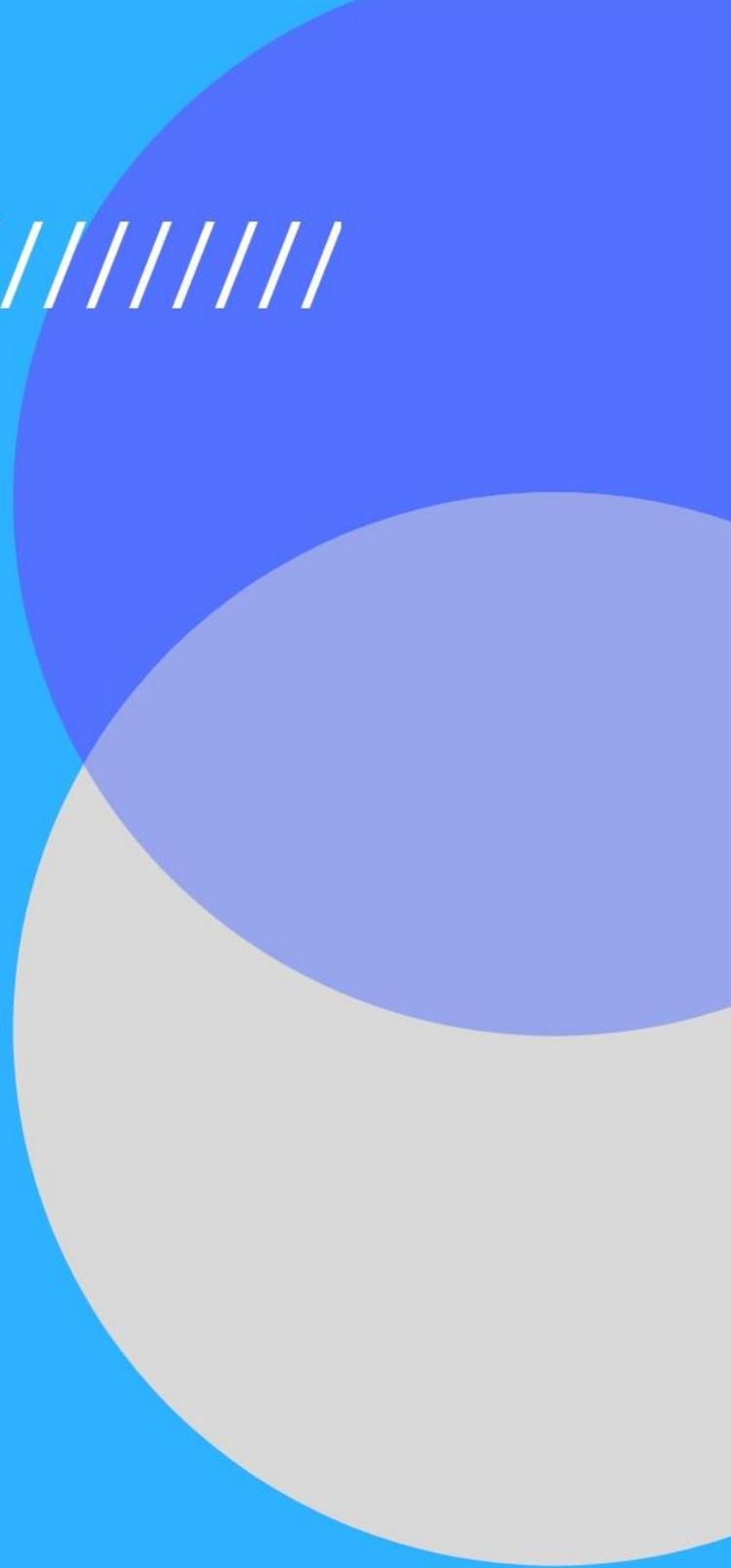




# HERE

APPLICATION DE  
CONFIRMATION DE PRESENCE



OMAR MHAIMDAT

MOHAMED AMINE BENBADA

UNIVERSITÉ INTERNATIONALE DE CASABLANCA

## Table des matières

<b>I - Problématique et solution .....</b>	<b>2</b>
<b>II - Diagrammes.....</b>	<b>4</b>
1. Diagramme de classe .....	4
2. Diagramme d'activité.....	5
3. Diagramme de cas d'utilisation.....	7
<b>III - Prototypage .....</b>	<b>11</b>
<b>IV - Partie QR Code .....</b>	<b>14</b>
1. Définition .....	14
2. Fonctionnement.....	14
3. Implémentation .....	15
<b>V - Technologies utilisées .....</b>	<b>16</b>
<b>VI - Création de la base de données.....</b>	<b>19</b>
<b>VII - Fonctionnement de l'application.....</b>	<b>22</b>
<b>VIII - Perspectives.....</b>	<b>25</b>
<b>IX - Webographie .....</b>	<b>27</b>

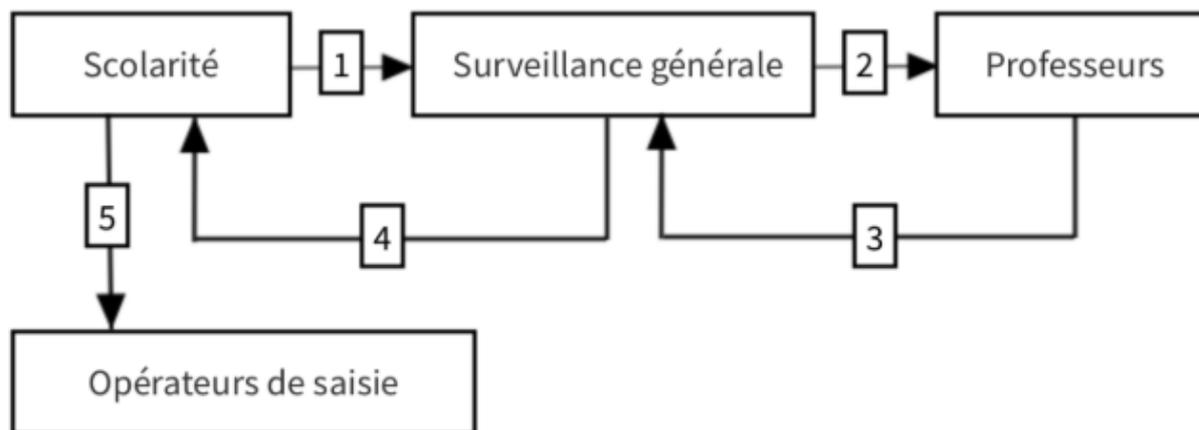
## I - Problématique et proposition de solution

**HERE** est une application mobile qui a pour objectif de confirmer la présence d'étudiants par le simple fait de scanner un QR code qui est affiché par le professeur durant la séance de cours ;

Pour bien comprendre pourquoi une telle solution est nécessaire, il faut tout d'abord poser la problématique existante au sein de notre établissement en posant la question : comment est-ce que l'élaboration des absences/présences se fait-elle ?

