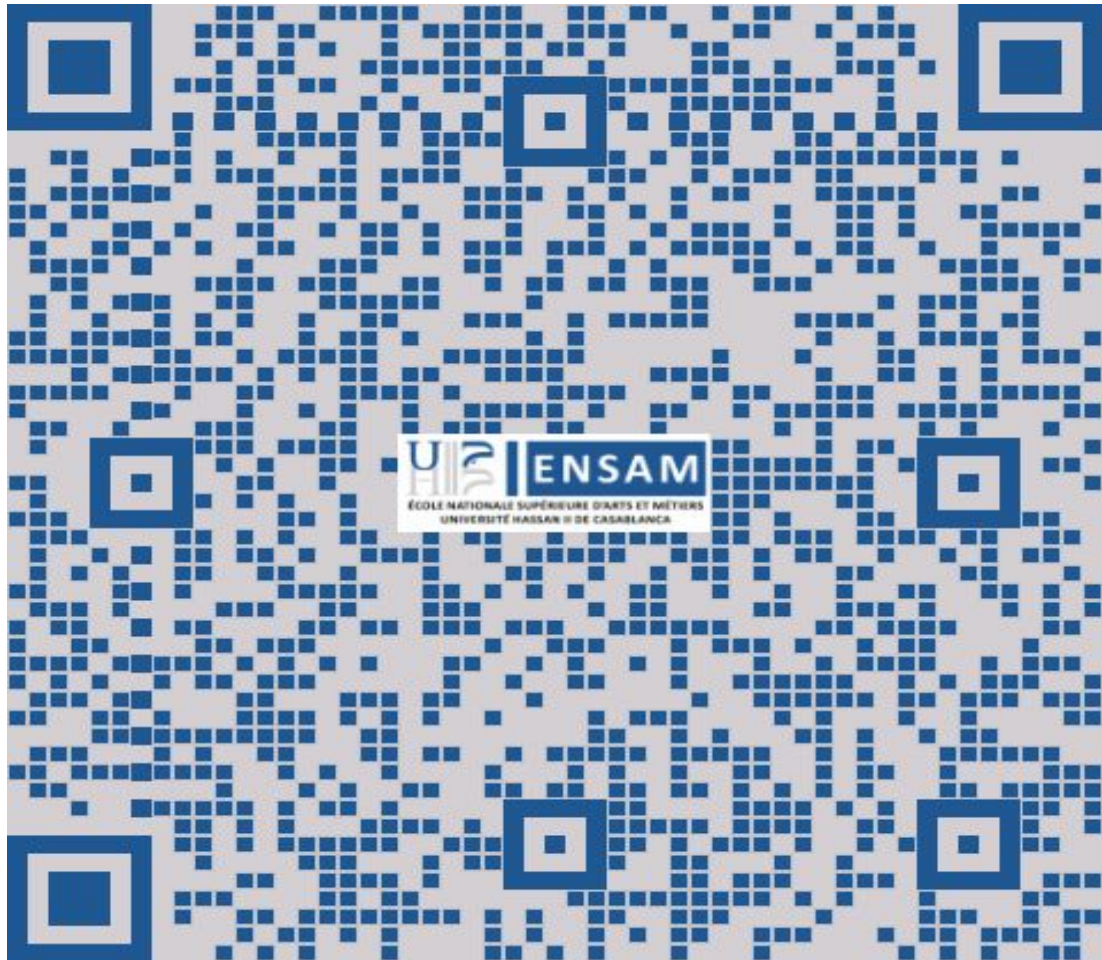


# Rapport de projet

## QrAttendance



Auteurs : BARJ Ali, NABIL Mohamed et QASBAJI Zakaria





# Remerciements

Nous tenons premièrement à remercier l'école nationale supérieure d'arts et métiers de Casablanca pour nous avoir offert cette opportunité. Dans une période de confinement très dure où la majorité des entreprises ont fermé leurs portes de recrutement, l'administration de l'ENSAM de Casablanca a fait preuve d'une très grande flexibilité vis-à-vis des stages de fin d'année, en offrant à ses étudiants l'alternative de réaliser des projets à vocation industrielle.

Nous remercions également nos chers professeurs qui ont veillé sur la qualité de la formation même en période extrêmement délicate. Leur générosité à nous enseigner leurs maîtrises et compétences nous a été d'une immense aide lors de la réalisation de ce projet.

# Table des matières

<b>Remerciements</b>	<b>3</b>
<b>Introduction</b>	<b>5</b>
<b>Description du projet</b>	<b>7</b>
Les QR Codes	7
Utilisations du produit et fonctionnalités	9
Parties prenantes	11
<b>Design et architecture</b>	<b>12</b>
Architecture logicielle	13
Bibliothèques utilisées	15
Mise en oeuvre et structure logicielle	20
Diagramme de cas d'utilisation	20
Diagramme de séquences	29
Diagramme d'activité	34
Base de données	35
Modèle conceptuel de données	35
Modèle logique de données	37
Technologies	37
<b>Exigences (Requirements)</b>	<b>40</b>
Exigences matérielles	40
Exigences fonctionnelles	40
Exigences de performance (performance requirements)	41
<b>Conclusion</b>	<b>43</b>

# Introduction

Le projet QrAttendancy est une solution logicielle innovante dédiée au problème de la gestion d'absence des étudiants, qui prend la forme d'une application web à usage interne basée sur la technologie des QR Codes (Quick Response Codes). Développée par Qasbaji Zakaria, Barj Ali et Nabil Mohamed, trois étudiants de l'école nationale supérieure d'arts et métiers de Casablanca (ENSAM), filière intelligence artificielle et génie informatique, l'application vient assister les professeurs en une tâche pouvant s'avérer fastidieuse : le contrôle des présences.

La prise de présence des étudiants est une étape majeure dans le processus d'apprentissage. Notre école, comme la plupart des autres établissements d'enseignements, lui assigne un très haut degré d'importance. Le taux de présence offre aux professeurs une vue générale sur l'engagement des étudiants en plus d'un mode additionnel d'évaluation en fonction de cet engagement.

La méthode traditionnelle de prise de présence en classe est le passage d'une liste de présence auprès des étudiants pour la signer. Ce procédé est sujet à la fraude, parce que rien n'empêche un étudiant de signer à la place de ses camarades absents, cela affecte l'intégrité du processus et peut même inciter à des conflits professeurs-étudiants. Une alternative est l'appel, où le professeur crie le nom des étudiants un par un pour marquer leur présence dans un registre . Certes, cela élimine le risque de fraude, mais ça puise dans l'énergie des professeurs et la durée des séances. Dans les deux cas, le professeur se retrouve avec un très grand nombre de fiches de présence à trier et à évaluer en fin de compte.

L'ENSAM Casablanca a depuis toujours exprimé son intention de trouver des alternatives innovantes aux tâches manuelles et récurrentes. Notamment, l'un des axes discutés durant le conseil annuel de l'établissement du 08/07/2019 était la recherche d'une solution technologique au problème de la gestion d'absence. Nous avons donc pris l'initiative de développer une application intitulée QrAttendancy mettant en pratique nos acquis cumulés durant la première année en cycle d'ingénieur comme projet de fin d'année.

Nous aspirons offrir aux professeurs un outil pratique et sécurisé, qui permettra l'automatisation du procédé de la prise de présence, la digitalisation des données et un accès facile aux informations relatives à l'absence et pertinentes à l'évaluation.

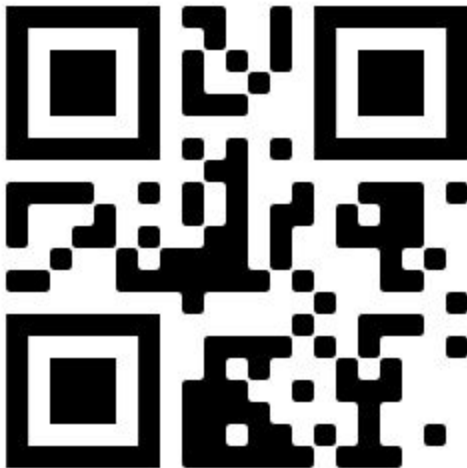
Le premier chapitre de ce rapport discute la technologie de QR Codes, ainsi qu'une description du projet, ses fonctionnalités et ses parties prenantes. Le deuxième chapitre détaille l'architecture du logiciel, les technologies utilisées et la conception de base de données. Le troisième et dernier chapitre présente une liste d'exigences pour le logiciel.

# 1. Description du projet

## 1.1. Les QR Codes

### Définition:

Le QR Code est une figure carrée, composée d'une multitude de petits modules colorés, qui codent des données. Différentes dimensions sont disponibles permettant des capacités de données adaptées. On parle de code-barres en 2D, de code-barres bidimensionnels ou de code matrice 2D.



Les données contenues dans un QR Code sont déchiffrables par une caméra et un logiciel spécifique. La plupart du temps, les caméras sont les webcams et appareils photo/caméra des téléphones portables de type smartphone, des tablettes tactiles ou des pc portables. Les logiciels et applications (ou Apps) sont des "lecteurs" ou "scanners" disponibles sur les plateformes de téléchargement mobile (Android Market ou Play Store, App Store, ...).

Les possibilités d'utilisations sont diverses. Le QR Code permet de se connecter à une page web, de déclencher un appel téléphonique, d'afficher une simple information textuelle, etc. En termes d'utilisation, les QR Codes ouvrent un moyen de communication supplémentaire apportant une expérience enrichie à l'utilisateur. Leur capacité à jouer le rôle de passerelle entre le physique et le numérique facilite l'accès aux informations et services, sans avoir à saisir de longues adresses URL par exemple et dans un délai immédiat. Il est à relever que générer un QR Code et le mettre à disposition a un coût égal ou proche de 0. En termes d'apparences, outre leur aspect mystérieux, ils transmettent une image moderne et positive.

### Historique:

La technologie du QR Code a été créée en 1994 par Denso-Wave, une société japonaise, pour assurer le suivi des pièces automobiles de la marque Toyota. Cinq ans plus tard, la société créatrice du code le publie en licence libre, ce qui a contribué à son expansion au Japon, puis dans le monde, au travers du marketing et de la publicité, permettant ainsi une interactivité nouvelle aux personnes. Les japonais l'ont adopté depuis longtemps. Ils scannent plus de QR Code qu'ils n'envoient de SMS.

