élève soit notifié quand il est absent | ? | 3 |
| Module d'absences Android | ETQ Professeur je souhaite pouvoir avoir la dernière durée inscrite pour un cours donné | ? | 2 |
| Module d'absences web | ETQ Secrétaire, je souhaite pouvoir accéder à la liste des fiches non traitées | 7 | 1 |
| Module d'absences web | ETQ Secrétaire, je souhaite pouvoir voir le détail d'une fiche | 5 | 3 |
| Module d'absences web | ETQ Secrétaire, je souhaite pouvoir modifier une valeur d'une fiche | 11 | 3 |
| Module d'absences web | ETQ Secrétaire, je souhaite pouvoir fusionner deux fiches de deux groupes différents | ? | 4 |
| Module d'historique web | ETQ Secrétaire, je souhaite accéder à l'historique de mes fiches | 5 | 2 |
| Module d'historique web | ETQ Secrétaire, je souhaite pouvoir télécharger mes fiches | 7 | 3 |

B. Elaboration d'une planification

Comme nous sommes en SCRUM, il est bien sûr difficile de faire un graphe d'ordonnancement car nous sommes censés pouvoir passer d'une tâche à une autre sans dépendre de tâches antérieures. Afin de réaliser cet exercice, nous avons donc décidé de définir une antériorité selon l'ordre dans lequel nous avons pour l'instant fait les tâches mais cette ordre aurait pu être très différent notamment sur les tâches liées à la programmation (H, I par exemple).

Il est à noter que les durées sont arrondies au point supérieur 2.5 devient donc 3. Pour ce qui concerne du rapport, il ne peut être commencé qu'une fois une bonne partie de la conception réalisée car une partie de ce dernier est dédiée à la conception. Cependant il ne peut réellement être réalisée entièrement que quand le projet est totalement fini. Il est donc difficile de le placer sur un graphe d'ordonnancement mais nous avons décidé de le placer derrière la phase de conception.

| Lettre | Groupement | Tache antérieure | Durée |
|--------|---------------------------------------|------------------|-------|
| A | Planification du projet | | 3 |
| B | Analyse et conception global | A | 5 |
| C | Analyse et conception Android | A | 4 |
| D | Analyse et conception WEB Haut niveau | A | 3 |
| E | Analyse et conception de la BDD | A | 6 |
| F | Création de la BDD | E | 2 |
| G | Query SQL | F | 2 |
| H | Module d'authentification Android | C,G,B | 9 |
| I | Module d'authentification Web | D,G,B | 9 |
| J | Gestion du rapport | A | 12 |
| K | Module d'absences Android | H | 12 |
| L | Module d'absences web | I | 11 |
| M | Module générique Android | K | 3 |
| N | Module d'historique web | L | 5 |
| O | Infrastructure | B,C,D,G | 4 |

