

IUT2 INFO - Université Grenoble Alpes

M3301 - Itération 1  
Compte-Rendu Groupe 8

---

## **Logiciel de suivi de cours**

---

GUITON Clément  
CEREN Carlos  
DUPLESSIS Thomas  
FRAGNON Adrien  
LAROUCSI Abel  
VIAUD Nathan

# Table des matières :

<b>Cadrage du projet</b>	<b>2</b>
Contexte et Objectifs	2
Analyse du terrain et de l'existant	2
Contraintes et Risques	3
<b>Expression des besoins</b>	<b>4</b>
Identification des besoins fonctionnels	4
Priorisation des besoins fonctionnels	4
Mise en avant des critères qualité logicielle	4
Mise en avant des critères ergonomiques	5
<b>Solution</b>	<b>5</b>
Solution proposée	5
Réduction de la bande passante	5
Annotation	5
Question des élèves	5
Quizz en direct	5
Présentation des solutions techniques	6
Solution générale	6
Interface IHM	6
Accueil	6
Vue lecteur	6
Vue présentateur	7
Choix technologique	7
Nos critères	7
Choix possible	7
Choix final (Java EE vs Node.JS)	8
Environnement	9
Protocole réseau	9
HTTP	10
WebSocket	10
Interface	10
Conclusion	10
Gestion du travail	11
Discord	11
Trello	11
Tests	11
Répartition des tâches	11
Description d'un modèle de développement	12
<b>Conception</b>	<b>13</b>
Diagramme de classe	13
Diagramme de cas d'utilisation	13
Phase de connexion	13
Présentation	13
Échanges serveur-client	13
Mise en place	14
Lecteur PDF	14
Bouton Lever la Main	15
Fonctionnalité Quizz	15
Communication Client - Serveur	16
Synchronisation page	16
Connexion lecteur	16
Droit vis-à-vis de code	17
<b>Réalisation</b>	<b>17</b>
<b>Annexe</b>	<b>18</b>

# 1. Cadrage du projet

## 1.1. Contexte et Objectifs

Pendant l'année scolaire 2019-2020, le monde a fait face de manière spontanée à une pandémie mondiale qui a obligé le système éducatif français à prendre la décision d'annuler les cours en présentiel. Cet événement a obligé l'administration à prendre une décision au plus vite pour s'assurer que les étudiants continuent le suivi de leur filière. La solution choisie a été des cours en distanciel. Ceci était un réel défi pour les professeurs afin de trouver des logiciels adaptés à chaque enseignement. En effet ceux-ci devaient répondre à plusieurs critères : la lecture d'un fichier depuis la machine professeure jusqu'aux machines étudiantes ainsi que la communication écrite ou vocale entre les enseignants et les étudiants - pour ne citer que les plus importants.

Ainsi, plusieurs solutions ont été prises mais ayant chacune des défauts non négligeables. Dans l'initiative de notre projet, nous allons chercher comment est-il possible d'assurer les enseignements à distance tout en prenant en compte des conditions de connexion variables des concernés soit le corps enseignant et les étudiants.

## 1.2. Analyse du terrain et de l'existant

Lors du confinement, nous avons principalement utilisé Discord pour les cours à distance :

- Cette application apportait un chat direct écrit et oral ainsi qu'une possibilité de hiérarchiser les membres (enseignants / étudiants) pour les catégoriser selon leur groupes de TD/TP. Cependant le principal problème était l'instabilité des connexions internet (côté professeur ou élève). Cependant, cette plateforme a un inconvénient majeur pour le suivi des cours : il diffuse un flux vidéo.

Malheureusement, la connexion de certains étudiants et de certains enseignants n'était pas assez importante pour supporter ce flux ce qui pouvait provoquer des coupures, voire rendre le texte illisible à l'écran.

Certains enseignants utilisaient seulement Discord pour la communication vocale, nous devions donc suivre un PDF sur notre machine.

Cependant il était parfois compliqué de suivre les cours : si nous voulions revenir en arrière sur le cours ou si nous avions tout simplement perdu notre attention pendant quelques instants, il devenait difficile de retrouver le fil du cours.