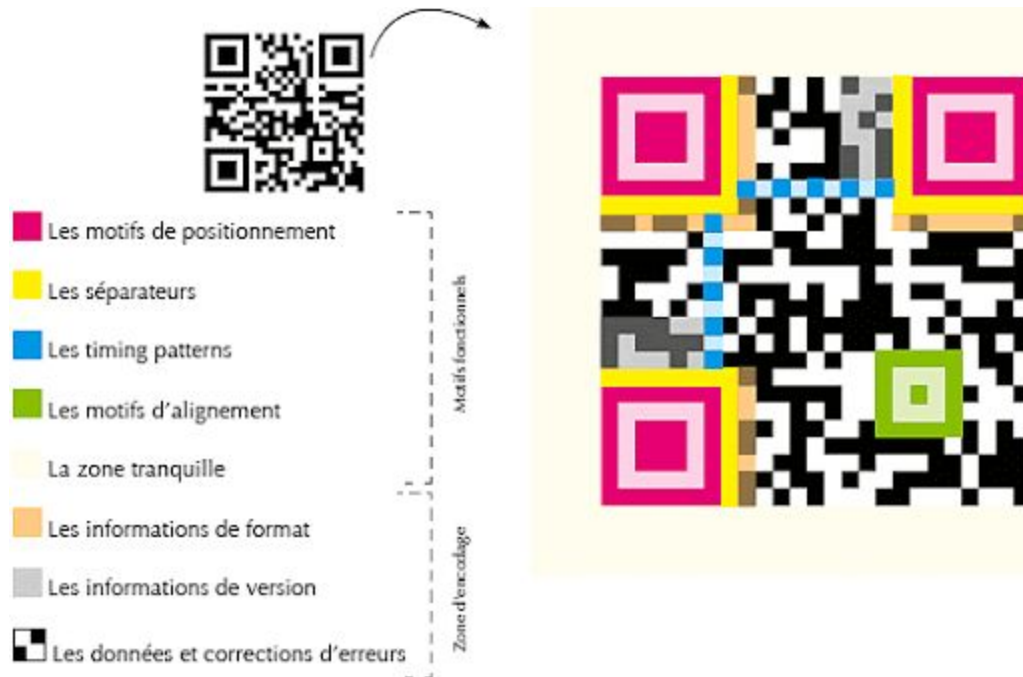
### **Technologie:**

Les lettres QR sont les initiales de Quick Response. Cela sous-entend que le décodage est rapide.

La taille d'un QR Code est variable. Chaque version comporte 4 modules de plus par rapport à la version précédente. Il y a donc autant de versions que de formats soit 40. Sa dimension s'adapte au contenu selon son importance. Lorsque la dimension augmente et que le QR Code se complexifie, il peut arriver des erreurs de lecture. Ces codes-barres bidimensionnels peuvent supporter un certain taux d'erreurs et rester tout de même lisibles. Cela est dû au système Reed-Solomon qui est utilisé pour la correction des erreurs lors de la génération des QR Codes. Ceux-ci peuvent contenir jusqu'à 30% de redondance (les données étant codées plusieurs fois, la lecture reste possible même si une partie est illisible). Ce système permet donc d'insérer des logos et faire des modifications d'aspect tout en gardant un code parfaitement déchiffrable. Il existe 4 niveaux de correction. Le niveau L contient environ 7% de redondance, le niveau M : environ 15 %, le niveau Q : 25 % et enfin le niveau H qui contient environ 30% de redondance.

Ci-dessous, le schéma explique les caractéristiques techniques des QR Codes :





La capacité de stockage d'un QR Code dépend de sa version. Au maximum (version 40 avec 177 modules de côté), le stockage est de:

- 7 089 caractères numériques ou de,
- 4 296 caractères alphanumériques ou de,
- 2 953 octets (binaires 8-bits) ou de,
- 1 817 caractères de Kanji/Kana.

Les types de données pouvant être encodés sont nombreux . Parmi les plus intéressants, on peut citer les liens internet, les données textes, les données de géolocalisation, les données de connexions Wi-Fi et les possibilités de paiement.

Des variantes existent comme les iQR qui ont une forme rectangulaire et les QR Micro qui encodent peu de données.

## 1.2. Utilisations du produit et fonctionnalités

Cette section liste les différents cas d'utilisation du produit sous forme de scénario d'une prise de présence en classe avec QrAttendance.

Au début de chaque cours, le professeur s'authentifie au site web et choisit la classe qu'il est en train d'enseigner parmi une liste exhaustive des classes qui

lui sont affectées, cette liste est générée automatiquement par le système. Il demande ensuite la création d'un QR Code qui sera spécifique à la séance en cours et qui sera projeté depuis son ordinateur à travers le vidéo projecteur. Ce QR Code encode une information de la forme suivante : concaténation du username du professeur concerné et la date complète courante. De leur part, les étudiants se connectent aussi au site web pour scanner le code depuis leur téléphone. Les noms des étudiants ayant scanné le code s'affichent en temps réel dans l'écran du professeur pour la valider avant d'être envoyé à la base de données. Le professeur possède également la possibilité d'ajouter manuellement des étudiants à la liste des présents, en cas d'oubli du téléphone portable par exemple, ou de supprimer un étudiant de la liste des présents si souhaité.

De plus, le professeur peut consulter librement à la liste de présence de n'importe quelle séance déjà enseignée en saisissant simplement sa date.

L'application inclut également une interface administrateur. L'administrateur est chargé de gérer les étudiants, professeurs, matières et classes, ceci inclut :

- Créer les comptes d'authentification des étudiants et professeurs.
- Créer et supprimer les classes et matières.
- Affecter les professeurs aux matières et les matières aux classes.
- Affecter les étudiants aux classes.

Ces informations (qui doivent évidemment être renouvelées chaque semestre) servent à générer aux professeurs la liste des classes mentionnées précédemment, mais aussi à assurer une fonctionnalité très importante : le score. En effet, pour faciliter la partie évaluation, l'application offre aux professeurs un score de présence pour chacun de ses étudiants à portée d'un clic, et tout au long du semestre. Le score est calculé en divisant le nombre de séances auxquelles l'étudiant a assisté par le nombre total des séances, et est présenté en pourcentage.

Pour optimiser l'aspect de sécurité de notre application, nous avons implémenté une mesure permettant de vérifier si un étudiant a bel et bien scanné le QR Code depuis la classe, et non pas de l'extérieur. Cette mesure prend la forme d'un coefficient de précision calculé à partir de la distance spatiale entre l'appareil de l'étudiant et celui du professeur. Ce coefficient est alors affiché dans l'écran du professeur pour chaque étudiant ayant scanné le code. Pour activer cette fonctionnalité, il est évidemment nécessaire d'activer la géolocalisation dans tous les appareils concernés.

$$Coef = 1 - 10^3 \sqrt{(Latitude\ etudiant - Latitude\ professeur)^2 + (Longitude\ etudiant - Longitude\ professeur)^2}$$

### 1.3. Parties prenantes

#### **Professeurs**

Étant la cible majeure du produit, notre but dans la phase de développement était de proposer aux professeurs le service le plus convivial possible tout en assurant sa performance. Les professeurs sont en quelque sorte les clients du produit et seront notre source principale du feedback.

L'utilisation de l'application n'exige que le strict minimum en matière de connaissances technologiques, chaque interface de l'application concernant les professeurs comporte un bref guide d'utilisation pour une meilleure manipulation. Rappelons que cette manipulation se limite à demander un QR Code du système, consulter les listes de présences des séances assurées précédemment par le professeur ainsi que le score des étudiants.

#### **Étudiants**

Les étudiants sont évidemment indispensables pour le fonctionnement correct de l'application. Néanmoins, leur part du travail est la moins compliquée, puisqu'elle consiste simplement à scanner les QR Codes de chaque séance grâce à au cadre de caméra fourni par l'application.

#### **Administrateur**

Le rôle de l'administrateur consiste principalement en l'affectation des étudiants et professeurs aux différentes classes. Cette tâche est relativement critique, puisqu'une mauvaise affectation pourrait avoir des répercussions négatives sur le bon déroulement du service assuré par l'application. Les professeurs risqueraient, à titre d'exemples, de se retrouver devant des listes d'étudiants erronées, ou même ne pas avoir accès aux classes qu'ils enseignent; dissipant ainsi leur motivation à utiliser l'application. Dans ce contexte, des indications sur la manipulation correcte des fonctionnalités à risque sont explicitées dans les interfaces correspondantes, en plus de la possibilité qu'offre l'application d'apporter des modifications aux informations déjà entrées pour remédier aux erreurs.

## 2. Design et architecture

La phase d'étude du cahier des charges et des exigences techniques est primordiale dans le cycle de développement d'un logiciel. Elle doit précéder toute autre phase, car elle permet aux équipes une réflexion profonde sur les détails et particularités du projet en main et une bonne planification par la suite. Durant cette phase, l'équipe a pour but de trouver la meilleure stratégie vis-à-vis des contraintes et des ressources disponibles.

Après considérations, nous avons opté pour le modèle en cascade (figure 2.1) comme processus de développement.

Le modèle en cascade est en effet idéal pour les projets à court délai et à cahier des charges invariable, il oblige en fait les équipes à fixer le plan d'attaque et le respecter tout au long du développement.

Les autres types de modèles, notamment les modèles non linéaires, s'appuient fortement sur le feedback délivré par des parties extérieures à l'équipe de développement, ce qui n'était pas disponible dans notre cas.

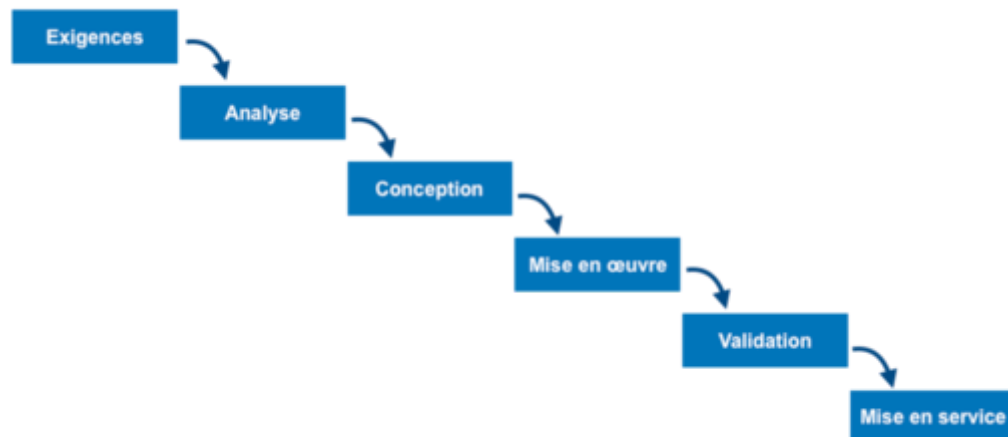


Figure 2.1 : modèle en cascade

La phase d'analyse nous a également servi à définir les paramètres clés de développement de notre application tels que l'architecture logicielle, la conception de base de données et le paradigme de programmation.