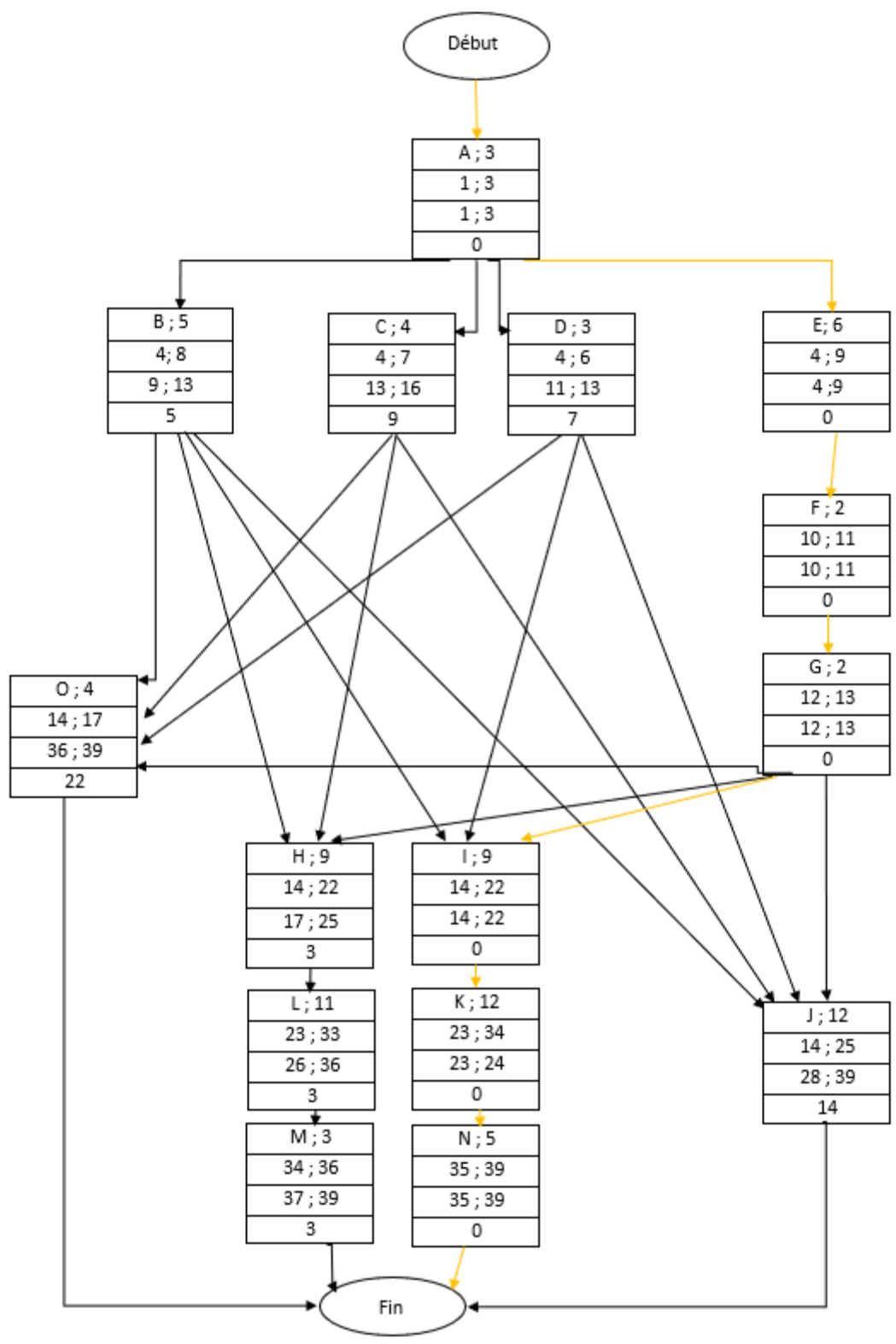


Figure 32 Graphe d'ordonnement du projet

Rappel de la légende

| |
|---|
| Lettre Tâche ; Durée Tâche |
| Date de début au plus tôt ; Date de fin au plus tôt |
| Date de début au plus tard ; Date de fin au plus tard |
| Marge sur la tâche |

Les flèches orange forment le chemin critique.

D'après le graphe d'ordonnancement, nous serions à une durée de 39 jours.homme afin de mener à bien ce projet. On voit qu'il n'y a pas plus de 4 tâches se déroulant en même temps et nous sommes 5 membres, pour réduire au maximum la durée du projet, il suffit de mettre à chaque fois 2 personnes sur la tâche critique et une personne sur les autres tâches.

Quand il n'y a qu'une seule tâche alors tout le monde se concentre dessus.

Ce qui fait qu'au final la durée des tâches critiques se retrouvent divisées par 2 lorsqu'il y a 3 autres tâches en cours et est divisé par 5 lorsque c'est la seule tâche.

On va donc faire un diagramme de Gantt avec ce que cela pourrait donner même si en pratique le scrum ne fait pas de réelle planification.