Un autre inconvénient des logiciels de chat en ligne est la capacité des élèves à poser des questions. En effet, pour cela les élèves doivent soit poser une question dans le chat qui risque de ne pas être remarqué dans le flot continu de texte, ou alors intervenir à l'oral et couper le professeur, ce qui peut déranger le déroulement du cours ou freiner les élèves les plus timides.

Nous avons pensé à Google Docs qui est un logiciel de traitement de texte collaboratif en ligne :

- Ce site a pour fonctionnalité intéressante un curseur permettant de voir à quel endroit du document un utilisateur se trouve. Il s'agit d'une fonctionnalité que nous reprendrons, mais cet outil n'est pas vraiment adapté pour une présentation et ne permet pas le suivi du fil du cours.

Dans ces deux logiciels, des difficultés à poser des questions sont remarquables. Le professeur ne peut pas poser de questions aux élèves sans qu'il y ait de trop nombreuses réponses dans le salon textuel, rendant impossible à l'enseignant d'avoir un réel compte rendu des résultats.

Nous avons ensuite découvert BigBlueButton :

- Ce site ressemble beaucoup à l'idée de base que nous avons de notre application. En effet, cette application web est très complète, apporte des fonctionnalités très intéressantes telles que les quizz, l'option "lever la main", si intéressantes que nous avons décidé de nous en inspirer pour le développement de notre projet. Cependant BigBlueButton a le même problème que Discord : Il diffuse un flux vidéo.

De plus, plusieurs améliorations nous ont été proposées par nos enseignants coachs, comme par exemple la possibilité pour les professeurs de préparer les quizz à l'avance et de choisir quand les lancer.

### 1.3. Contraintes et Risques

Ce projet doit respecter quelques contraintes. En effet, nous avons premièrement un délai limité pour réaliser notre projet. Nous avons aussi une limite technologique puisqu'un de nos buts est de fournir un service qui ne nécessite pas un débit trop élevé. Il faut aussi que l'application puisse tourner sur des ordinateurs peu puissants afin de s'adapter à tous les étudiants. Il y a aussi une contrainte de développement puisque nous sommes limités par les technologies présentes à l'IUT et à ce qu'elles ont à nous proposer. Une grande stabilité doit aussi être présente puisque le domaine traité ici est l'enseignement, un problème technique impactant l'apprentissage n'est ici pas envisageable.

Le projet comporte peu de risques, premièrement, un premier est lié à l'augmentation des informations puisque cela pourrait surcharger visuellement le client. Nous pouvons aussi craindre une augmentation de la bande passante nécessaire si trop de fonctionnalités futiles sont ajoutées.

## 2. Expression des besoins

### 2.1. Identification des besoins fonctionnels

Dans ce projet, nous trouvons plusieurs besoins fonctionnels. Ceux-ci doivent résoudre les problèmes mentionnés précédemment. Ces besoins sont les suivants:

- Accéder à l'application sans problèmes de connexion: cela peut bien résoudre le problème le plus courant entre étudiants et enseignants.
- Téléchargement du pdf en fin de cours.
- L'ajout des notes des professeurs: ainsi les professeurs pourront ajouter des notes en direct dans le pdf.
- Quizz: la possibilité des enseignants de pouvoir lancer des quizz préparés à l'avance ou même pendant le cours pour savoir si la notion abordée est bien comprise.
- Bouton "poser une question": de cette manière les étudiants pourront poser des questions et les enseignants pourront bien la voir.

## 2.2. Priorisation des besoins fonctionnels

Tous les besoins énumérés dans le paragraphe précédent ont été priorisés. La liste est la suivante :

1. Accéder à l'application sans aucun problème de connexion
2. L'ajout des notes de la part des enseignants
3. Télécharger le pdf à la fin du cours
4. Bouton lever la main
5. Quizz

## 2.3. Mise en avant des critères qualité logicielle

Nous avons choisi des critères qualités concernant notre projet qui nous tenait à coeur, pour commencer nous voulons principalement nous concentrer sur les critères de facilité d'usage comme la facilité d'apprentissage et de compréhension, nous voulons qu'il y ait le moins de temps de compréhension possible, que tout le monde puisse lancer l'application et l'utiliser sans problème. L'exploitabilité est également un critère important, il faut que l'utilisateur puisse sans investissement sérieux comprendre toutes les possibilités qui lui sont offertes. Enfin, la stabilité du système est primordiale afin de garantir une expérience optimale à l'utilisateur.