## 2.1. Architecture logicielle

QrAttendance a été développé avec le paradigme de programmation procédurale et suivant l'architecture monolithique. Une architecture monolithique représente le modèle traditionnel unifié de conception d'un programme informatique. Cette architecture implique que l'ensemble du code et fonctionnalités sont implémentés dans un seul programme, et que l'application est indépendante de toute autre entité informatique, et alors parfaitement autonome. La philosophie de conception monolithique fait que le programme soit déployé sous forme d'un seul bloc où toutes les composantes doivent être opérationnelles pour délivrer le service désiré ; les composants du programme sont en fait interconnectés et interdépendants plutôt qu'associés de manière flexible comme dans le cas des programmes modulaires.

L'architecture monolithique correspond fortement au degré de complexité de notre application. Ces architectures ont l'avantage d'être simples à construire, à tester et à déployer. En cas de croissance, la mise en échelle horizontale ne présente pas non plus de problème, où un répartiteur de charge permet d'exécuter plusieurs copies de l'application par exemple. De plus, les architectures monolithiques se caractérisent par une forte performance car les composantes de l'architecture unifiée partagent la mémoire vive, ce qui permet une communication optimale entre ces composantes.

En pratique, l'architecture monolithique fait en sorte que le programme soit naturellement monté autour d'une base de données unique répartie sur des tables. Le code est organisé de manière que les données parcourent l'application à travers des couches. Les données manipulées par l'application y accèdent par le sommet (habituellement une vue), et frayent leur chemin jusqu'au fond : la base de données. Le sens inverse, où les données ont pour source la base de données et pour destination la vue est d'autant commun. Le long du chemin, les couches intermédiaires assurent le traitement correct et la cohérence des données à travers des opérations telles que des conversions, transformations, parsages, calculs, etc.

Bien que l'architecture monolithique implique diverses approches de développement logiciel, l'approche MVC (Model-View-Contrôleur) est la plus répandue dans l'industrie. Le Model, View et Controller représentent les différentes couches de l'application dans l'approche MVC, où chaque couche a une responsabilité particulière (source : Openclassrooms) :

- Le Modèle gère les données du site. Son rôle est de récupérer les données brutes dans la base de données, les organiser, les assembler,

et ensuite les acheminer vers le contrôleur pour les traiter. On y trouve donc les requêtes SQL responsables de l'interrogation de la base de données.

- La Vue se concentre sur l'affichage et ne fait quasiment aucun traitement logique. Son rôle se limite à récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML.
- Le Contrôleur gère la logique de l'application et prend les décisions décisives. C'est l'intermédiaire entre la vue et le modèle : le contrôleur récupère les données du modèle, conduit une analyse et du traitement, puis les canalise vers la vue pour de l'affichage.

On remarque tout de suite le rôle important que joue le contrôleur. Il est responsable du traitement des requêtes des utilisateurs en contrôlant le flux d'informations entre les différents composants de l'application. Il joue essentiellement le rôle de relayeur entre la vue et le modèle en demandant les données du modèle et les transmet à la vue (après traitement) pour l'affichage, ou en recevant des données de la vue à écrire dans la base de données en passant par le modèle. Il est à préciser que le contrôleur gère aussi, entre autres, les autorisations d'accès aux différentes pages, les sessions, cookies, etc.

Nous avons développé QrAttendancy avec le modèle MVC en tête, mais pour des raisons de flexibilité, nous n'avons pas appliqué le modèle de la manière la plus rigoureuse possible. En effet, le modèle MVC procure une méthodologie de programmation réfléchie, détaillée et stricte. Respecter l'ensemble de ses exigences et règles s'avère le plus souvent très coûteux en matière de temps et ressources, c'est donc un investissement que seuls les projets imposants pourraient en bénéficier et tirer profit de son ample potentiel.

La flexibilité que cela a fourni nous a permis de tacler certains défauts de l'architecture monolithique qu'on découvrira par la suite.

Hormis le puissant arsenal d'avantages qu'offre l'architecture monolithique, elle n'est évidemment pas parfaite. Ses désavantages découlent principalement du fait que les composantes de l'application deviennent étroitement couplées, définissons ce que c'est que le couplage avant de procéder. Le couplage est une métrique (mesure) indiquant le degré d'interaction entre deux ou plusieurs composantes logicielles (fonctions, modules, object, etc.), deux entités sont alors couplées s'ils échangent des données. Un couplage étroit ou fort signifie alors que les composantes sont étroitement liées, enchevêtrés et interdépendantes. Le couplage étroit peut résulter en nombreuses complications :

- Difficulté à déterminer la composante responsable et/ou la raison d'une modification de données.

- Les composantes logicielles sont difficilement réutilisables et testables.
- Risque d'interblocage : Lors de l'accès à une ressource commune, si une composante bloque en session critique, l'autre bloque automatiquement.
- Perte d'autonomie : chaque composante doit être présente et fonctionnelle pour permettre l'exécution de l'application.
- Pour modifier ou remplacer une composante, le développeur est dans l'obligation de retoucher l'application toute entière, ceci peut entraîner des imprévus au sein d'autres éléments. Apporter des mises-à-jour est alors problématique.

Ayant conscience de ces défauts, une de nos priorités était d'adopter des mesures préventives pour assurer l'optimisation de notre application vis-à-vis des limites imposées par l'architecture monolithique. En effet, le fonctionnement de l'application repose sur deux concepts principaux : la génération de QR codes (par les professeurs) et la lecture de QR codes (par les étudiants), la flexibilité offerte par le modèle MVC dérivé nous a permis de séparer la logique qui implémente ces deux fonctionnalités du reste du code pour garantir un couplage faible entre les différents composants. Ceci a été réalisé par le biais de bibliothèques externes open source discutés dans la section suivante.

## 2.2. Bibliothèques utilisées

Durant la conception de notre application, une grande partie de la discussion a porté sur la manipulation des QR codes. Nous devons réaliser des algorithmes pour la création et la lecture des QR codes, or la réalisation d'un tel code est une tâche assez ambiguë qui nécessite beaucoup de temps, d'efforts et de maîtrise. De ce fait, nous avons effectué un petit tour sur les bibliothèques existantes de Javascript et nous avons choisi deux bibliothèques de QR codes auxquelles nous avons délégué ces deux tâches :

### 2.2.1. Easy-QR-Code-Js

EasyQRCodeJS est une bibliothèque de génération de QRCode JavaScript pure multi-navigateur riche en fonctionnalités. Cette bibliothèque emploie diverses méthodes de dessin telles que Canvas, SVG et Table, ainsi que des paramètres de style de point, logos, images d'arrière-plan, couleurs, titres, etc.



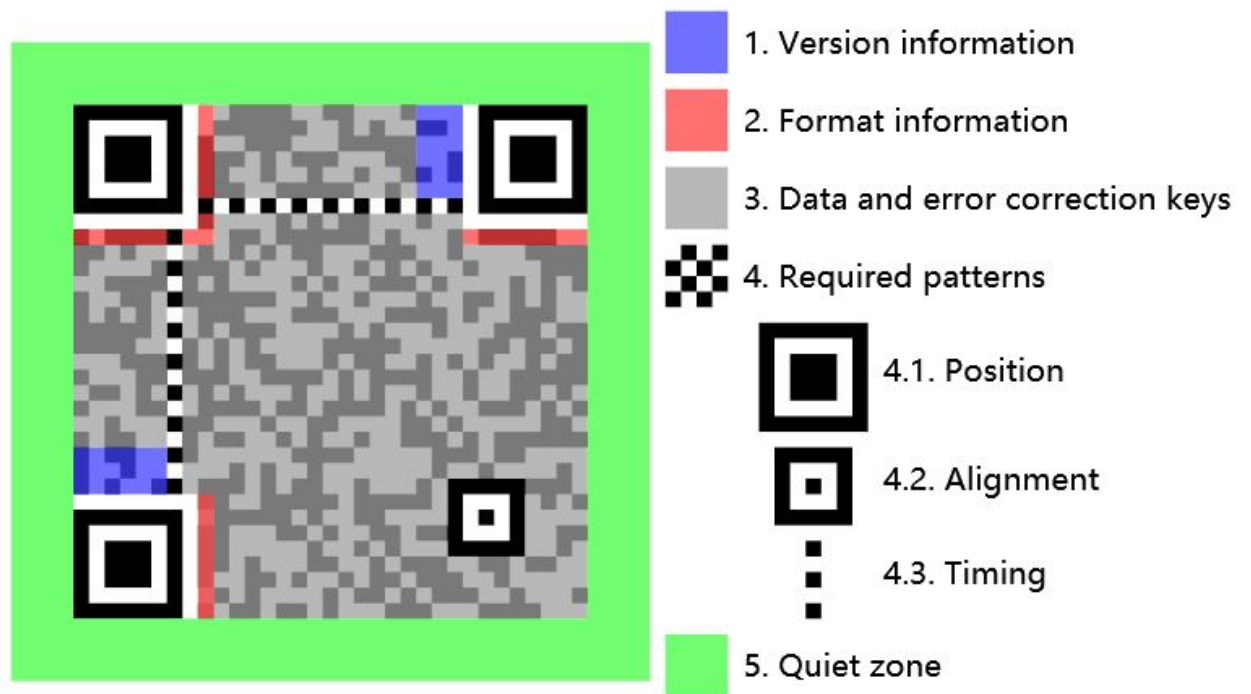
Voici une brève fiche technique de la bibliothèque :