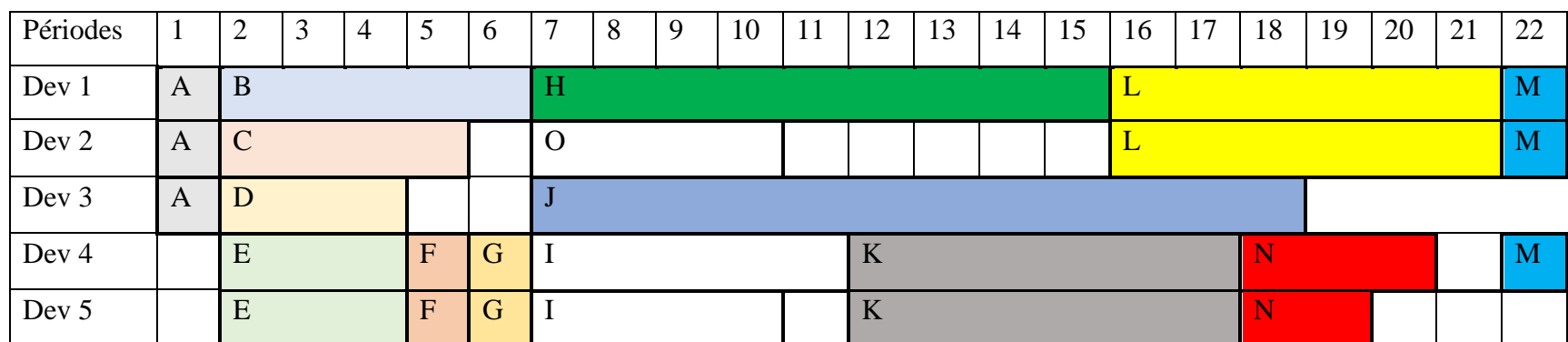


Figure 33 Diagramme de Gantt prévisionnel

On voit donc qu'avec un rythme de travail de type entreprise mais surtout en faisant en sorte que deux personnes se concentrent tout le temps sur le chemin critique on se retrouve avec un projet qui se termine en 22 jours au final. Cela semble un résultat réaliste on se retrouve à un peu plus de 4 semaines de travail à plein temps sur ce projet afin de le boucler.

Dans notre cas actuel, on peut estimer que le temps disponible pour travailler chaque semaine sur le projet est assez variable. En moyenne, on peut estimer que chaque étudiant peut consacrer environ 1 à 2 journées pour deux semaines selon le nombre de contrôles.

Depuis que nous avons commencé le 19 octobre, il s'est écoulé un peu plus de 14 semaines donc chaque membre devrait selon nos chiffres s'investir environ 7 à 14 jours sur le projet soit pour faire l'entre deux 11 jours. Si on se réfère au diagramme de Gantt au dessus alors la phase « Conception » devrait être totalement terminée ce qui est bien le cas puisque les tâches A, B, C, D, E, F, G sont terminées. L'infrastructure devrait être mise en place ce qui est le cas. La tâche I est également terminée puisque l'interface de connexion WEB est terminée. De l'avance a été prise puisque la tâche H est quasiment terminée et la tâche L est dans un état particulièrement avancé. Nous sommes donc à un peu plus de la moitié de ce projet et nous sommes en avance sur ce qui était prévu. On pourrait tabler sur la fin du projet vers fin mars pour la fin du semestre 3. Cependant, il arrive que des tâches viennent se rajouter au fil des démonstrations et peut être des oubliées sur certains aspects du projet. Donc cela pourrait allonger la durée du projet mais une équipe agile est bien évidemment prête à tous les obstacles. Par exemple, il a été décidé que les étudiants devraient recevoir une notification lorsqu'ils sont marqués absents donc la tâche H a dû être reprise.