## 2.4. Mise en avant des critères ergonomiques

En ce qui concerne les critères ergonomiques nous voulons nous concentrer sur la distinction et le groupement des informations afin que l'utilisateur comprenne rapidement quel élément de l'interface est relié à quelle fonctionnalité. Nous souhaitons également insister sur la lisibilité et la diminution de la densité informationnelle afin que l'utilisateur ne se perde pas dans des données superflues et puissent interagir avec l'interface le plus efficacement possible.

## 3. Solution

### 3.1. Solution proposée

#### 3.1.1. Réduction de la bande passante

Pour résoudre les problèmes évoqués précédemment, voici les solutions auxquelles nous avons pensé. Chaque utilisateur possède son propre lecteur pdf exécuté dans son navigateur. Un pdf contenant le cours est transmis à chacun par des moyens similaires à Chamilo (le pdf est disponible à la lecture sans pour autant avoir été téléchargé localement) puis, lorsque le présentateur change de page, le numéro de celle-ci est transmis au serveur qui le renvoie à chaque "lecteur". Cette technique limite donc le transfert de données à de simples numéros de page (pour simplifier) au lieu de transmettre un flux vidéo complet qui est beaucoup plus gourmand en terme de bande passante.

#### 3.1.2. Annotation

En plus de cette idée qui résout le problème de base d'inégalité de connexion, s'ajoutent d'autres fonctionnalités nécessaires à un enseignant pour faire cours. La première étant la possibilité d'ajouter des commentaires sur le document. Ces commentaires sont transmis à chaque lecteur **en direct**. Le document annoté reste à disposition au téléchargement pour les étudiants n'ayant pas pu se connecter lors du cours, ainsi ils ne prendront pas de retard sur les explications de l'enseignant.

#### 3.1.3. Question des élèves

Une fonctionnalité qu'on trouve intéressante à ajouter est de donner la possibilité aux étudiants de poser une question à l'enseignant en envoyant une notification assez visible. Cela évitera de couper la parole au professeur ou que la question soit oubliée dans le chat.

#### 3.1.4. Quizz en direct

Une autre fonctionnalité qui pourrait être utile non seulement pour les étudiants, mais également pour les enseignants, est la possibilité de pouvoir créer à l'avance des quizz qui pourront être lancés pendant le cours. Ainsi les résultats de ces petits QCM seront envoyés en direct aux enseignants, afin de voir si la notion abordée est bien assimilée par les étudiants, ou s'il est nécessaire de revenir dessus. De même les étudiants pourront voir s'ils ont bien compris ce qui a été expliqué. Les QCM peuvent être préparés à l'avance ou, si le professeur le voit nécessaire, être faits rapidement pendant le cours et le lancer.

### 3.2. Présentation des solutions techniques

#### 3.2.1. Solution générale

Pour résoudre les problèmes évoqués précédemment, voici nos solutions techniques. Chaque utilisateur possède son propre lecteur pdf exécuté dans son navigateur grâce à un script javascript. Un pdf contenant le cours est transmis à chacun et ensuite, lorsque le présentateur change de page, le numéro de celle-ci est transmis au serveur qui le renvoie à chaque "lecteur". Cette technique limite donc le transfert de données à de simples numéros de page (pour simplifier) au lieu de transmettre un flux vidéo complet qui est

beaucoup plus gourmand en terme de bande passante. Pour mettre en œuvre cette solution, plusieurs technologies sont nécessaires. Côté serveur, un serveur HTTP qui gère les connexion des utilisateurs, qui leur transmet les bonnes pages html ainsi qu'un serveur websocket qui fait la liaison entre les clients en temps réel.