Il y a deux cheminements, on expliquera tout d'abord le plus utilisé.

Cela commence au sein de la scolarité, où la fiche d'absence de la semaine est déposée puis transférée à la surveillance générale. Avant chaque cours, le professeur doit tout d'abord récupérer celle-ci, avant d'établir qui est présent et absent durant une séance durant un horaire donné et un jour précis. Cela se termine avec une saisie manuelle des absences qui se fait à un intervalle de temps irrégulier.



Il naît de cette opération de saisie plusieurs problèmes dont on pourra citer :

- Possibilité d'erreur humaine lors de la saisie longue et laborieuse des absences ;
- Chaque professeur note les absences différemment, soit par un « a », d'autre par un simple tiret, faisant ainsi planer le doute au sein de l'esprit de l'opérateur qui effectue la saisie ;

L'autre cheminement possible étant la saisie directe des absences sur la plateforme MyUnivers. Or bien qu'elle soit pratique, celle-ci présente deux complications :

- Les listes ne sont pas actualisées, certains étudiants ne figurent pas sur les listes alors qu'ils devraient l'être ;
- Utilisation peu répandue parmi le corps enseignant ;

C'est ici que survient **HERE**. L'application a essentiellement pour but de formaliser et normaliser la saisie des présences et absences au sein d'un établissement, et ce comme suit :

- Pour chaque séance, un QR code est généré ;
- Durant la séance, le QR code est affiché par le professeur, l'étudiant doit scanner celui-ci pour valider sa présence ;
- La précédente opération n'est pas le seul critère de présence, puisqu'un critère de localisation est aussi pris en compte ;
- La validation de ces deux critères confirme ainsi la présence de l'étudiant ;

Une solution comme celle-ci présente plusieurs avantages :

- Très peu d'acteurs entrent en jeu
- Normalisation du système
- Plus de transparence
- Probabilité d'erreurs moins élevée
- Responsabilisation des étudiants
- Gain de productivité pour la scolarité et la surveillance générale