VI. Analyse de risques

A. Identification des risques

- **Indisponibilité de membres de l'équipe** (cours/contrôles, concours, maladies, problèmes personnels, etc)
- Des **technologies pas forcément maîtrisées** par l'équipe
- **Problèmes de planification** via Jira
- **Non-respect du RGPD**
- **Insécurité des produits du projet** (piratage possible par exemple, ou autre)
- Délais dépassés, **retards**
- **Fuite des clés de cryptage**
- **Cahier des charges** pouvant être **peu clair**
- **Mauvaise expression des besoins du client**
- Bombe nucléaire
- Risque de pandémie, et donc **risque de distanciel**
- **Mauvaise gestion des sprints**
- **Conflits internes**
- **Changement des spécifications** de l'application
- **Problèmes d'infrastructure** (PC perso, serveur)
- **Perte du code source**
- **Problèmes logiciels** (déjà arrivé avec Xampp)
- **Problèmes de fonctionnement de Git**
- **Bug dans l'application en production**
- **Perte de la base de données**

L'identification de ces risques s'est faite par une méthode très simple :

En effet, nous nous sommes laissés un jour complet (environ 20h) afin de **lister chacun de notre côté** quelques problèmes possibles (**brainstorming**). Nous avons ensuite **réuni ces différentes listes individuelles en une seule liste commune** ; nous avons éliminé les doublons, et avons ajouté des risques supplémentaires suite à **une concertation**. Ainsi, un travail individuel associé à une simple concertation entre les membres de l'équipe du projet tuteuré auront suffi à établir une liste commune composée de risques tous différents. Pour un peu plus de détail sur le travail durant la concertation : nous avons procédé par étapes, d'abord les risques qui, s'ils arrivent, seraient les plus graves, tels que l'arrêt de Git, les problèmes d'infrastructure...Bombe nucléaire ? Suivis des risques qui seraient moins graves, etc. Nous avons donc procédé par ordre décroissant des impacts de ces risques si jamais ils arrivaient. Au final, nous avons regroupé ce que nous pensions similaire dans une même « catégorie », sous un même nom. Nous nous retrouvons donc avec exactement **vingt risques** repérés par l'équipe.

B. Echelle de probabilité d'occurrence d'un risque

| | |
|-------------------|--|
| Très probable | Risque qui a de grandes chances d'arriver |
| Plutôt probable | Risque qui a des chances d'arriver |
| Plutôt improbable | Risque qui a assez peu de chance d'arriver |
| Peu probable | Risque qui n'a que très peu de chance d'arriver, qui serait presque impossible en temps normal |

C. Echelle de l'impact d'un risque sur le projet

| | |
|-----------------------------|---|
| Gros impact sur le projet | Risque qui va perturber tout le projet et pourrait causer de gros dégâts (que ce soit sur les produits, le code, la planification...) |
| Impact modéré sur le projet | Risque surmontable si on y est préparé mais dont il faut tout de même y faire attention |
| Peu d'impact sur le projet | Risque à considérer mais qui est surmontable même si nous n'y sommes pas préparés |
| Aucun impact sur le projet | Risque que l'on pourrait presque ignorer tant l'impact est moindre |

D. Matrice de l'impact du risque sur le projet en fonction de la probabilité du risque

| | | | | |
|-----------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| Gros impact | Seuil de risque acceptable | Risque inacceptable | Risque inacceptable | Risque inacceptable |
| Impact modéré | Risque acceptable | Seuil de risque acceptable | Risque inacceptable | Risque inacceptable |
| Peu d'impact | Risque acceptable | Risque acceptable | Seuil de risque acceptable | Risque inacceptable |
| Aucun impact | Risque acceptable | Risque acceptable | Risque acceptable | Seuil de risque acceptable |
| Impact Probabilité | Peu probable | Plutôt improbable | Plutôt Probable | Très probable |