Côté client, un script Javascript est exécuté dans le navigateur. Ce script permet de communiquer au serveur via websocket, et gère l'interface utilisateur propre à l'application. gourmand en terme de bande passante. Pour mettre en œuvre cette solution, plusieurs technologies sont nécessaires.

### 3.2.2. Interface IHM

#### 3.2.2.1. Accueil

En arrivant sur le site, l'utilisateur verra la page d'accueil qui se présente sous une forme assez simple avec un champ de texte permettant de remplir son pseudonyme ainsi que 2 options: rejoindre un salon ou créer un salon. Pour rejoindre un salon l'utilisateur aura simplement à renseigner le code du salon qu'il souhaite rejoindre. Lors de la création d'un salon, l'utilisateur aura juste à joindre le document sur lequel il souhaite travailler puis à partager le code au lecteur.

C.F [Annexe 1](#)

Il y a aussi un accueil alternatif lorsqu'on rentre le code du salon dans l'url pour le cas où le présentateur inviterais le lecteur avec un lien. Dans ce cas en arrivant le lecteur aura juste un champ pseudo à remplir avant de rejoindre le salon renseigné dans l'url.

#### 3.2.2.2. Vue lecteur

Au niveau de l'interface pendant un cours, dans la fenêtre centrale sera affiché le pdf du cours, les lecteurs pourront se déplacer librement sur le document comme sur un pdf classique à la différence près que celui-ci peut être annoté par le présentateur. Dans le salon il y a deux rôles différents à savoir : les lecteurs qui représenteront les élèves et qui auront simplement le droit de regarder le cours sans pouvoir l'éditer et le présentateur (de base celui qui crée le salon) qui représente le prof qui a le droit d'annoter le pdf. Le système de synchronisation des pages sera basé sur la vue du présentateur donc il ne peut en avoir qu'un seul.

Ensuite nous allons retrouver au niveau de la barre de défilement de l'utilisateur un curseur rouge représentant la scrollbar du présentateur pour connaître l'avancée du cours.

Dans une fenêtre sur la droite de l'écran nous trouverons en bas à droite un bouton permettant de rejoindre rapidement la position du présentateur et une case à cocher pour activer la synchronisation de la position de la scrollbar avec celle du présentateur. Un peu au-dessus de ces deux boutons nous trouverons un champ texte accompagné d'un bouton "envoyer" correspondant au bouton question, la question devra être écrite dans ce champ texte et après l'envoi la question apparaîtra sur la partie haut droite de la fenêtre sous la forme d'une case de notification.

Il y aura une dernière fenêtre sur la gauche de l'écran avec la liste des noms des utilisateurs présents dans le salon. Ces deux fenêtres latérales pourront être réduites pour

laisser plus de place à la fenêtre centrale.

C.F [Annexe 2](#)

### 3.2.2.3. Vue présentateur

La vue présentateur sera assez similaire à la vue lecteur à la différence que les éléments de synchronisation du document pdf seront absents ainsi que le bouton “poser une question”, et dans la partie haute de la fenêtre de gauche sera ajouté des éléments permettant de gérer le système de quizz.

C.F [Annexe 3](#)

## 3.3. Choix technologique

### 3.3.1. Nos critères

Pour choisir quelle technologie utiliser, nous avons établi une liste de nos attendus pour une technologie. Ces critères très différents sont ici destinés à nous aider à choisir la technologie la plus adaptée à la réalisation de notre projet. Voici la liste de ces critères:

- Langage abordable par des étudiants en 2e année de DUT INFO (haut niveau / bas niveau)
- Installation et configuration rapide
- Compatible avec les technologies mises à disposition à l'IUT 2 Grenoble.
- OpenSource
- Documentation complète
- Scalabilité
- Vitesse de développement
- Compétence du groupe

### 3.3.2. Choix possible

Nous avons réunis les différentes technologies web que nous avons utilisées à l'iut ou d'on nous avons entendu parlé. L'objectif est ici de faire une présélection de technologie pour pouvoir comparer plus finement les plus pertinentes.