Premier concept



## II – 1. Diagramme de classe

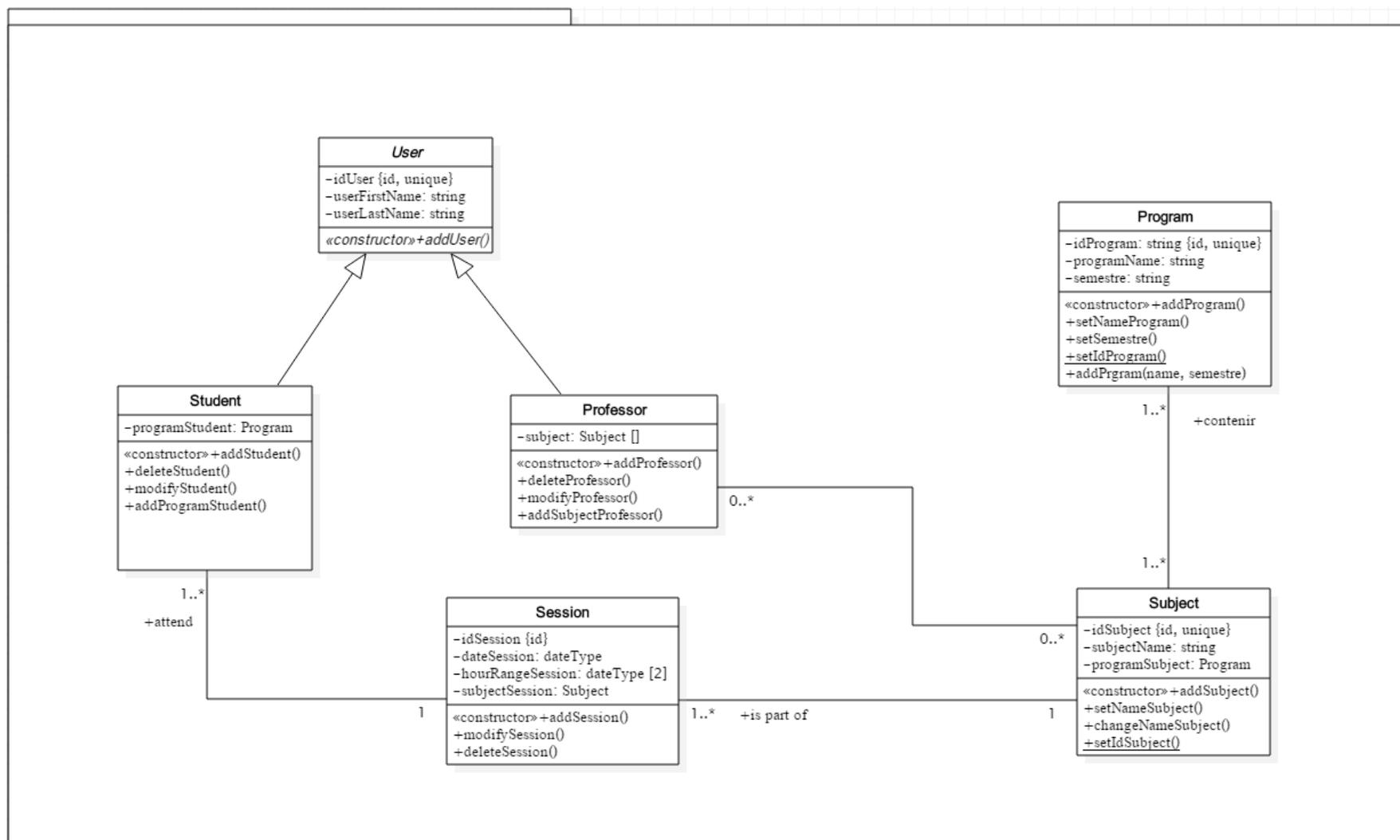


Fig 1. Diagramme de classe

## 2. Diagramme d'activité

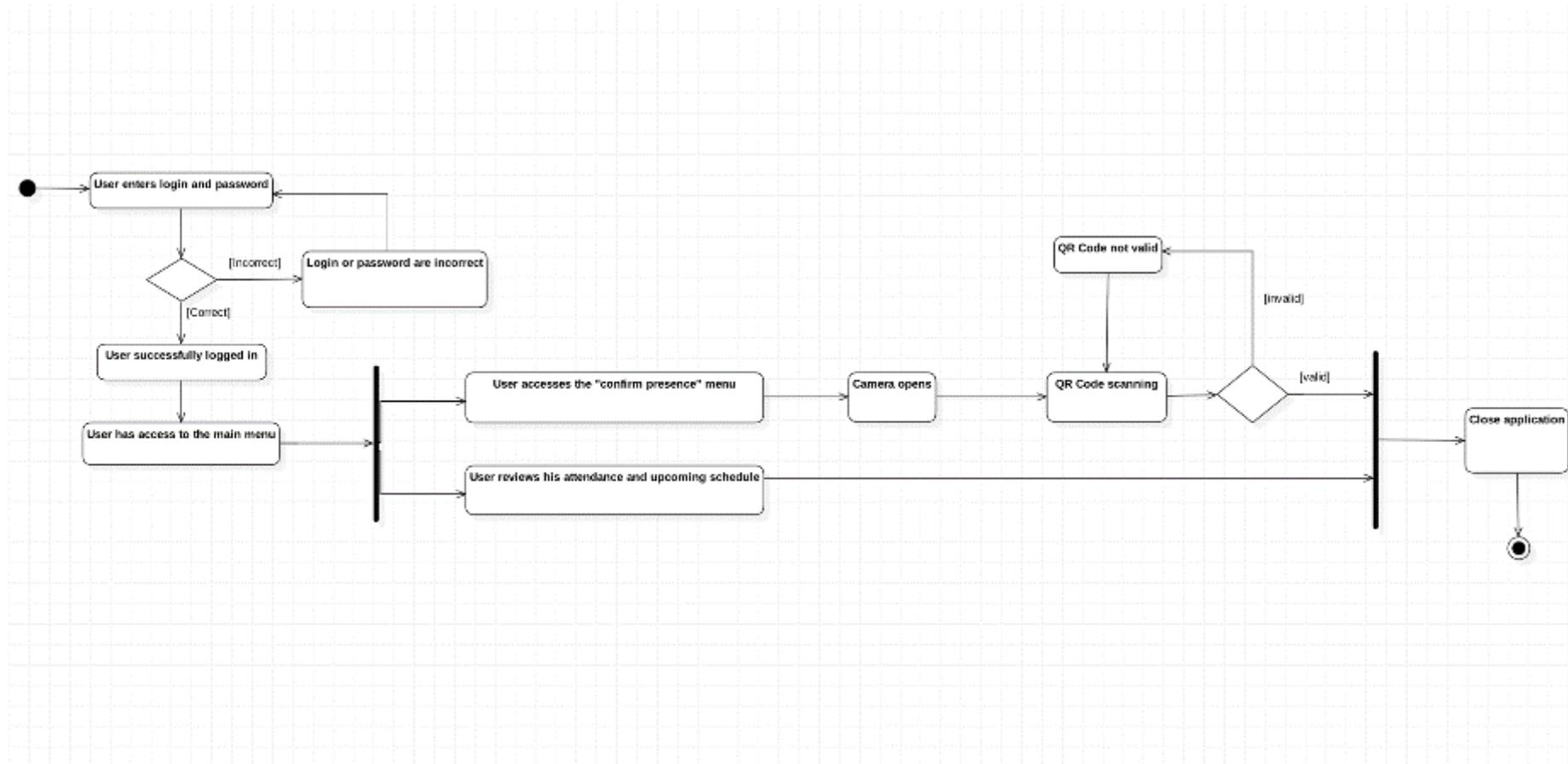
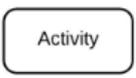
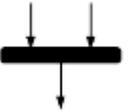
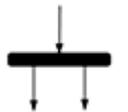


Fig 2. Diagramme d'activité

	Début du process
	Activité : indique les activités d'un modèle. Ces symboles sont les principaux composants d'un diagramme d'activité.
	Connecteur : démontre le sens du flot.
	Barre de synchronisation : combine deux activités pour les introduire au sein d'un seul flot où une seule activité est réalisée.
	Barre de partage du flot : divise une seule activité en deux activités concurrentes.
	Symbole de décision : représente une décision, a toujours au moins deux flots sortants, chacun accompagné d'un texte-condition.
	Fin du process