### **Caractéristiques :**

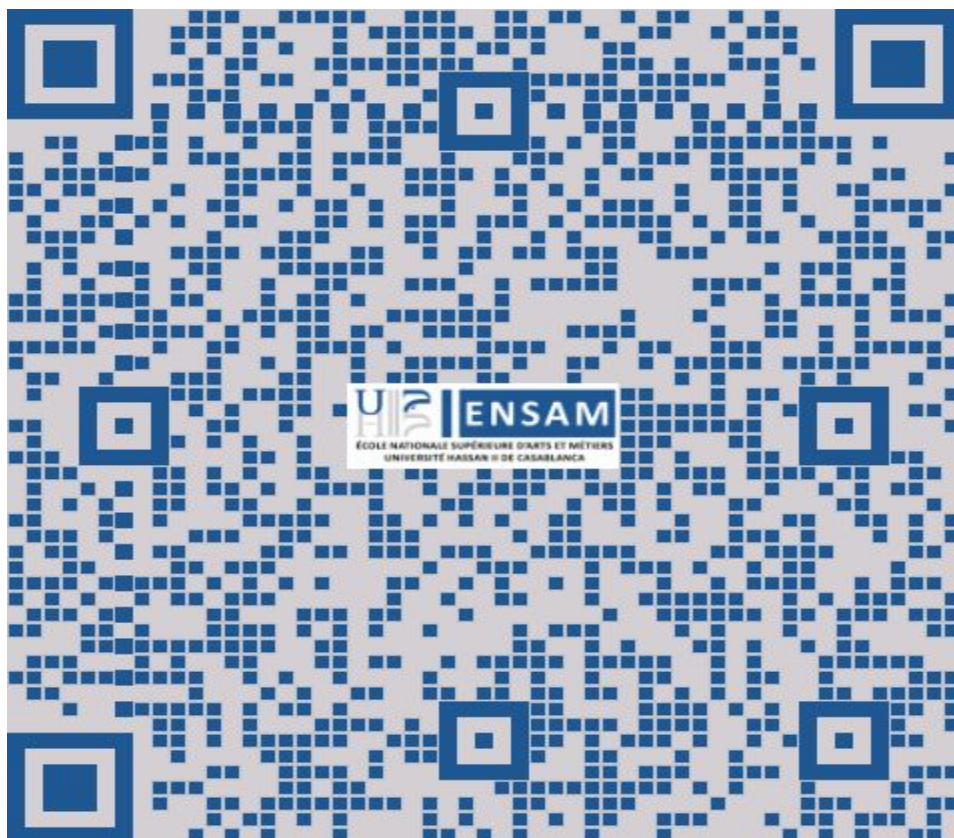
- Prise en charge de plusieurs navigateurs pour la génération de code QR basée sur HTML5 Canvas, SVG et Table.
- Requiert les Patterns qui supportent le dot style.
- Prise en charge des paramètres de zone silencieuse (The quiet zone).
- Prise en charge du remplissage intérieur et de la couleur de la bordure extérieure du Pattern de position personnalisé.
- Prise en charge du remplissage intérieur et de la couleur de la bordure extérieure du Pattern d'alignement personnalisé.
- Prise en charge des modèles de synchronisation personnalisés couleur verticale et horizontale.
- Prise en charge des images de logo (y compris les images PNG transparentes).
- Prise en charge d'arrière-plan.
- Prise en charge des paramètres de titre et de sous-titres.
- N'a pas de dépendances.
- Prise en charge des modules JavaScript AMD, CMD, CommonJS / Node.js.
- Prise en charge angulaire, Vue.js, React, Next.js.
- Prise en charge du mode de données binaire (hexadécimal).
- Prise en charge de TypeScript.

### **Structure de QR Code créé:**





**Exemple de QR Code créé:**



**Compatibilité avec les navigateurs :**

IE6 +, Chrome, Firefox, Safari, Opera, Mobile Safari, Android, Windows Mobile, ETC.

**Licence :**

Licence MIT.

**Code source :**

<https://github.com/ushelp/EasyQRCodeJS>

**2.2.2. Intascan:**

Instascan est une bibliothèque de lecture de QR Code, elle permet de scanner en temps réel le QR Code en se basant sur la webcam.

**Performance de la bibliothèque**

De nombreux facteurs affectent la rapidité et la fiabilité d'Instascan pour détecter les codes QR :

- Un code physique plus volumineux est préférable. Un code carré de 2 "est préférable à un code carré de 1".
- Les surfaces planes, lisses et mates sont meilleures que les surfaces courbes, rugueuses et brillantes
- Inclure une quiet zone suffisante, la bordure blanche entourant le code QR. La quiet zone doit avoir au moins quatre fois la largeur d'un élément individuel de votre code QR.
- Un code plus simple est préférable.
- Pour la même longueur, le contenu numérique est plus simple que le contenu ASCII, qui est plus simple que le contenu Unicode
- Un contenu plus court est plus simple. Si vous encodez une URL, pensez à utiliser un raccourcisseur tel que [goo.gl](http://goo.gl) ou [bit.ly](http://bit.ly).

Lors de la numérisation, tenez compte des éléments suivants:

- L'orientation du code QR n'a pas d'importance.

- Une vidéo de résolution plus élevée est meilleure, mais elle est plus gourmande en CPU.
- Un balayage orthogonal direct est meilleur que le balayage à un angle.
- Une vidéo floue réduit considérablement les performances du scanner.
- La mise au point automatique peut entraîner des retards de détection lorsque la caméra ajuste la mise au point. Pensez à le désactiver ou à utiliser un appareil photo à mise au point fixe avec le sujet positionné au point focal.
- Le réglage de l'exposition sur les caméras peut entraîner des retards de détection. Pensez à le désactiver ou à avoir un fond blanc fixe.

### **Limites**

La bibliothèque n'est pas directement compatible avec les périphériques IOS, ce problème vient du fait que le tableau qui permet de retourner les caméras du périphérique commence par les caméras d'arrière dans les périphériques IOS, et par les caméras d'avant dans les périphériques non-IOS. On a réglé le problème en ajoutant une condition sur le type du périphérique en utilisant la condition suivante : `if(/iPad|iPhone|iPod/.test(navigator.userAgent) && !window.MSStream).`

La bibliothèque n'est pas directement compatible avec les périphériques qui possèdent un nombre de caméras qui dépasse deux. On a réglé ce problème en ajoutant une condition sur le nombre de caméras du périphérique.

### **Crédits:**

Powered by the Emscripten JavaScript build of the C++ port of the ZXing Java library

### **License:**

Copyright © 2016 Chris Schmich  
MIT License.

### **Code source:**

<https://github.com/schmich/instascan>

## 2.3. Mise en oeuvre et structure logicielle

Cette section décrit la logique de l'application et la manière dont les différentes parties interagissent pour délivrer les services souhaités. Pour cela, nous nous servons des diagrammes du langage unifié de modélisation UML. L'UML est un support de communication performant, il permet grâce à sa représentation graphique d'exprimer visuellement la solution logicielle et de faciliter la comparaison et l'évolution de la solution. Bien que l'UML est un outil orienté objet, cela ne nous a pas empêcher de profiter de la puissance de quelques-uns de ses nombreux diagrammes. Les diagrammes de cas d'utilisation, de séquences et d'activité sont utilisés ici pour communiquer au lecteur l'enchaînement des actions et interactions effectuées par l'application pour mener à bien la prise de présence en classe.

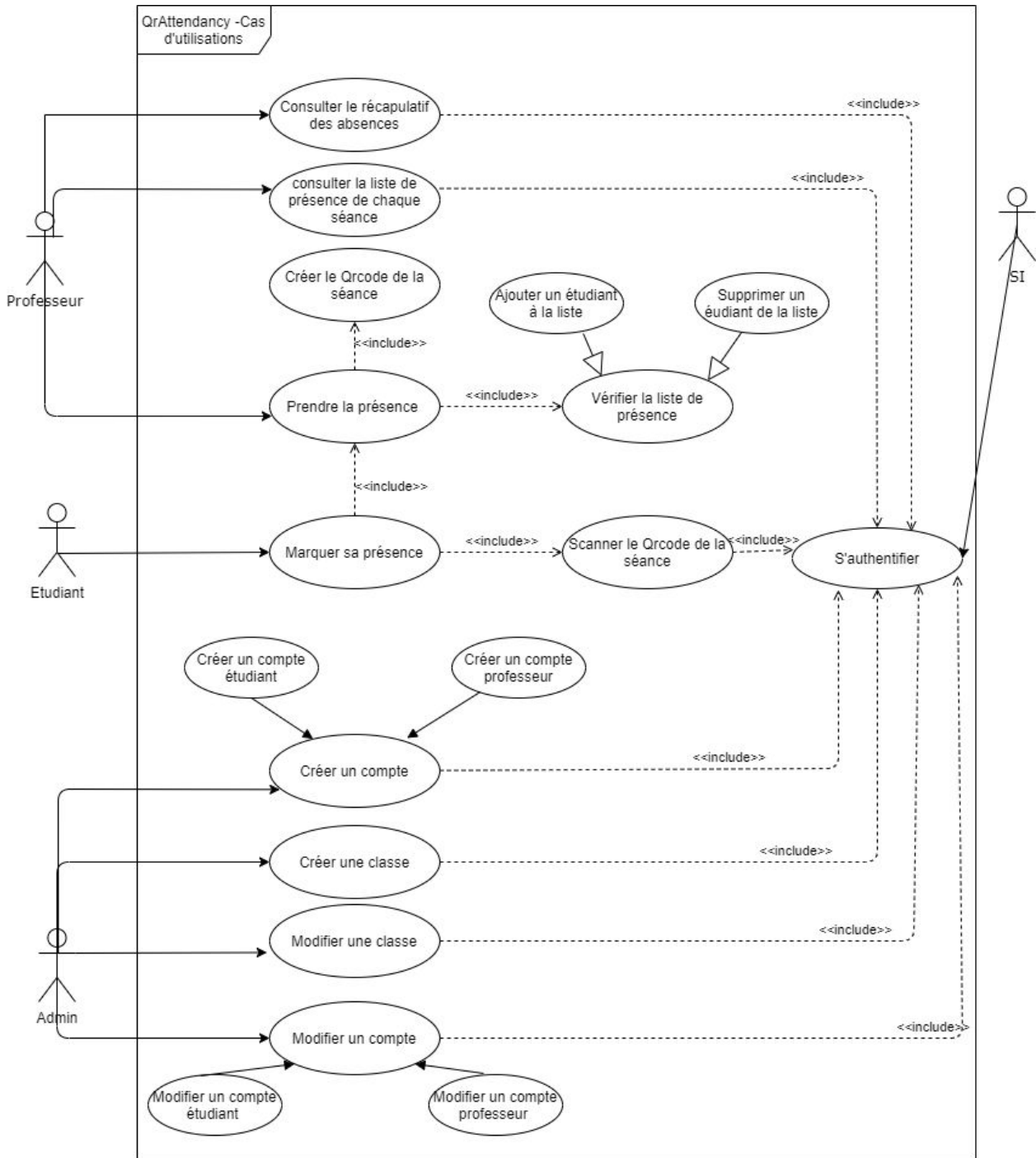
Les diagrammes UML utilisés offrent une explication haut niveau du fonctionnement de l'application. Pour une vue bas niveau, c'est-à-dire les détails de l'implémentation de l'application, nous vous invitons à consulter la page [GitHub](https://github.com/alibarj/QrAttendance) du projet où nous fournissons une explication exhaustive du code, en précisant le rôle de chaque dossier et fichier dans le résultat final du projet. Nous y mettons également le point sur les détails de la communication entre les deux bibliothèques de création et de scan de Qr codes et le reste de l'application.