E. Tableau associant un risque à sa probabilité

| Numéro risque | Risque | Facteur risque | Probabilité | Impact |
|---------------|---|---|------------------------------------|-------------|
| 1 | Indisponibilité de membres de l'équipe | Cours/contrôles, concours, maladies, problèmes personnels, etc. | Plutôt probable | Modéré |
| 2 | Des technologies pas forcément maîtrisées par l'équipe | La découverte de nouvelles technologies telles qu'Android, ou Angular pour le web, que tout le monde n'a pas l'occasion de voir en entreprise | Très probable (au début du projet) | Gros Impact |
| 3 | Problèmes de planification via Jira | Des modifications pas prises en compte, erreurs humaines (« miss-click » par exemple, ou mauvaise lecture, ou vraie erreur dans la prévision de la planification) | Plutôt improbable | Gros Impact |
| 4 | Non-respect du RGPD | Une méconnaissance des points importants du RGPD | Plutôt improbable | Gros Impact |
| 5 | Insécurité des produits du projet | Des individus voulant s'introduire dans l'infrastructure du projet ou bien qui veulent récupérer des données éventuellement laissées sans protection | Plutôt Improbable | Gros Impact |
| 6 | Délais dépassés, retards | Association d'une vie étudiante et d'une méthode de gestion de projet qui requiert d'être totalement investi | Très probable | Modéré |
| 7 | Fuite des clés de cryptage | Un code qu'on pourrait partager avec nos camarades sans le faire exprès, qu'on pourrait garder sur notre clé USB et l'oublier à l'IUT... | Peu probable | Modéré |
| 8 | Cahier des charges pouvant être peu clair | Le rédacteur du cahier des charges n'est pas assez précis, ce qui nuit à la compréhension du problème par les développeurs | Plutôt probable | Gros Impact |
| 9 | Mauvaise expression des besoins du client | Les clients qui peuvent ne pas forcément être clairs ou assez précis sur leurs besoins | Plutôt probable | Gros Impact |

| | | | | |
|----|---|--|--|--------------|
| 10 | Bombe nucléaire | Certains pays pas totalement amicaux avec la France sont en possession de cette arme | Peu probable | Gros Impact |
| 11 | Risque de pandémie, et donc risque de distanciel | Explication de notions plus compliquées à distance, motivation en baisse, communication plus difficile | Très probable (si on se place à aujourd'hui), et Plutôt improbable si on se place à avant le premier confinement | Peu d'impact |
| 12 | Mauvaise gestion des sprints | Mauvais « calculs » (prévisions, estimations) dans la durée notamment pour notre groupe, ou alors au niveau du nombre de user stories à mettre dans un sprint, ce qui peut mener à une faible vélocité | Plutôt probable | Gros Impact |
| 13 | Conflits internes | Planning chargé, fatigue...peuvent mener à parfois des tensions qui pourraient dégénérer | Plutôt probable | Modéré |
| 14 | Perte du code source | Incendie qui brûle tout notre matériel, accidents... | Très improbable | Gros Impact |
| 15 | Changement des spécifications de l'application | Le client veut un produit avec des caractéristiques différentes de ce qu'il avait précisé au départ, il faut donc changer le produit | Très probable | Aucun |
| 16 | Problèmes sur l'infrastructure (PC perso, serveur) | difficulté d'un membre à avoir les installations requises pour coder sur le projet | Plutôt improbable | Gros Impact |
| 17 | Problèmes logiciels (déjà arrivé avec Xampp) | Mauvaises configurations, bugs... | Très probable | Modéré |
| 18 | Problèmes de fonctionnement de Git | Support de versionning qui tombe en panne (dans notre cas le service Atlassian (plus précisément Jira)) | Peu probable | Gros Impact |
| 19 | Perte de la base de données | Une mauvaise manipulation sur ce qui est en production, un problème chez l'hébergeur... | Peu probable | Gros Impact |
| 20 | Bug dans l'application en production | Lors d'une nouvelle fonctionnalité, cette dernière comporte un bug non-perçu par le développeur mais est quand même ajouté au produit final | Plutôt probable | Gros Impact |