Pour ce faire nous avons évalué sur 5 points ces technologies sur les différents critères définis précédemment. Ces notes sont subjectives et sont issues de nos expériences personnelles et de nos recherches. Le but n'est pas ici de définir quelle est la meilleure technologie mais simplement laquelle nous allons utiliser.

Critères\Technologie	PHP	Node.JS	Java EE	C++	Ada
Abordable	3	4	2	1	1
Installation et configuration rapide	2	5	3	2	2
Compatible avec les technologies de l'IUT	4	3	3	1	1
OpenSource	5	5	5	5	5
Documentation complète	4	5	3	4	1



Scalabilité	3	3	5	4	3
Vitesse de développement	3	5	3	2	1
Compétence du groupe	2	1	4	3	3
Total	26	31	28	22	17

Nous avons donc choisis de comparer Node.JS et Java EE plus précisément

### 3.3.3. Choix final (Java EE vs Node.JS)

Pour faire notre choix final entre Java EE et Node.JS, nous avons séparé le projet en différents domaines pour choix plus précisément.

Voici une introduction à Java EE et à Node.JS :

Java EE signifie Java Enterprise Edition. Il s'agit d'un ensemble d'extension à Java SE (Standard Edition). L'objectif principal de Java EE est de fournir des outils pour le développement de logiciels professionnels à grande échelle et pour le web.

La fonctionnalité qui nous intéresse le plus est le développement de serveur en java avec une intégration avec des serveurs HTTP comme Tomcat ou Glassfish.

Java EE est une extension de Java qui est un langage que nous avons étudié en cours et dont nous avons de bonnes bases. L'utilisation de son architecture orientée objet le rend très adapté pour le développement de logiciels complexes.

Node.JS est un moteur JavaScript. Il permet d'utiliser le langage JavaScript pour le développement de serveur et donc d'utiliser la même technologie pour le côté serveur et le côté client. Il est développé à partir du moteur JavaScript V8 utilisé notamment par Google Chrome, ce qui le rend très rapide. De plus, son fonctionnement par boucle d'événement et de thread unique le rend très simple d'utilisation car l'utilisateur n'a pas à gérer la gestion des requêtes en parallèle. Pour ces raisons Node.JS est parfait pour le développement d'applications web qui nécessite de transmettre des données en temps réel, ce qui est notre cas. JavaScript n'est cependant pas un langage conçu pour l'orienté objet, même s'il est possible de le simuler. Il est de ce fait moins adapté au développement d'une application à grande échelle. De plus, nous n'avons pas appris le javascript à l'iut et nous devons donc nous former.

#### 3.3.3.1. Environnement

Les environnements de développements entre NodeJS et Java EE sont très différents :

Côté exécution, Java EE ne gère pas le protocole HTTP, et demande donc un logiciel serveur tierce comme Tomcat ou Glassfish. L'installation et l'utilisation de ce logiciel externe complexifie grandement l'installation et l'utilisation de Java EE. Node.JS, a contrario, embarque tout le système et est donc très simple d'utilisation. Il est même possible, grâce à une extension, d'avoir un "hot reload", une fonctionnalité qui permet de relancer automatiquement le serveur lors d'un changement dans le code. Cette fonctionnalité peut faire gagner beaucoup de temps au final car c'est une action très fréquente.

La complexité de l'environnement Java EE nécessite l'utilisation d'une IDE intégrant ses dépendances et compatible avec le serveur web. L'utilisation de cette IDE ajoute un apprentissage à tous ceux qui ne sont pas familiarisés avec.

A l'inverse, le développement avec Node.JS peut se faire avec n'importe quel éditeur de texte, et donc d'utiliser des logiciels avec lesquels nous sommes déjà familiers.

Un autre critère important pour la comparaison de l'environnement de développement est la gestion des dépendances.

Node.JS étant une des technologies les plus populaires actuellement, il existe une énorme quantité de bibliothèques utilisables. Des frameworks complets qui simplifient grandement le développement d'application sur quasiment tous ses aspects aux petites bibliothèques pour gérer des actions très précises en passant par des frameworks de tests, tout y est. De plus l'immense popularité de Node et plus largement de javascript permet d'avoir des guides détaillés pour toutes les bibliothèques populaires en plus de la doc très bien rédigée.