<https://github.com/alibarj/QrAttendance>

### Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation (figure 2.2) offre une vue détaillée sur l'ensemble des fonctionnalités de l'application en mettant en évidence le rôle de chaque acteur et utilisateur et la manière dont ils interagissent avec le système.

En plus du diagramme, des descriptions textuelles sont fournies pour les cas d'utilisations principaux : prendre la présence (acteur professeur) et marquer sa présence (acteur étudiant), ainsi qu'un cas d'utilisation relatif à l'acteur 'administrateur' : créer un compte étudiant/professeur.



**Diagramme des cas d'utilisations**

Figure 2.2

### **Description textuelle pour cas d'utilisation Prendre la présence:**

<b>Sommaire d'identification</b>
<p><b><u>Titre</u></b> : Prendre la présence.</p> <p><b><u>But</u></b> : Cette fonctionnalité permet au professeur de saisir la liste de présence.</p> <p><b><u>Résumé</u></b> : le professeur crée le QR code de la séance, les étudiants scannent et le professeur enregistre la liste des étudiants qui ont scanné le QR code.</p> <p><b><u>Acteurs</u></b> : Professeur(principale), Etudiant(secondaire).</p>
<b>Description de l'enchaînement</b>

**Pré condition** : Le professeur est authentifié.

**Post condition** : une nouvelle liste de présence sera enregistrée.

**Scénario nominal** :

1-Le système demande au professeur d'activer la géolocalisation pour qu'il puisse calculer la distance entre le professeur et l'étudiant qui a scanné le QR code.

2-Le professeur accepte l'activation de la géolocalisation.

3-Le système affiche la liste des classes et des éléments de module du professeur.

4-Le professeur choisit la classe et l'élément de module concerné.

5- Le système affiche les fonctionnalités possibles pour le professeur.

6- Le professeur crée le QR Code de la séance.

7- Le système affiche le QR Code de la séance.

8- Les étudiants scannent le qrcode.

9-Le système enregistre temporairement les étudiants qui ont scanné le qrcode.

10-Le système affiche la liste des étudiants qui ont scanné le qrcode.

11-Le professeur vérifie la liste de présence (en se basant principalement sur la distance entre l'étudiant et le professeur...).

12-Le professeur confirme la liste de présence.

13-Le système enregistre la liste de présence et supprime la liste temporaire.

14-Le système détruit le QR Code de la séance.

**Scénarios alternatifs** :

**2-a : Le professeur ne veut pas**

**activer la géolocalisation.**

-Le système affiche la liste des classes et des éléments de module du professeur.

- Le professeur choisit la classe et l'élément du module concerné.
- Le système affiche les fonctionnalités possibles pour le professeur.
- Le professeur crée le QR Code de la séance.
- Le système affiche le QR Code de la séance.
- Les étudiants scannent le qrcode.
- Le système enregistre temporairement les étudiants qui ont scanné le qrcode.
- Le système affiche la liste des étudiants qui ont scanné le qrcode.
- Le professeur vérifie la liste de présence (la distance prof étudiant n'est pas calculée, le professeur vérifie la liste en se basant par exemple sur le nombre d'étudiants existants dans la salle...).

Le scénario reprend au point 12.

**11-a : Le professeur trouve que la liste n'est pas valide (par exemple un étudiant absent et scanne le QR Code depuis l'extérieur de la classe, un étudiant présent mais il a oublié son téléphone...)**

- Le professeur corrige la liste en ajoutant et/ou supprimant des étudiants.

Le scénario reprend au point 12.

#### **Scénario d'échec :**

**8-a : Aucun étudiant n'a scanné le qrcode.**

- Le professeur ne peut pas confirmer la liste (le système ne peut pas enregistrer une liste vide).

#### **Description textuelle pour cas d'utilisation Marquer sa présence :**



### Sommaire d'identification

**Titre** : Marquer sa présence.

**But** : Cette fonctionnalité permet aux étudiants de saisir la de présence.

**Résumé** : L'étudiant scanne le qrcode de la séance, le système enregistre l'étudiant dans la liste de présence.

**Acteur** : Étudiant.

### Description de l'enchaînement

**Pré condition** : L'étudiant est authentifié, Le professeur a créé le qrcode de la séance.

**Post condition** : Une nouvelle demande de présence sera enregistrée.

**Scénario nominal** :

1-Le système demande à l'étudiant d'activer la géolocalisation (sera utilisée pour calculer la distance étudiant professeur).

2-L'étudiant accepte l'activation de la géolocalisation.

3-Le système demande à l'étudiant d'activer la caméra.

4-L'étudiant active la caméra.

5- Le système ouvre la caméra et demande à l'étudiant de scanner le QR code.

6- L'étudiant scanne le QR code.

7- Le système enregistre temporairement l'étudiant qui a scanné le qrcode en calculant la distance professeur étudiant.

8-Le système affiche à l'étudiant que sa demande est bien enregistrée.

**Scénario alternatif** :

**2-a : L'étudiant ne veut pas activer la géolocalisation.**

-Le système demande à l'étudiant d'activer la caméra.

-L'étudiant active la caméra.

- Le système ouvre la caméra et demande à l'étudiant de scanner le QR code.

- L'étudiant scanne le QR code.

- Le système enregistre temporairement l'étudiant qui a scanné le qrcode sans calculer la distance professeur étudiant.

Le scénario reprend au point 8.

**Scénario d'échec** :

**4-a : l'étudiant n'active pas la caméra.**

-Le système arrête la procédure.

.

**Description textuelle pour cas d'utilisation Créer un compte :**

<b>Sommaire d'identification</b>
<p><b><u>Titre</u></b> : Créer un compte (pour le prof ou étudiant).</p> <p><b><u>But</u></b> : Cette fonctionnalité permet à l'admin de créer des comptes.</p> <p><b><u>Acteur</u></b> : Admin.</p>
<b>Description de l'enchaînement</b>

**Pré condition** : L'admin est authentifié.

**Post condition** : Un nouveau compte sera enregistré.

**Scénario nominal** :

- 1-Le système demande à l'admin de choisir le service correspondant.
- 2-L'admin choisit la création d'un compte.
- 3-Le système demande les informations requises.
- 4-L'admin saisit les informations requises.
- 5-Le système vérifie les informations.
- 6- Le système enregistre les informations du compte.
- 7- Le système annonce que le correct est créé.

**Scénarios alternatifs** :

**4-a : L'admin saisit des informations erronées.**

- Le système vérifie les informations
- Le système avertit l'admin en précisant une erreur de syntaxe.

Le scénario reprend au point 3.

**4-b : L'admin saisit des informations existantes.**

- Le système vérifie les informations
- Le système avertit l'admin en précisant que le compte existe déjà.

Le scénario reprend au point 3.

## Diagramme de séquences

A travers un diagramme de séquences UML, nous illustrons les interactions des deux acteurs 'professeur' et 'étudiant' avec le système plus en détails. Cette illustration prend la forme de séquences chronologiques, c'est-à-dire des vas et viens entre les acteurs et les composantes du système pour éventuellement exécuter la finalité de l'application. Pour offrir une meilleure explication aux différentes interactions, le diagramme sera divisé en sous-groupes chacun discuté à part.

- Le premier groupe de séquences (figure 2.4) concerne l'authentification des deux acteurs, il fait référence à un autre diagramme nommé 'Authentification' (figure 2.5). Cette séquence est encadrée par un opérateur 'weak' qui indique que l'ordre d'exécution des authentifications n'est pas important.