### 3. Diagramme de cas d'utilisation

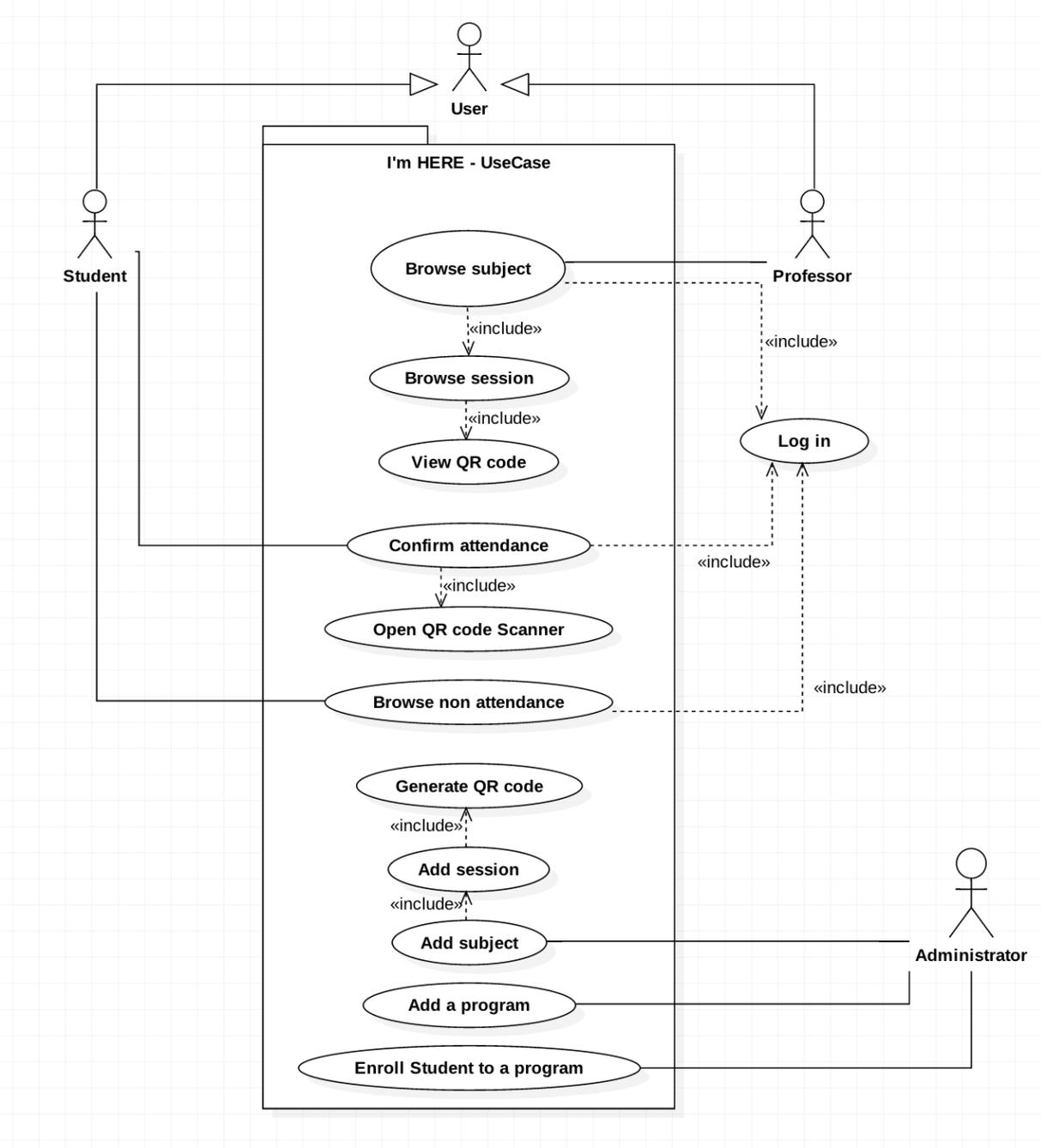
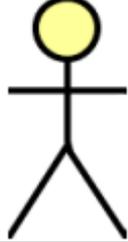
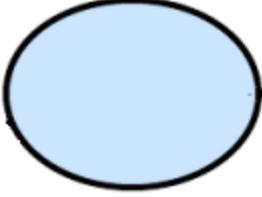


Fig 3. Diagramme de cas d'utilisation

	<p>Acteur : représente les personnes sujettes au cas d'utilisation.</p>
	<p>Cas d'utilisation : représente les différentes utilisations qu'un utilisateur peut avoir.</p>
	<p>Association : ligne entre acteur et cas d'utilisation.</p>
	<p>Frontière du système : « boîte » qui permet de déterminer la portée d'un système pour subvenir aux utilisations.</p>

➤ Activité : authentification

Cas d'utilisation 1.1 : **“Log in”**. Authentification de l'utilisateur

- **Objectif** : Permettre à l'utilisateur d'accéder à l'ensemble des fonctionnalités de l'application le concernant.
- **Acteurs concernés** : Un utilisateur pouvant être un étudiant ou un professeur.
- **Pré-conditions** : L'utilisateur doit tout d'abord être dans la base de données. Il doit aussi être déconnecté.
- **Post-conditions** : Aucunes.
- **Scénario nominal** : L'utilisateur a saisi les bonnes informations concernant le nom d'utilisateur et le mot de passe en étant présent dans la base de données.
- **Scénarios alternatifs** :
  - Cas 1 : L'utilisateur, bien qu'il soit inscrit dans la base de données, s'est trompé lors de la saisie du nom d'utilisateur et/ou du mot de passe et doit donc les saisir à nouveau.
    - Le système réaffiche le formulaire en lui indiquant de bien vouloir saisir à nouveau les informations d'authentification.
  - Cas 2 : L'utilisateur n'est pas inscrit dans la base de données.
    - Le système affiche un message demandant à l'utilisateur de contacter directement l'administrateur.

➤ Activité : gestion de la présence

Cas d'utilisation 2.1 : **“Browse subject”**. Affichage des matières associées au professeur.

- **Objectif** : Permettre au professeur d'accéder à l'ensemble des matières ainsi qu'aux séances associées à celle-ci.
- **Acteurs concernés** : Professeur uniquement.
- **Pré-conditions** : L'utilisateur doit être authentifié et reconnu comme professeur.
- **Post-conditions** : Aucunes.
- **Scénario nominal** : Le professeur accède aux matières qu'il enseigne ainsi qu'aux séances et leur QR Code.
- **Scénarios alternatifs** : aucuns.

Cas d'utilisation 2.2 : **“Confirm attendance”**. Confirmation de la présence de l'étudiant en scannant le QR Code.

- **Objectif** : Confirmer la présence ou pas lors d'une séance”
- **Acteurs concernés** : Etudiant uniquement.
- **Pré-conditions** : L'utilisateur doit être authentifié et reconnu comme étudiant.
- **Post-conditions** : Aucunes.
- **Scénario nominal** : L'étudiant accède à la séance, ce qui enclenche automatiquement le QR Scanner qui scanne sans action supplémentaire le code affiché, validant ainsi sa présence. LOCALIZATION
- **Scénarios alternatifs** :
  - Cas 1 : L'application n'a plus la permission d'accès à la caméra.
    - Le système affiche un message indiquant à l'utilisateur de bien vouloir autoriser l'accès à la caméra.

Cas d'utilisation 2.3 : **“Browse non attendance”**. Consultation des absences précédentes.

- **Objectif** : Afficher l'historique des absences.
- **Acteurs concernés** : Etudiant uniquement.
- **Pré-conditions** : L'utilisateur doit être authentifié et reconnu comme étudiant.
- **Post-conditions** : Aucunes.
- **Scénario nominal** : L'étudiant accède directement à ses absences antérieures.
- **Scénarios alternatifs** : aucuns.

### III - Prototypage

Le prototypage de l'application HERE suit une logique intuitive.

Après l'écran d'accueil, une page d'authentification s'affiche demandant à l'utilisateur de saisir son email ainsi que son mot de passe.

L'accès à l'application affiche un menu contenant le planning de journée, ainsi que les absences préalables.

Le menu « mes cours de la journée » permet d'accéder au cours que l'étudiant souhaite valider. L'appui sur une des séances non validées active automatiquement la caméra, permettant ainsi à l'étudiant de scanner le QR Code affiché par le professeur.