| | | | | |
|----------------------|--|---|--|---|
| Gros impact | Bombe nucléaire ; Perte du code source; Problèmes de fonctionnement de Git ; Perte des données de la base de données; | Problèmes de planification avec Jira ; Insécurité des produits du projet ; Problèmes sur l'infrastructure ; | Cahier des charges peu clair ; Mauvaises expressions des besoins du client ; Mauvaise gestion des sprints; Bug dans l'application en production | Technologies pas forcément maîtrisées par les membres de l'équipe |
| Impact modéré | Fuite des clés de cryptage ; | | Indisponibilité de membres de l'équipe ; Conflits internes ; | Délais dépassés, retards par rapport au planning prévu; Problèmes logiciels; |
| Peu d'impact | | | | Risque de pandémie, et donc risque de distanciel ; |
| Aucun impact | | Non-respect du RGPD | | Changement des spécifications de l'application ; |
| Impact / Probabilité | Peu probable | Plutôt improbable | Plutôt Probable | Très probable |

F. Tableau associant un risque à une stratégie de réponse et un point de situation de notre projet

| Risque | Stratégie | Action | Point situation |
|--|-----------|---|---|
| Indisponibilité des membres de l'équipe | Atténué | Chaque membre qui pense être absent durant une certaine période l'annonce aux autres en avance afin de pouvoir s'y préparer et pouvoir tenir le rythme du sprint avec le/s membres en moins | Ce risque a déjà eu lieu de nombreuses fois et cela n'a pas causé du retard sur le projet |

| | | | |
|---|--|--|---|
| Des technologies pas forcément maîtrisées par l'équipe | Atténué | Apprentissage via internet dès que le membre travaille sur une partie dont il ne maîtrise pas la pratique, ou grâce aux camarades de notre groupe qui ont été nommés « responsables » d'une technologie utilisée. | Déjà eu lieu mais les responsables technologies ont accompagné des non-érudits |
| Problèmes de planification | Evité | Plusieurs membres qui suivent en même temps les changements effectués, mais aussi la mise en place d'un système automatique sur Jira qui envoie des mails lorsque des changements qui nous concernent ont été réalisés | Jamais eu lieu |
| Non-respect du RGPD | Evité | Grâce à notre cours de Droit des TIC, nous connaissons les points importants des différentes réglementations et les appliquons rigoureusement au projet | Nous respectons les règles de la RGPD actuellement (peu de données personnelles par exemple) |
| Insécurité des produits du projet | Evité | Applications sécurisées avec utilisation d'une clé de cryptage assez longue, qui, même associée à des mots de passe éventuellement volés, ne peuvent être traduits en messages clairs par des logiciels spécialisés (nous avons testé) | Les mots de passe sont bien chiffrés (MD5) dans la base de données avec une clé assez longue pour ne pas être déchiffrée facilement |
| Délais dépassés, retards | Acceptation active | Redynamisation de l'équipe, concertations, rappel des tâches à réaliser, etc. | Nous avons de l'avance sur la date de rendu du projet |
| Fuite des clés de cryptage | Elaboration d'une réponse conditionnelle | Changement de la clé de cryptage | Le code n'est accessible qu'aux membres du projet |
| Cahier des charges pouvant être peu clair | Evité | Ce risque est évité puisque chaque développeur aurait pu rédiger ce cahier des charges, ce qui permet de toute manière d'avoir une compréhension parfaite du projet | Bonne compréhension des spécifications du projet pour le moment par tous les membres |
| Mauvaise expression des besoins du client | Atténué | Questionnement du client, concertation mensuelle/bimensuelle avec le client | Déjà arrivé mais grâce aux réunions bimensuelles avec M. Huguin et M. Barreau alors les besoins ont été plus |