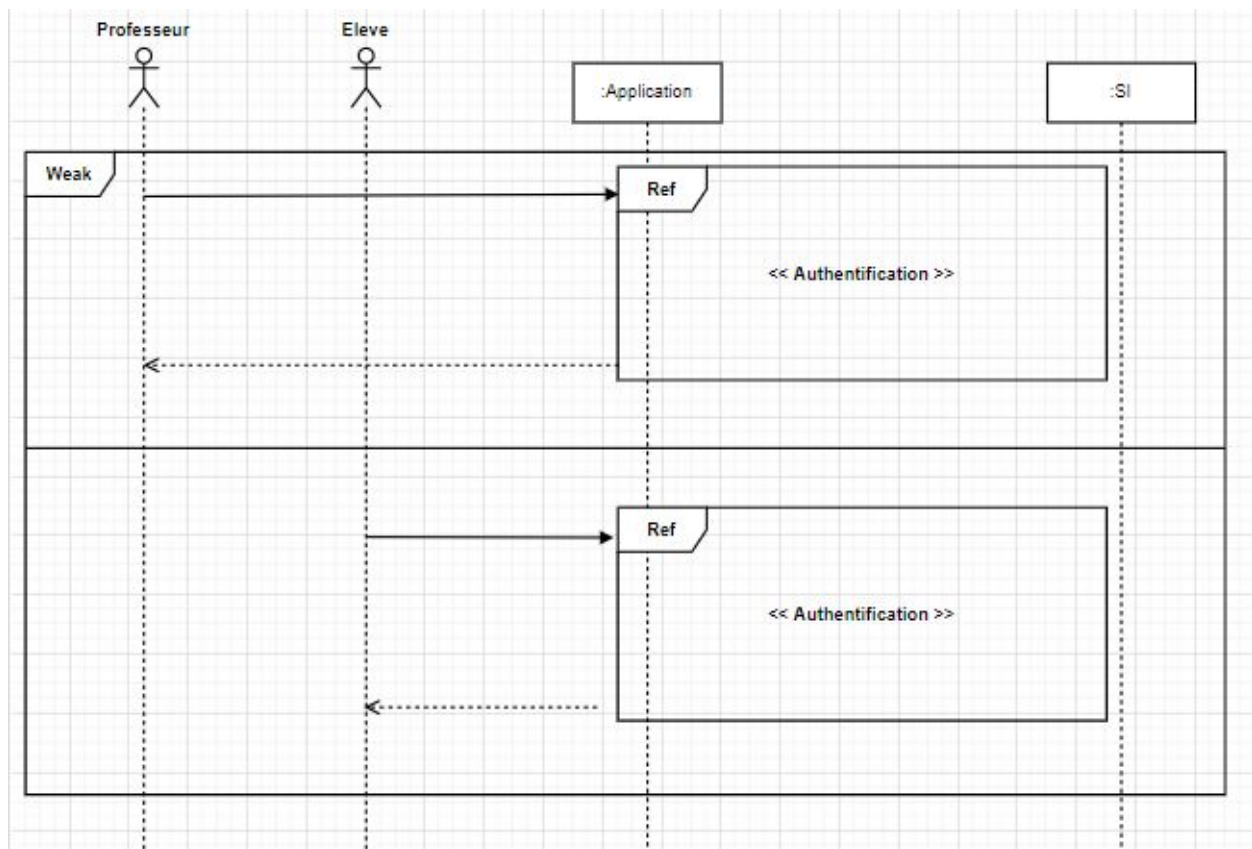


Figure 2.4

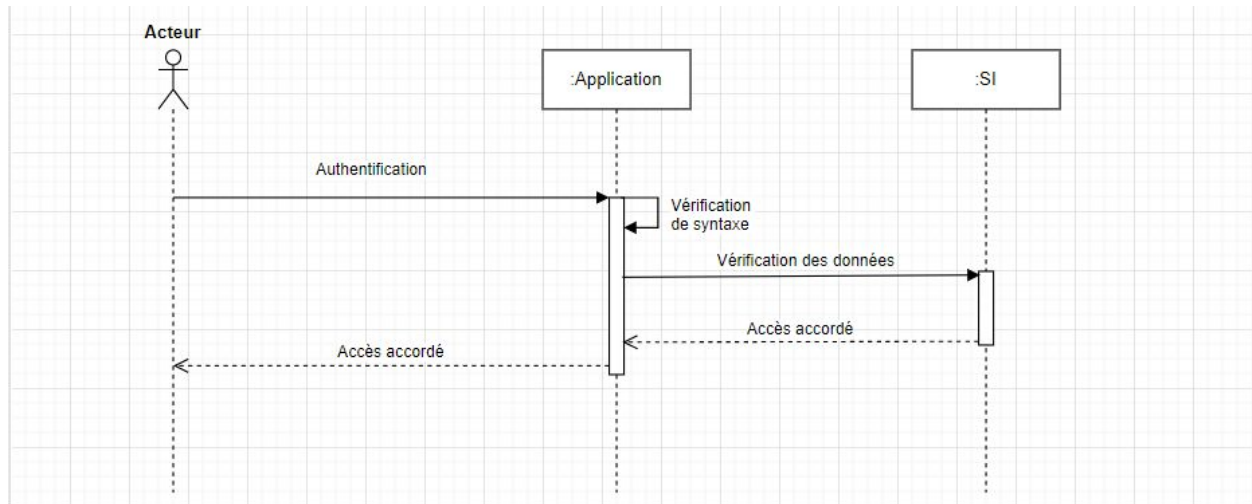


Figure 2.5

- Le deuxième groupe de séquences (figure 2.6) : le système demande au professeur de choisir une classe, et le professeur fait son choix.

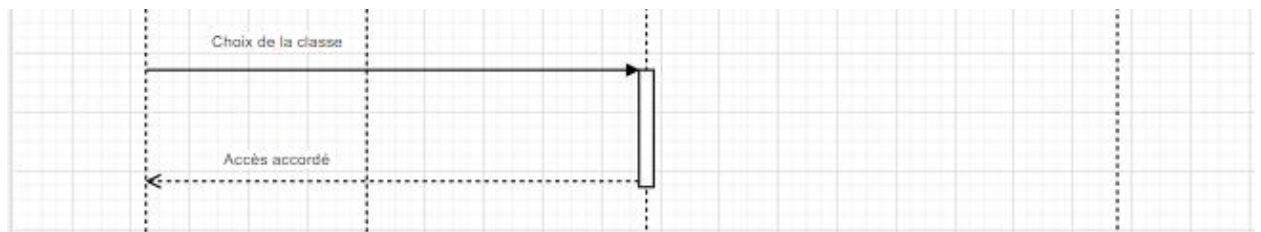


Figure 2.6

- Le troisième groupe de séquences (figure 2.7) : Le professeur clique sur le bouton de génération de QR code, l'application crée alors un QR code unique pour la séance, celui-ci est ensuite affiché et les étudiants procèdent au scan depuis leur appareils, et finalement un message de succès s'affiche aux étudiants qui ont scanné le code pour leur informer que leur présence a été marqué pour la séance en cours.  
Notons qu'après le choix de classe, le professeur a également la possibilité de consulter la liste de présence d'une séance antérieure ou le score de présence d'un étudiant, bien que cela n'est pas représenté dans ce diagramme.

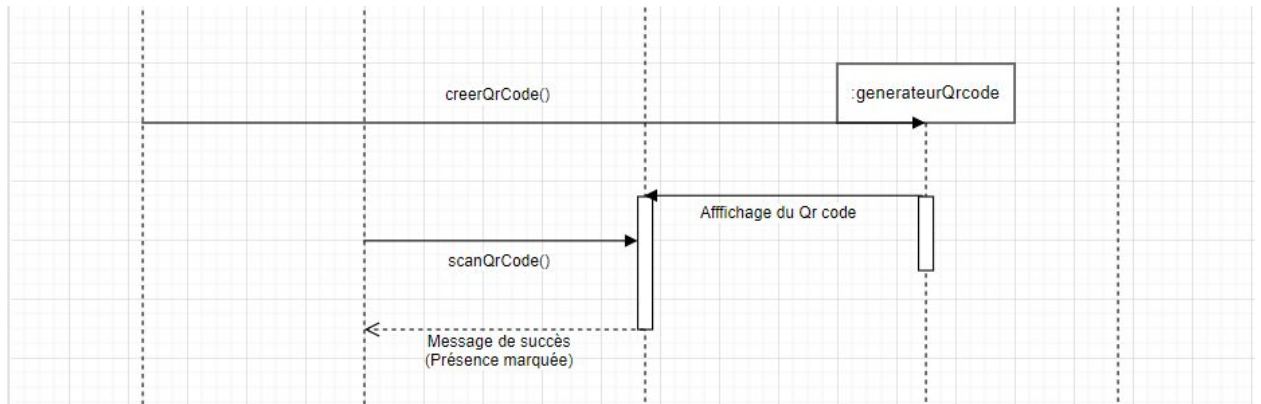


Figure 2.7

- Le quatrième groupe de séquence (figure 2.8) est une interaction optionnelle, d'où l'opérateur 'opt' : après que les étudiants ont scanné le QR code, le professeur a la possibilité de modifier la liste de présence manuellement.

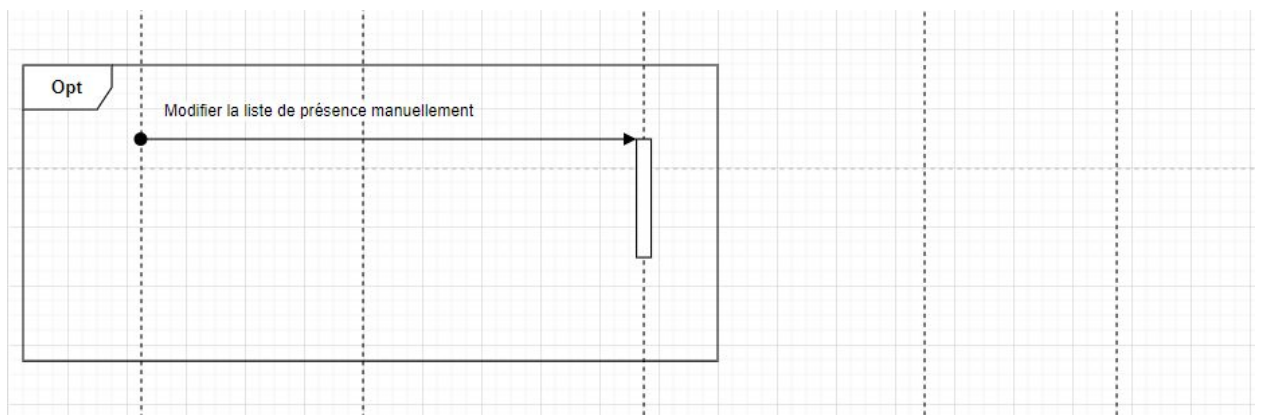


Figure 2.8

- Dans la dernière séquence (figure 2.9), le professeur valide la liste de présence, le QR code est détruit et la liste de présence est enregistrée dans la base de donnée.