Son gestionnaire de paquet, NPM, rend l'installation de dépendance très simple, en effet il suffit d'une ligne de commande pour installer n'importe quelle extension.

Java EE étant par définition une suite de dépendances, il est conçu pour fonctionner seul sans avoir besoin d'installer autre chose. Cette philosophie peut faire gagner du temps car cela évite de devoir chercher parmi la multitude de frameworks similaires en javascript, mais peut aussi se montrer limitant pour certains domaines qui ne sont pas supportés nativement par Java EE. L'installation de dépendance externe est possible mais doit se faire manuellement. Et le nombre de bibliothèques existantes est très réduit par rapport à Node.JS. De plus, il est plus difficile avec Java EE de trouver des guides explicatifs ou de trouver de l'aide en ligne.

### 3.3.3.2. Protocole réseau

La partie protocole réseau se distingue en deux parties: la partie HTTP, et la partie WEBSOCKET.

#### 3.3.3.2.1. HTTP

Le protocole http sert à la connexion entre le navigateur et le serveur. C'est ce protocole qui est utilisé, notamment, pour que le navigateur reçoive la page web. C'est ce protocole qui va demander le script javascript qui sera exécuté sur le navigateur du client.

Pour la gestion du protocole HTTP, les deux technologies ont un fonctionnement assez similaire, avec la notion de méthodes appelées lorsqu'une action se produit, par exemple la réception d'une requête GET. Les deux technologies, bien qu'utilisant des noms de méthodes différentes, se valent pour cette partie.

#### 3.3.3.2.2. WebSocket

Le protocole websocket est un protocole réseau qui permet de communiquer en temps réel entre l'application javascript qui tourne dans le navigateur du client. La principale différence entre ce protocole et le HTTP est la création d'un flux de données qui reste ouvert entre le client et le serveur. Nous utilisons ce protocole pour permettre au serveur d'envoyer les paquets seulement au bon moment et d'éviter que le client doit demander toutes les informations avec http.

Pour ce protocole, la partie client étant développée en js, il est plus simple d'utiliser le javascript côté serveur afin d'utiliser les mêmes fonctions et de simplifier le développement.

#### 3.3.3.3. Interface

Pour gérer une interface homme machine les deux technologies sont assez similaires, elles utilisent toutes les deux une technologie web classique en passant par le navigateur web et en utilisant les langages html et css. Sur cette partie il n'y a donc pas vraiment de choix évident à faire, les deux technologies se comportent plus ou moins de la même manière.

Le seul point qui pourrait faire pencher la balance serait la connaissance du langage par les membres de l'équipe car nous connaissons le Java et pas le JavaScript mais là encore même si Node.JS ne demande que de connaître le JavaScript, Java EE demande un temps d'apprentissage important même si on est déjà familier avec le Java.

#### 3.3.3.4. Conclusion

Pour conclure, ces technologies sont semblables sur de nombreux points et le choix se joue finalement sur des critères assez subjectifs. Même si nous avons étudié le Java et l'orienté objet à l'iut, le Java EE est très différent de ce que nous avons vu et nous demanderait beaucoup de travail d'apprentissage sur le sujet, quasiment autant que le javascript. De plus, nous allons utiliser le javascript dans tous les cas dans l'application car c'est le langage qui est exécuté côté client. Pour cette raison nous avons décidé d'utiliser Node.JS car cela nous permet d'utiliser le même langage partout dans l'application et donc de garder la même logique de développement et accélérer l'apprentissage. Aussi, l'utilisation du même langage nous permet d'utiliser les mêmes dépendances côté client et côté serveur et nous permet de mieux les maîtriser.

### 3.4. Gestion du travail

La réalisation de ce projet demandant une quantité conséquente de travail, la mise en place de méthodes et d'outils est indispensable à l'avancement du projet. Voici quelques solutions mises en place pour organiser le développement du projet.