| | | | |
|---|--|--|--|
| | | | détaillés ce qui nous a aidé à satisfaire au mieux les clients. |
| Bombe nucléaire | Acceptation passive | Le projet passe au plan secondaire | Aucune action |
| Risque de pandémie, et donc risque de distanciel | Elaboration d'une réponse conditionnelle | Adaptation avec par exemple la création d'un groupe privé Discord très bien organisé où chacun peut discuter d'un sujet avec le reste de l'équipe d'un seul coup, regroupement dans un salon vocal de toute (ou presque toute) l'équipe assez régulièrement. De même avec les sprints retrospectives, adaptées pour le distanciel avec des sites internet tels que https://easyretro.io , etc. | L'action prévue a été totalement mise en place ce qui nous permet de communiquer et travailler efficacement même à distance. |
| Mauvaise gestion des sprints | Atténué | Déterminer la vitesse moyenne de l'équipe durant les deux-trois premiers sprints afin de ne plus embarquer trop d'US | Déjà arrivé, notamment des sprints trop ambitieux en terme de vitesse car ils ne prenaient pas assez en compte les indisponibilités de chacun. La vitesse a donc été diminuée pour mieux correspondre à notre vie d'étudiants. |
| Conflits internes | Elaboration d'une réponse conditionnelle | Pause de quelques jours, ou intervention d'autres membres de l'équipe afin de calmer les choses | Jamais produit pour le moment |
| Problèmes d'infrastructure | Evité | Un camarade du groupe vient l'aider afin de régler le problème (installation par exemple) | On a respecté l'action prévue grâce à Discord et les partages d'écran. |
| Changement des spécifications de l'application | Acceptation passive | Philosophie agile (SCRUM), nous sommes constamment prêts à modifier les produits | Déjà produit (par exemple le fait que le professeur puisse rentrer sa signature dans l'application mobile) ; ce n'a jamais posé un quelconque soucis. |
| Perte du code source | Evité | Nous avons des parties de back-up dans chacun de nos | Ce n'est jamais arrivé car nous |

| | | | |
|---|--|---|--|
| | | ordinateurs, mais nous avons également l'utilitaire Git qui permet de retrouver la très grande majorité des codes sources (avec les différentes versions du code) sur internet grâce au versionning. (Bitbucket dans notre cas) | avons suivi l'action prévue. |
| Problèmes logiciels | Elaboration d'une réponse conditionnelle | Réinstallations, recherches sur internet, explications par un camarade si jamais ça lui est arrivé... | Déjà arrivé notamment avec le logiciel Xampp et nous avons du fabriquer une réponse sur le tas (réinstallation dans le cas de Xampp pour certains) |
| Problèmes de fonctionnement de Git | Elaboration d'une réponse conditionnelle | Ou bien nous changerions de support de versionning, ou bien nous ferions « comme à l'époque », avec par exemple des transferts de fichiers entre membres de l'équipe, par exemple avec des clés USB, ou par exemple de nos jours via le cloud sur internet. | Cela n'a pas encore eu lieu et heureusement car cela rendrait le travail collaboratif à distance plus difficile |
| Bug dans l'application en production | Atténué | Système de recette : un autre développeur regarde le travail de celui qui a travaillé sur la fonctionnalité afin de déceler le bug et l'avertir | Malgré l'action que nous avons suivie, il est déjà arrivé qu'un bug soit dans l'application mobile en production. Nous avons du corriger ce bug dans l'urgence absolue pour compenser notre manque de rigueur. |
| Perte de la base de données | Evité | Nous avons un Back-up (script SQL de création de la BD) dans chacun de nos ordinateurs personnels. | Chacun a bel et bien un script SQL en local. Des pertes de la base de données en local pour certains ont déjà eu lieu et ce script a évité de perdre du temps. |