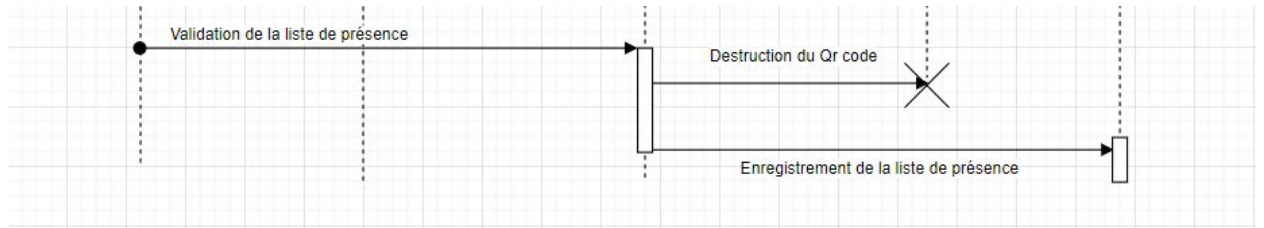


Figure 2.9



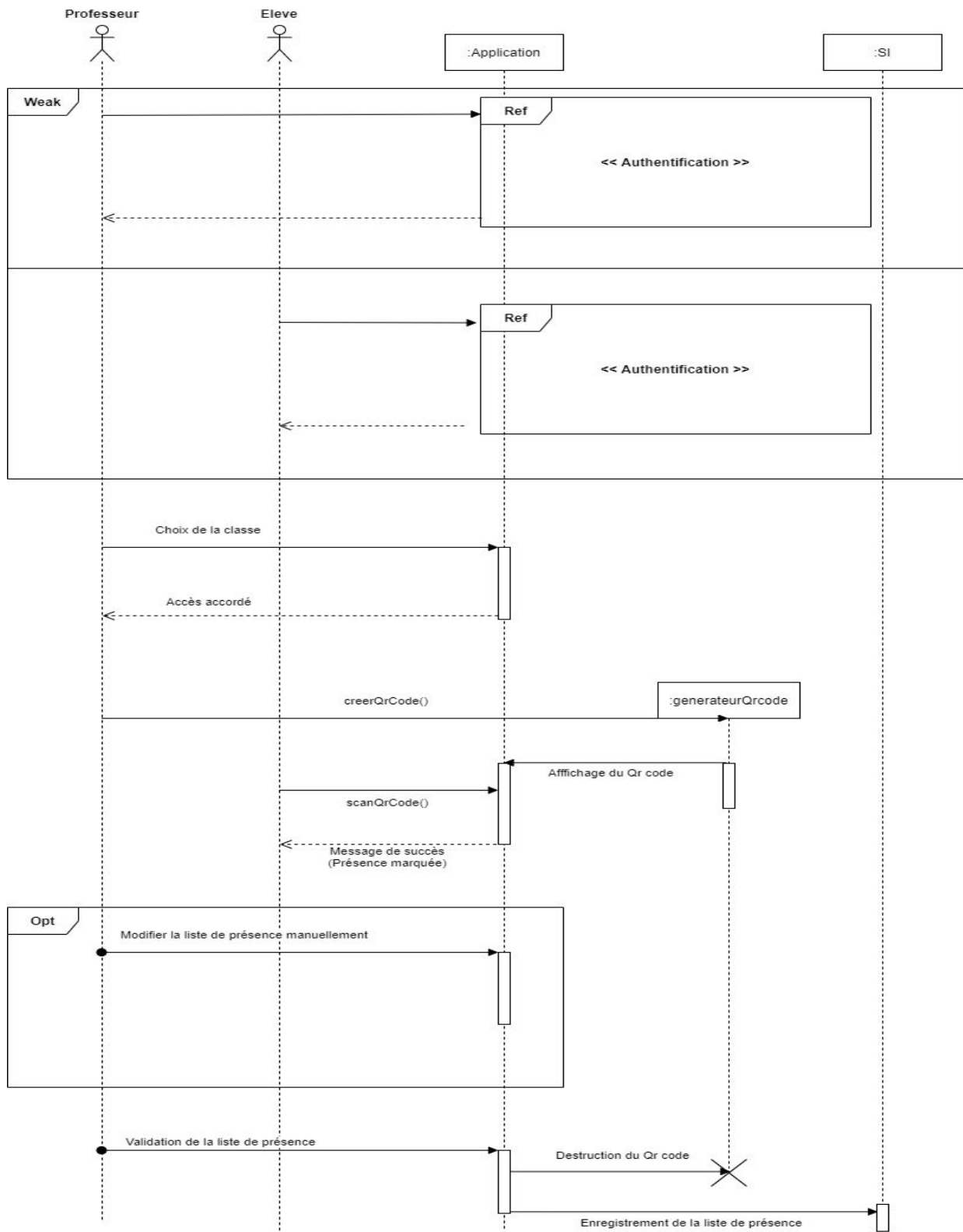


Diagramme de séquences : Prendre la présence.

### 2.3.1. Diagramme d'activité

En plus des diagrammes des cas d'utilisation et séquences, nous proposons un diagramme d'activité pour l'action de prise de présence (figure 2.10).

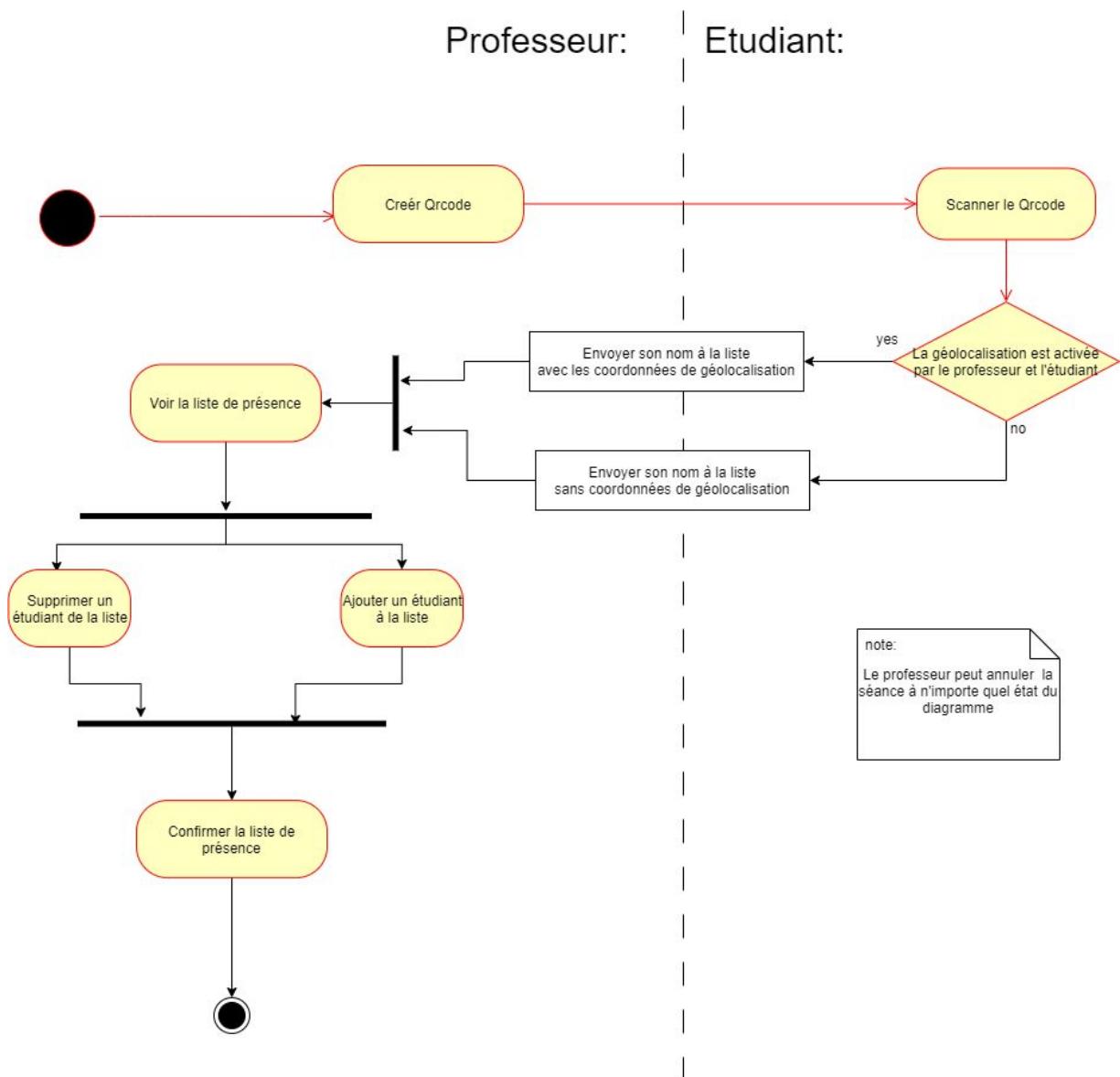


Figure 2.10

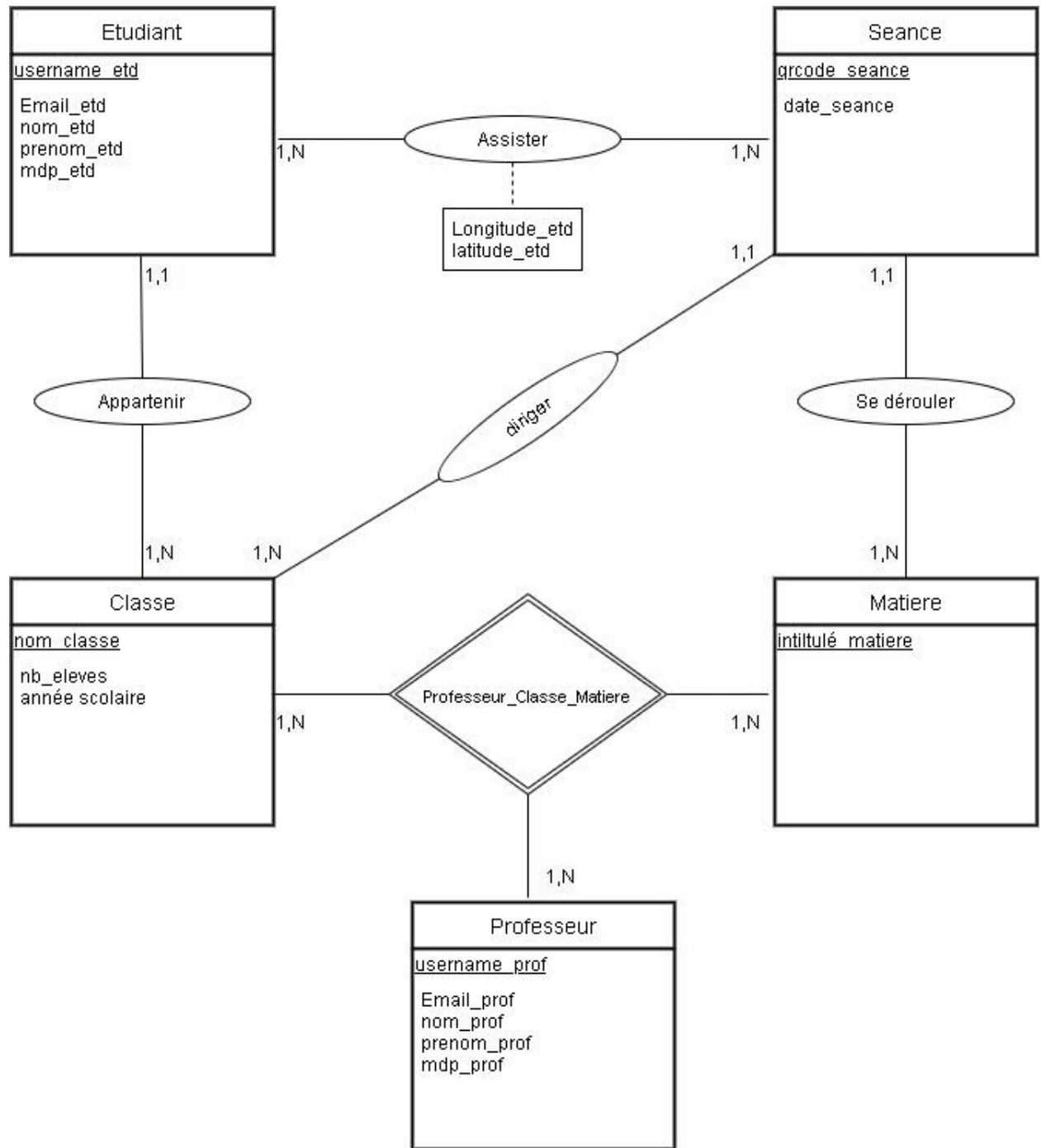
Le diagramme d'activité est une représentation graphique UML comportemental qui traduit le flux de travail (workflow) d'une activité assurée par l'application étape par étape, il met l'accent sur les conditions/choix, l'itération et la concurrence.