#### 3.4.1. Discord

Discord est un logiciel de chat vocal et écrit très populaire avec notamment la possibilité de créer plusieurs salons. La grande majorité de nos communications passe par ce logiciel avec différents salons textuels qui nous permette d'avoir accès directement aux informations d'une catégorie en particulier.

#### 3.4.2. Trello

Le modèle scrum n'est pas adapté pour notre projet, car nous avons très peu de créneau de travail en commun. Nous nous répartissons les tâches avec un délai et chacun s'organise comme il peut pour faire le travail. Cependant un outil d'organisation et de répartition du travail est nécessaire. Pour ce projet, nous utilisons l'outil en ligne Trello, car c'est un outil que nous avons utilisé notamment pour les projets des semestres précédant et avec lequel nous sommes familier.

Nous avons découpé notre trello en trois catégories : A faire, En cours, Fait. Cet outil nous permet de découper le travail en petites tâches à effectuer, puis d'y ajouter un membre et une date butoire (ainsi que pleins d'autres fonctionnalités que nous n'utilisons pas pour l'instant). Cette méthodologie permet à tout le monde de voir ce sur quoi les autres sont en train de travailler. Nous gardons les tâches effectuées car cela nous permet de voir a posteriori qui a effectué la tâche et d'aller voir cette personne en particulier en cas de problème ou de questions. De plus, ce logiciel est très visuel et permet de repérer très facilement, notamment avec un code couleur, quelle tâche nous concernent et à quelle catégorie elles appartiennent.

### 3.4.3. Tests

Pour un projet de cette envergure, le développement de tests en même temps que le code est indispensable. Le modèle TDD (test driven development), qui est basé sur le fait de développer les tests avant le code, n'est pas imposé dans le groupe car nous n'avons eu aucune formation a ce sujet, mais il est encouragé. Cependant, le développement de tests est obligatoire, ces tests permettent de vérifier le bon fonctionnement du projet, même après modifications. Ils aident également à identifier la source de bug, et peuvent même servir à comprendre le fonctionnement de code écrit par un autre membre du groupe, en voyant celui-ci en application.

### 3.4.4. Répartition des tâches

#### 3.4.5. Première itération

Pour la première partie, nous avons discuté tous ensemble du projet et avons réfléchi communément aux différents enjeux et contraintes du projet. Puis nous nous sommes réparties les tâches pour la rédaction du compte rendu.

#### 3.4.6. Deuxième itération

Pour la deuxième itération, nous avons découpé le projet par fonctionnalités:

- Nathan était chargé de l'accueil
- Carlos était chargé de la partie question des lecteurs
- Abel était chargé de la partie Quizz Présentateur
- Thomas était chargé de la partie affichage et manipulation du PDF
- Clément était chargé de tout ce qui est communications clients-serveur
- Adrien était chargé de la partie Droit

Chacun était responsable de modéliser sa partie, de commencer la réalisation d'un prototype et de la rédaction dans le compte rendu.

## 3.5. Description d'un modèle de développement

Comme expliqué précédemment, l'utilisation d'une méthode Scrum n'est pas adaptée à notre projet car il n'est pas possible ni justifié d'organiser des réunions quotidiennes. Cependant nous allons essayer d'utiliser un maximum d'éléments des

méthodes agiles. Notamment en fonctionnant par cycle et en travaillant sur un modèle itératif. Nous aurons tout de même des réunions fréquentes (Plusieurs fois par semaine) pour que tout le monde puisse communiquer son état d'avancement.

Le projet sera découpé en plusieurs parties (lecteur pdf, serveur, ihm...) qui travaillent ensemble.

Le découpage par partie permet d'avoir des "spécialistes" d'un domaine qui connaissent très bien le code et la technologie sur laquelle il travaille.

## 4. Conception

### 4.1. Diagramme de classe

L'ensemble du projet est réalisé sous forme d'un diagramme de classe disponible dans l'[Annexe 16](#).