VII. Plan de communication

| Phase 1: Préparation et lancement du projet | | | | | | | |
|--|-------------------------------------|--------------|-----------------------|---|-------------------------------|--------------------------------------|-------------------------------------|
| Objectif de communication | Action | Date début | Groupe cible | Message | Canal | Emetteur | Fréquence |
| Faire connaître les objectifs du projet | Annoncer le lancement du projet | 29 septembre | Les membres du groupe | Remise du sujet du projet qui consiste en la création d'une application mobile et web qui vont être reliées par un serveur et qui vont permettre de suivre les présences dans la formation DUT INFA-2 | Lors d'un cours en présentiel | Fabrice HOGUIN | 1 fois |
| Phase 2: Réalisation du projet (conception et programmation) | | | | | | | |
| Objectif de communication | Action | Date début | Groupe cible | Message | Canal | Emetteur | Fréquence |
| Faire part de l'avancement du projet | Organisation de Daily Scrum Meeting | 19 octobre | Les membres du groupe | Communiquer sur les tâches que l'on fait quotidiennement | En présentiel ou sur Discord | Tous les membres du groupe (Aurelien | 1 fois par jour en semaine scolaire |

| | | | | | | | |
|--|--|-------------------|-------------------------------------|---|--|--|--|
| | | | | | | organise la réunion) | |
| Avoir des retours clients et/ou faire part de l'avancement du projet | Démonstration de l'application aux professeurs | 13 novembre | Monsieur BARREAU et Monsieur HOGUIN | Le projet à avancer et nous aimerions avoir des retours sur ce que nous avons déjà suite à ce que l'on vous a montré | Sur ZOOM ou en présentiel avec support diaporama pour la présentation officielle du S3 | Aurélien qui organise la réunion et la démonstration | Environ 1 fois par mois |
| | Démonstration de l'application à MMe Girard | 26 mars | Madame Girard | Montrer le projet afin que Mme Girard nous fasse remonter des idées d'amélioration pour l'application qui lui sera dédiée | sur ZOOM ou présentiel si possible | Aurélien qui organise la réunion et la démonstration | 2 fois avec deux mois d'écarts |
| Faire le point sur l'état de l'équipe et du projet | Organisation d'une sprint retrospective | 2 novembre | Les membres du groupe | Communiquer sur les ressentis de chaque membre de l'équipe et essayer de s'améliorer en vue de la suite du projet | Sur Discord et avec FunRetro.com par exemple | Tous les membres du groupe avec Aurélien qui organise la réunion | A chaque fin de sprint, toutes les 2 ou 3 semaines |
| Phase 3: Faire la promotion du projet | | | | | | | |
| Objectif de communication | Action | Date début | Groupe cible | Message | Canal | Emetteur | Fréquence |

| | | | | | | | |
|--|--|-----------------------|---|---|-------------------------------------|----------------------------|--------|
| Faire la “publicité” du projet une fois fini | Organisation d’une présentation du fonctionnement de l’application | date provisoire: Juin | Directeur de l’IUT, professeurs DUT Info, Madame Girard | Présenter l’application et la manière dont on s’en sert correctement. | En présentiel à l’IUT (on l’espère) | Tous les membres du groupe | 1 fois |
|--|--|-----------------------|---|---|-------------------------------------|----------------------------|--------|

VIII. Annexes

A. Vocabulaire

- **Out of Scope** -> Fonctionnalité à voir plus tard ou qui seront changer dans le futur
- **Backlog** -> Liste ordonnancée des besoins, généralement formulés sous forme de User story
- **Epic** -> Tâches importantes qui peuvent être subdivisées en plus petites tâches (appelées stories)
- **Sprint** -> Intervalle de temps court, pendant lequel la Dev Team va concevoir, réaliser et tester de nouvelles fonctionnalités formalisées sous forme de **User Story**
- **[US] (User) Story** -> Technique permettant de formaliser synthétiquement les besoins sans perdre de vue l'essentiel : le besoin concerne QUI, en QUOI il consiste et dans quel BUT
- **[DSM] Daily Scrum Meeting** -> Point quotidien, rapide et efficace sur les sujets en cours.