Le diagramme est divisé en deux colonnes : étudiant et professeur. L'activité est initiée par l'action "Créer Qr code" par le professeur et l'action "Scanner le Qr code" par l'étudiant, vient ensuite une décision faite par l'étudiant et le professeur "Activer la géolocalisation", après le scan, le nom de l'étudiant ayant scanné le code est envoyé à l'écran du professeur dans les deux cas, le professeur a en effet le mot final concernant la validité des données de présence. L'action suivante est "Voir la liste", le professeur visionne en temps réel les étudiants qui scannent le QR code. Cela conduit à un split où le professeur peut "Ajouter un étudiant" ou "Supprimer un étudiant" de la liste qui s'affiche dans son écran en concurrence. L'activité se joint alors en l'action "Confirmer la liste" par le professeur, la liste est envoyée à la base de données et l'activité prend fin.

## **2.4. Base de données**

### **2.4.1. Modèle conceptuel de données**

le modèle conceptuel de données est illustré dans la figure suivante:



Un étudiant peut assister à plusieurs séances et une séance peut être assistée par plusieurs étudiants, lors du scan de QR code on doit récupérer les longitudes et les latitudes des étudiants afin d'éviter qu'un étudiant tente de scanner le qr code à partir d'un endroit autre que la classe, d'où la mise en place d'une association avec les attributs (longitude\_etd, latitude\_etd).

Un professeur enseigne une matière à une classe d'où l'association ternaire entre l'entité professeur, l'entité matière et l'entité classe, cette association va être transformée à une table qui sera principalement utilisée par les jointures pour extraire les informations souhaitées.

#### 2.4.2. Modèle logique de données

**Etudiant** (username\_etd, email\_etd, nom\_etd, prenom\_etd, mdp\_etd, #nom\_classe)

**Seance** (qrcode séance, date\_seance, #nom\_classe, #intitulé\_matiere)

**Matiere**(intitulé\_matiere)



**Professeur** (username\_prof, email\_prof, nom\_prof, prenom\_prof, mdp\_prof)

**Classe** (nom\_classe, nb\_eleves, année\_scolaire)

**Assister** ((#qrcode\_seance, #username\_etd), longitude\_etd, latitude\_etd)

**Professeur\_Classe\_Matiere** ((#username\_prof, #nom\_classe, #intitulé\_matiere))

### 2.5. Technologies

	<p><b>HTML5</b> pour <b>Hypertext Markup Language 5</b> est la dernière version du HTML (Lancée en octobre 2014). Ce langage est utilisé pour concevoir les sites Internet.</p>
	<p>CSS est l'acronyme de « Cascading Style Sheets » ce qui signifie « feuille de style en cascade ».</p> <p>Le CSS correspond à un langage informatique permettant de mettre en forme des pages web (HTML ou XML).</p> <p>Ce langage est donc composé des fameuses « feuilles de style en cascade » également appelées fichiers CSS</p>



**JavaScript** est un langage interprété par le navigateur. Le JavaScript est un langage « client », c'est-à-dire exécuté chez l'utilisateur lorsque la page Web est chargée. Il a pour but de dynamiser les sites Internet.



**PHP** (officiellement, ce sigle est un acronyme récursif pour **PHP Hypertext Preprocessor**) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML.



**MySQL** dérive directement de **SQL (Structured Query Language)** qui est un langage de requête vers les bases de données exploitant le modèle relationnel, mais ne possède pas toute la puissance du langage SQL.



jQuery est une bibliothèque JavaScript libre et multiplateforme créée pour faciliter l'écriture de scripts côté client dans le code HTML des pages web



Ajax est l'acronyme d'*asynchronous JavaScript and XML* : *JavaScript et XML asynchrones*.

Ajax combine [JavaScript](#) et [DOM](#), qui permettent de modifier l'information présentée dans le navigateur en respectant sa structure, les [API Fetch](#) et [XMLHttpRequest](#), qui servent au dialogue asynchrone avec le serveur Web ; ainsi qu'un format de données ([XML](#) ou [JSON](#)),



afin d'améliorer maniabilité et confort d'utilisation des des applications internet riches.

**Bootstrap** est une [collection d'outils](#) utiles à la création du design (graphisme, animation et interactions avec la page dans le navigateur, etc.) de [sites](#) et d'[applications web](#).

### 3. Exigences (Requirements)

Pour assurer le bon fonctionnement de l'application, un certain nombre d'exigences matérielles et fonctionnelles doivent être satisfaites, ainsi que des exigences de performance pour une optimisation du rendement de l'application.

#### 3.1. Exigences matérielles

- Le professeur doit disposer d'un ordinateur, un navigateur web compatible avec la bibliothèque Easy-Qr-Code-Js (voir section 2.3) et une connexion internet.
- L'étudiant doit disposer d'un téléphone ou tablette portable, un navigateur web compatible avec la bibliothèque Instascan (voir section 2.3) et une connexion internet.
- L'administrateur doit disposer d'un ordinateur et d'une connexion internet.
- La salle du cours doit être dotée d'au moins un vidéo-projecteur. La qualité d'image que doit procurer le vidéo-projecteur pour garantir que l'ensemble des étudiants présents puisse scanner le code n'est encore pas déterminé, des tests sur terrain doivent être effectués avant de conclure.

#### 3.2. Exigences fonctionnelles

Pour chacun des cas d'utilisation (voir section 2.3), l'application doit pouvoir satisfaire un nombre d'exigences pour être qualifiée de fonctionnelle.

Pour les cas d'utilisation du professeur :

- L'application doit permettre l'authentification correcte du professeur si les informations fournies sont justes.
- L'application doit pouvoir afficher la liste correcte des matières enseignées par le professeur, selon les informations fournies par l'administrateur.
- L'application doit pouvoir fournir un score juste pour chaque étudiant, en plus du nombre de séances auxquelles un étudiant a assisté.
- L'application doit pouvoir récupérer toutes les séances d'une matière conformément à la date saisie par le professeur.
- L'application doit pouvoir générer un code instantanément, afficher les étudiants présents, offrir la possibilité de modifier la liste en temps réel et sans latence, ainsi que annuler la séance à tout instant.



Pour les cas d'utilisation des étudiants :

- L'application doit permettre l'authentification correcte de l'étudiant si les informations fournies sont justes.
- L'application doit pouvoir ouvrir la caméra arrière de l'appareil utilisé par l'étudiant, quel que soit le type de l'appareil.
- L'application doit pouvoir scanner le QR Code si la distance entre l'étudiant et la projection le permet, et le marquer présent si le QR Code est bel et bien celui d'une séance enseignée par le professeur.

Pour les cas d'utilisation de l'administrateur :

- L'application doit pouvoir afficher des formulaires d'ajouts et de modification de classes, comptes étudiants et comptes professeurs cohérents, ainsi que vérifier la consistance de la base de données avant toute opération.

### 3.3. Exigences de performance (performance requirements)

Les exigences de performance définissent dans quelle mesure le système logiciel accomplit certaines fonctions dans des conditions spécifiques.

- Vitesse : La vitesse de chargement de toute interface entre l'utilisateur et le système ne doit pas dépasser les 5 secondes.  
Le système doit actualiser la liste des présents chaque 1.5 secondes pour le professeur.
- Précision : Le score des étudiants doit être juste au centième près.  
Le nombre de présences des étudiants doit toujours être exact.
- Capacité : L'application doit pouvoir gérer le trafic généré par le nombre d'étudiants et professeurs présents simultanément dans l'école.  
L'application doit pouvoir stocker les données générées durant deux années au moins.
- Fiabilité : Le temps de panne accepté est au maximum une heure; nous admettons que le professeur peut prendre la présence à n'importe quel instant du cours. Au cours d'une panne, aucune donnée ne doit être perdue ou endommagée pour éviter que le professeur reprenne la procédure.
- Disponibilité : L'application doit obligatoirement être disponible à l'utilisation durant les heures de cours (de 8h à 18h) et durant toutes les saisons exceptée celle des vacances.

- Accès : Les listes de présence et scores des étudiants doivent être uniquement accessibles par le professeur enseignant la matière. Les informations relatives aux classes, affectation des professeurs et étudiants, les comptes d'authentification doivent être accessibles uniquement à l'admin.

## Conclusion

Développer QrAttendancy a été une expérience extrêmement enrichissante pour nous. Elle nous a permis de mettre en œuvre nos acquis techniques de la première année en développement informatique, notamment en développement web back end et front end, systèmes d'informations et bases de données. Elle nous a également permis d'exercer notre compétence de travail d'équipe, particulièrement parce que le développement a eu lieu en période de confinement, et que l'ensemble des échanges et communications devaient se faire à distance.

Le travail sur QrAttendancy n'est évidemment pas achevé, le projet a un potentiel d'évolution énorme, et ce sur différents aspects.

Sur le plan architectural, nous estimons que le paradigme de programmation orientée objet sera plus adéquat aux besoins de l'application maintenant qu'on a suffisamment exploré sa puissance. La programmation orientée objet offre une toute nouvelle façon de résoudre les problèmes logiciels grâce à des solutions telles que les design patterns à titre d'exemple, ceux-ci permettent une écriture de code rapide, lisible, efficace et maintenable. Nous jugeons aussi que l'application pourrait bénéficier d'une architecture microservices au lieu de monolithique, où la communication avec les bibliothèques de création et scan de QR Codes aura lieu moyennant des API. Cette approche améliorera la vitesse de l'application en évitant que le code source des bibliothèques ne soit téléchargé chez le client chaque fois qu'une page de l'application est chargée. Une migration d'une application web vers une application mobile pour la partie étudiant est aussi envisageable. Cette solution offrirait une expérience plus conviviale et fluide pour les étudiants lors de l'utilisation de QrAttendancy.

Sur le plan fonctionnelle, un déploiement on premises (ou sur site), où l'application et ses dépendances seront installés et exécutés sur les serveurs de l'ENSAM, pourrait être une meilleure option. L'application ne sera alors accessible qu'à l'intérieur de l'école, s'opposant à l'hébergement on demand sur un serveur distant, optimisant ainsi la sécurité, la performance et le coût d'hébergement. Dans le même sens, l'application pourrait bénéficier d'une communication directe avec le système d'information de l'école pour récupérer les données sur les professeurs, étudiants et cours indispensables au fonctionnement de l'application, minimisant ainsi la saisie manuelle de la part de l'administrateur.

Finalement, nous croyons aussi en la portabilité de QrAttendancy. Avec les modifications nécessaires, le projet pourra s'adapter facilement à tout autre

environnement où les absences sont comptabilisées.