## IV - Partie QR Code

### 1. Définition

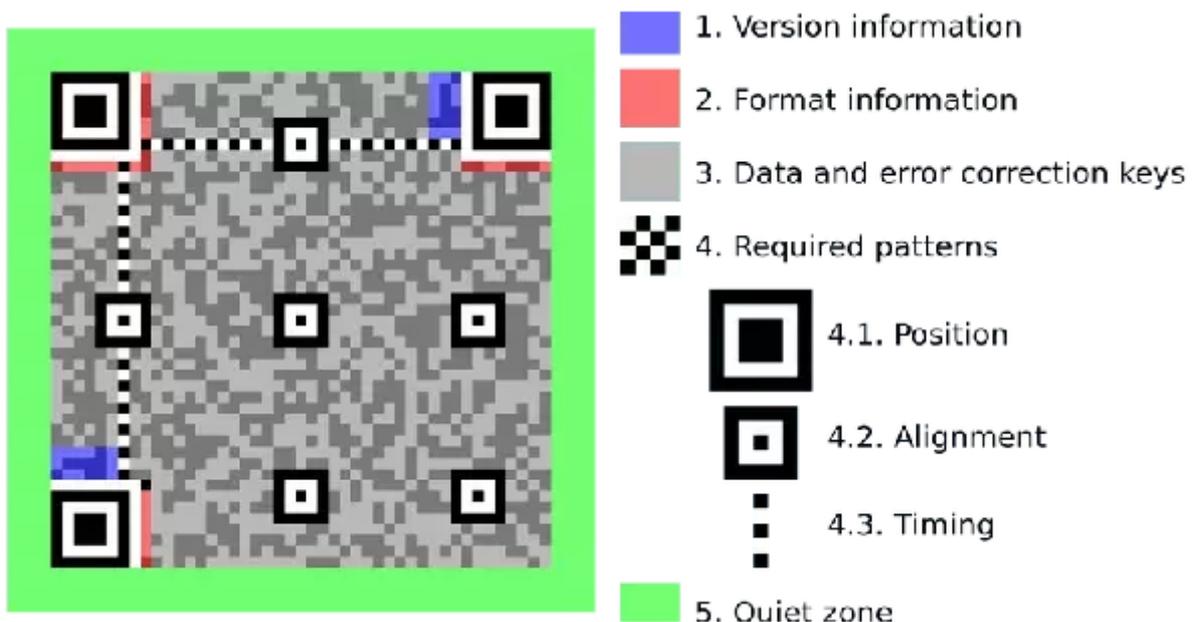
Avant d'aborder comment est-ce que la partie QRCode a été implémenté, on explique tout d'abord le principe d'un QR Code.

Un Code QR est un type de code-barres en deux dimensions constitués d'un carré noir sur lequel sont disposés des motifs noirs.

Ce code a été dénommé QR parce qu'il constitue une « quick response », ce qui est nécessaire dans notre cas. Le contenu de ce code peut être décodé rapidement à travers une caméra, permettant ainsi de déclencher les actions appropriées à notre application.

### 2. Comment ça marche ?

L'information (string) est encodée en suivant des motifs de noir et blanc, correspondant à des 0 et des 1. Les trois « bullseye » motifs sont utilisés pour déterminer le sens du code. Cela permet au décodeur de « lire » correctement le code QR.



On notera aussi que chaque Code QR a une capacité de correction d'erreur : si le code scanné est endommagé, il y a toujours possibilité d'envoyer la correcte information au décodeur. Il y a quatre niveaux de correction d'erreur, cela suit le tableau suivant :

QR Code Error	Correction capability
Level L	7%
Level M	15%
Level Q	25%
Level H	30%

Le niveau d'erreur est choisi dépendamment de l'utilisation ou l'environnement où le code sera affiché. Augmenter le niveau d'erreur revient à augmenter la taille du QR Code.

Dans la plupart des cas, le niveau M est celui sélectionné, mais on choisira ici le niveau H.

### 3. Implémentation

On rappelle que le QR Code qui servira à valider une séance sera généré à partir d'un string.

Pour répondre aux nécessités de notre application, nous avons décidé que la partie validation d'une séance suive le cheminement suivant :

1. Si la caméra est active, on effectue la lecture du Code QR affiché.
2. On compare le « string » du Code QR lu avec le « string » du Code QR propre à la séance, qui est elle propre à l'étudiant en question.
3. Si la comparaison est juste, la séance est validée, sinon on redemande une lecture.

On détaillera l'interaction de cette opération avec la base de données un peu plus tard.

## V - Technologies utilisées

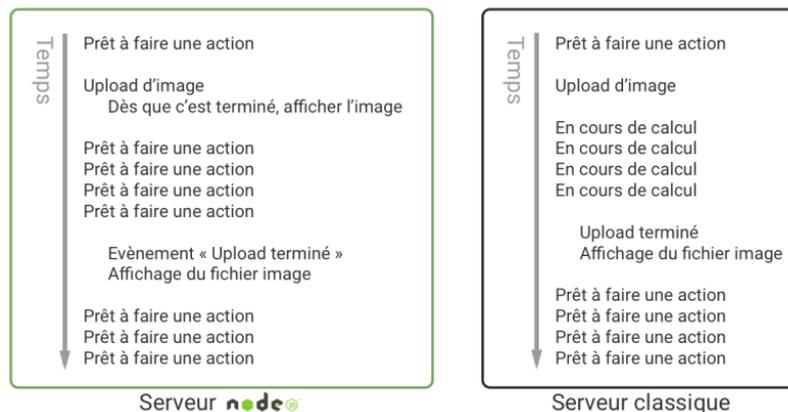
**Expo** : Expo est un outil gratuit et open-source construit autour de React Native, permettant de faciliter l'élaboration de projets native Android et iOS, utilisant JavaScript et React.



**Node.js** : Node.js est un environnement d'exécution multiplateforme JavaScript qui exécute du code JS côté serveur, alors qu'auparavant il était nécessaire d'utiliser du JavaScript pour le développement côté client frontend), ainsi qu'un autre langage côté serveur (backend). Node.js permet ainsi de coder les deux parties d'une application en JavaScript.



L'utilisation de Node.js en tant que serveur Web permet aussi de traiter un gros volume de requêtes simultanément de manière efficace.



Cet environnement met à disposition du codeur plus de 500.000 packages grâce au gestionnaire de packages npm. Celui-ci était initialement gestionnaire de packages spécifique à Node.js, mais a réussi à convaincre le monde JavaScript.

**React** : React est une librairie JavaScript flexible, efficace et déclarative pour établir des interfaces d'utilisateur (UI). Elle nous permet de réaliser de complexes UI à partir de plusieurs bouts de codes isolés appelés « Components ».

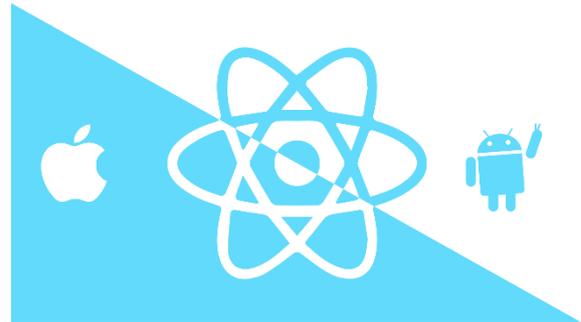


Le contenu créé sur React est référençable : grâce à l'utilisation d'un serveur Node, le code est généré côté client et côté serveur à la différence des autres frameworks JavaScript.

ReactJS est aussi extrêmement rapide, celui-ci crée son propre DOM virtuel où sont rattachés les composants créés. Cela nous apporte de la flexibilité et des performances exceptionnelles, puisque le DOM ne change que la partie qui a besoin d'être mise à jour à chaque exécution.

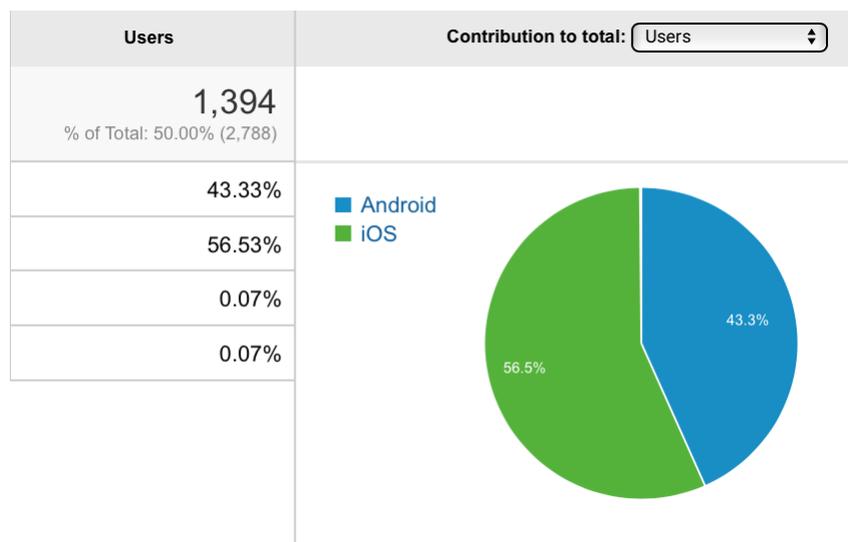
On pourra aussi ajouter que du code React est très simple à lire et à écrire, rendant ainsi celui-ci très facile à comprendre et manipuler.

**React Native** : Il s'agit de l'implémentation de React à Android et iOS. Il s'agit d'une librairie JavaScript développée par Facebook et Instagram, nous permettant de coder à la fois pour iOS et Android, entraînant ainsi un gain de productivité.



L'utilisation de React Native permet d'obtenir des applications mobiles natives très performantes. Elles sont fluides et responsives et offrent une très bonne « User Experience » et « User Interface »

Notre choix s'est tourné vers React Native puisqu'il offre la possibilité de développer une application cross-platform. Le besoin d'une telle application est justifié par les statistiques suivantes :



On constate que sur une période de 4 mois, les visiteurs du site <https://courses.ex-machina.ma> (site où les cours de différentes filières sont répertoriés) sont divisés en deux : 57% des étudiants ont un appareil iOS, tandis que 43% ont un appareil Android.

Il est donc inconcevable de créer une application pour l'une des deux plateformes seulement, d'où le choix d'un environnement de développement iOS/Android comme React-Native.

**Firestore Realtime Database** : Base de données hébergé dans le cloud. Les données sont stockées en tant que JSON et synchronisées en temps réel à chaque client connecté. S'agissant ici d'une application cross-platform, tous les clients partagent la même base de données et reçoivent automatiquement des mises à jours contenant les données les plus récentes.



Les packages utilisés sont résumés dans l'entête de **package.json**, qui se présente comme suit :

```
1 {
2   "main": "node_modules/expo/AppEntry.js",
3   "private": true,
4   "dependencies": {
5     "body-parser": "^1.18.3",
6     "expo": "^27.0.1",
7     "express": "^4.16.3",
8     "firebase": "^5.2.0",
9     "mysql": "^2.15.0",
10    "react": "16.3.1",
11    "react-native": "https://github.com/expo/react-native/archive/sdk-27.0.0.tar.gz",
12    "react-native-camera": "^1.1.4",
13    "react-native-elements": "^1.0.0-beta5",
14    "react-navigation": "^2.6.2"
15  }
16 }
```

**StarUML** : Logiciel de modélisation UML open-source. Il gère la plupart des diagrammes spécifiés dans la norme UML 2.0 comme ceux illustrés auparavant : diagrammes de classe, diagramme d'activité, diagramme de cas d'utilisation ou autres.



## VI - Création de la base de données

Pour l'implémentation de la base de données, notre choix s'est d'abord tourné vers Express.js couplé à MySQL.

Express.js est un Framework pour construire des applications web basées sur Node.js. Celui-ci, avec MySQL, nous permet de connecter des bases de données à l'application.

Malheureusement suite à une erreur rencontrée, nous nous sommes trouvés contraints à changer de méthode de création de la base de données.

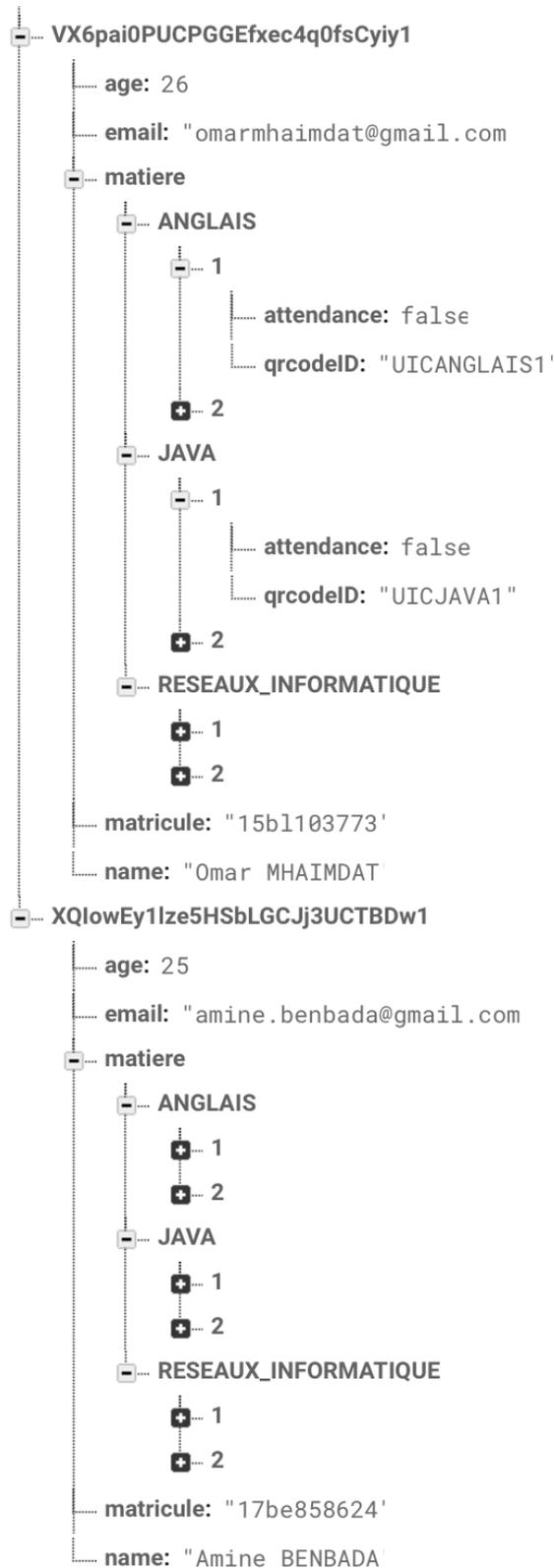
Nous avons finalement opté pour Firebase Realtime Database. On rappelle que celui-ci est un service qui permet de stocker et synchroniser des données dans le cloud NoSQL.

La structure de notre base de données se présente comme suit :



Naturellement le choix d'une telle structure implique une quantité importante de données pour un nombre important d'étudiants ou un nombre important de séances, mais le fait que la base de données est NoSQL nous permet d'avoir un traitement rapide et une flexibilité dans la gestion des données.

La structure détaillée de la base de donnée est donc sous la forme suivante :



On remarque que chaque utilisateur est identifié par un User ID, qu'on retrouve ici :

Identifiant	Providers	Created	Signed In	User UID ↑
amine.benbada@gmail.com	📧	Jul 9, 2018	Jul 10, 2018	XQIowEy1Ize5HSbLGCJj3UCTBDw1
omarmhaimdat@gmail.com	📧	Jul 8, 2018	Jul 9, 2018	XSU0PdI5NSWI89PgF8Rd3NEQU...

L'UID est généré automatiquement à chaque ajout d'un utilisateur, celui-ci est unique pour chaque utilisateur, et suffit à identifier celui-ci en le reliant à un email ainsi qu'un mot de passe. Cet UID est « world wide unique », et nous servira à identifier chaque utilisateur, cela représente deux avantages :

- Cela nous offre la possibilité de changer l'email de l'utilisateur à son souhait, sans changer ses attributs puisque l'UID est ce qui le définit réellement.
- L'administrateur ne peut pas modifier cet UID ainsi que le mot de passe propre à celui-ci. Il s'agit donc d'un niveau de sécurité supplémentaire pour l'utilisateur.

Détaillons maintenant la structure de notre base de données :

- Un utilisateur est créé à travers son UID. Il est caractérisé par son nom, âge, matricule, email ainsi que les matières auxquelles il devrait assister.
- Chaque matière est caractérisée par un certain nombre de séances. On précise que chaque séance possède deux « child nodes » :
  - Attendance : qui est par défaut défini à « false ».
  - qrcodeID : contient le string qui devra être comparé au QR Code lu par l'utilisateur.

Afin de connecter notre base de données à l'application, il suffit d'ajouter le snippet suivant :

### Add Firebase to your web app

Copy and paste the snippet below at the bottom of your HTML, before other script tags.

```
<script src="https://www.gstatic.com/firebasejs/5.2.0/firebase.js"></script>
<script>
  // Initialize Firebase
  var config = {
    apiKey: "AIzaSyCEkapfH2WmydCkU0Dz1TkVuN5Leqs70Ws",
    authDomain: "here-c7cdc.firebaseio.com",
    databaseURL: "https://here-c7cdc.firebaseio.com",
    projectId: "here-c7cdc",
    storageBucket: "here-c7cdc.appspot.com",
    messagingSenderId: "207796952812"
  };
  firebase.initializeApp(config);
</script>
```

Check these resources to learn more about Firebase for web apps:

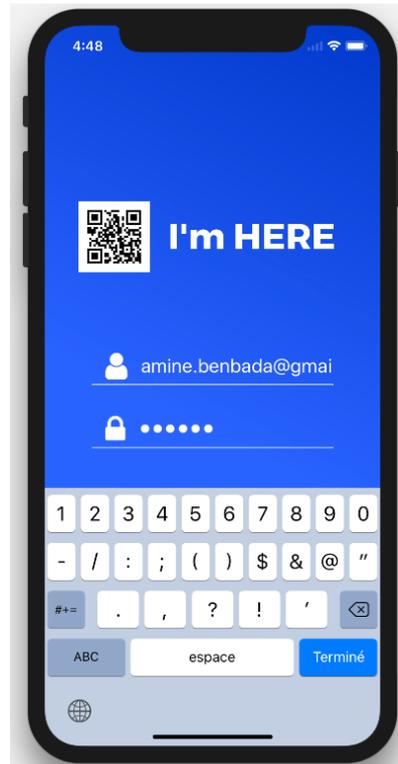
- [Get Started with Firebase for Web Apps](#)
- [Firebase Web SDK API Reference](#)
- [Firebase Web Samples](#)

## VII - Fonctionnement de l'application

Une fois la base de donnée créée et reliée, nous pouvons à présent établir une première expérience d'utilisation.

Nous essaierons de valider pour l'étudiant « Amine BENBADA » la 1<sup>ère</sup> séance de JAVA.

L'écran d'accueil nous demande tout d'abord de saisir un email ainsi qu'un mot de passe. Une erreur de saisie entraîne l'affichage d'une alerte redemandant à l'utilisateur de saisir à nouveau. Si les informations d'identification sont correctes, nous avons accès à la caméra.

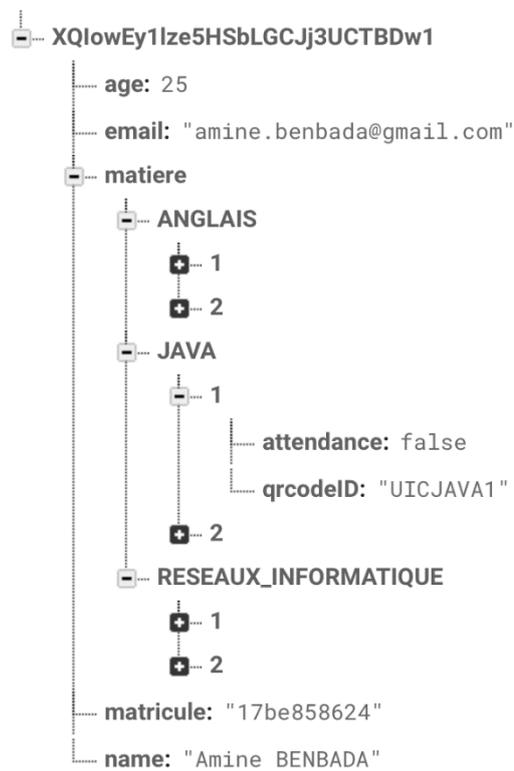


Si les informations d'identification sont correctes, nous avons accès à la caméra.

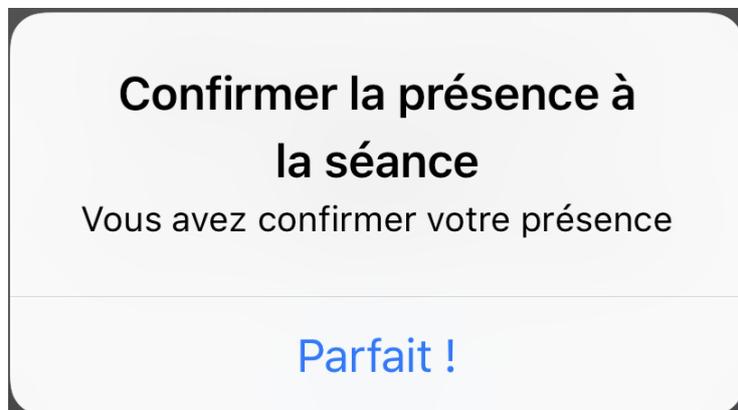
A l'ouverture de la caméra, il est donc nécessaire de scanner un QR Code dont le string correspond à celui associé à la séance. La 1<sup>ère</sup> séance de JAVA est associée à un QR Code dont le string est UICJAVA1, qui est sous la forme suivante :



Avant de scanner ce code, le node « attendance » est défini à « false » par défaut.



Si le QR Code scanné correspond au qrcodeID de la séance, l'alerte suivante apparaît :



Cette validation modifie « attendance » en la mettant à « true ».

The screenshot shows a user profile for 'XQlowEy1Ize5HSbLGCJj3UCTBDw1'. The profile includes fields for age (25), email ('amine.benbada@gmail.com'), matricule ('17be858624'), and name ('Amine BENBADA'). A 'matiere' section is expanded to show a list of subjects: ANGLAIS, JAVA, and RESEAUX\_INFORMATIQUE. Each subject has a list of items with plus signs. The 'JAVA' subject's first item is expanded to show 'attendance: true' and 'qrcodeID: "UICJAVA1"'. The 'matiere' section has a '+ x' button.

```

XQlowEy1Ize5HSbLGCJj3UCTBDw1
├── age: 25
├── email: "amine.benbada@gmail.com"
├── matiere + x
│   ├── ANGLAIS
│   │   ├── 1
│   │   └── 2
│   ├── JAVA
│   │   ├── 1
│   │   │   ├── attendance: true
│   │   │   └── qrcodeID: "UICJAVA1"
│   │   └── 2
│   └── RESEAUX_INFORMATIQUE
│       ├── 1
│       └── 2
├── matricule: "17be858624"
└── name: "Amine BENBADA"

```

## VIII - Perspectives

L'application en son état actuel permet de confirmer la présence d'un étudiant proprement, mais il est toujours possible d'améliorer « HERE ». Nous proposons ainsi les améliorations suivantes :

- Ajouter un menu permettant de consulter les absences préalables, offrant ainsi une visibilité globale de la situation de l'étudiant.
- Ajouter la possibilité pour l'étudiant de consulter le planning du jour en ajoutant un menu « séances du jour ».
- Ajouter une section « contact » pour signaler tout problème lié à une absence ou une erreur d'identification.
- Refonte de la base de données afin qu'elle soit conforme au modèle conceptuel défini, soit en passant par un modèle SQL (Express.js) ou un nouveau modèle NoSQL avec Firebase.

On notera aussi que le choix combinant Firebase et NoSQL ne freinera pas l'expansion en nombre d'utilisateurs de l'application. En supposant qu'une classe entière tente de valider sa présence, une base de données SQL mettra beaucoup plus de temps que la solution adoptée.

## **IX - Bibliographie / Webographie**

Getting started – React Native <https://facebook.github.io/react-native/docs/getting-started>

Quick start – Expo Documentation <https://docs.expo.io/versions/v28.0.0/>

Overview – React Native Elements <https://react-native-training.github.io/react-native-elements/docs/1.0.0-beta4/overview.html>

Getting started – React Navigation (v2) <https://reactnavigation.org/docs/en/getting-started.html>

Documentation – Firebase <https://firebase.google.com/docs/>

Github – React-native-Camera <https://github.com/react-native-community/react-native-camera/blob/master/docs/RNCamera.md>

ECMAScript 6: New Features: Overview and Comparison <http://es6-features.org/#Constants>

JavaScript | MDN <https://developer.mozilla.org/bm/docs/Web/JavaScript>

Getting Started – React <https://reactjs.org/docs/getting-started.html>