### 4.2. Diagramme de cas d'utilisation

Dans notre application web, il existe la possibilité d'avoir deux "rôles" pour le client: Il peut être Présentateur ou Lecteur. Le présentateur, comme le montre le diagramme de cas d'utilisation, aura accès à fonctionnalités différentes du Lecteur. Le présentateur pourra créer un salon, et la création d'un salon inclut aussi la charge du PDF que la personne va présenter. Au contraire du lecteur, il pourra aussi lancer un quiz et le lancement de ce quiz va inclure l'envoi du quiz à tous les lecteurs et seulement les lecteurs. De plus, le présentateur pourra annoter le PDF en cours de présentation, ainsi cela va inclure une mise à jour du PDF pour le reste des personnes dans le salon par la part du serveur. Finalement, le présentateur comme le Lecteur doit choisir un Pseudo lors de la création du salon.

Le Lecteur, lui aussi, aura accès à des fonctionnalités différentes que celles du Présentateur. Le Lecteur aura la possibilité de poser une question au présentateur grâce au bouton poser une question. Lorsque le lecteur pose une question cela va entraîner le serveur a renvoyé la question posée à tous les autres lecteurs dans le salon. Finalement, le lecteur aura aussi la possibilité de rejoindre un salon en choisissant un pseudo, de même le serveur va se charger de charger le PDF pour le client.

c.f [Annexe 6](#)

### 4.3. Phase de connexion

#### 4.3.1. Présentation

La phase de connexion du site commence à la requête de la page d'accueil par l'utilisateur lors de son arrivée sur le site et se termine à la redirection sur la page du suivi de cours, tous les échanges de cette partie seront fait avec le protocole HTTP.

#### 4.3.2. Échanges serveur-client

Dans cette phase de connexion 3 cheminements différents sont reconnus et présentés dans les schémas en annexe 8, 9 et 10.

Pour les cas "créer" et "rejoindre" un salon le début est similaire, l'utilisateur sera redirigé sur la page d'accueil par le serveur en arrivant sur le site. Sur cette page d'accueil, l'utilisateur renseignera son pseudonyme et validera son choix à savoir "créer un salon" ou "rejoindre un salon", ensuite le serveur redirigera l'utilisateur vers la page correspondant à son choix.

Dans le cas "rejoindre un salon" l'utilisateur devra renseigner le fichier pdf qu'il souhaite afficher sur son salon et il sera envoyé au serveur qui redirigera l'utilisateur vers la page présentateur et lui transmettra le code de son salon. c.f [Annexe 8](#)

Dans le cas “créer un salon” l'utilisateur devra renseigner le code du salon qu'il souhaite rejoindre et il sera envoyé au serveur qui lui enverra la page lecteur du salon correspondant au code. c.f [Annexe 9](#)

Dans le troisième cas, à savoir celui où le présentateur a transmis au préalable une URL à l'utilisateur, le serveur va récupérer le code du salon dans l'url et envoyer à l'utilisateur une page où l'utilisateur aura uniquement à renseigner son pseudonyme qui sera envoyé au serveur. Pour finir le serveur va simplement rediriger l'utilisateur vers la page lecteur et lui enverra le fichier pdf du salon. C.F [Annexe 10](#)

En [Annexe 11](#) on peut trouver un schéma des routes utilisées par le serveur pour diriger l'utilisateur et récupérer les informations nécessaires. Ensuite en [Annexe 12](#) se trouve un diagramme de séquence précisant en plus des routes les données envoyées à l'utilisateur.

### 4.3.3. Mise en place

Pour coder cette partie nous utilisons plusieurs frameworks principalement

D'un point de vue pratique pour effectuer le cheminement entre les différentes pages, le serveur va exécuter différentes procédures qui vont soit récupérer des variables soit envoyer des pages. On peut voir sur l'[annexe 11](#) un schéma représentant les différentes procédures exécutées par le serveur et les variables qu'il récupère. Enfin en [annexe 12](#) nous pouvons voir un diagramme de séquence représentant les échanges entre le serveur et le client et ainsi que le type d'échange (fichier html, variable).

## 4.4. Lecteur PDF

Le code du lecteur PDF est séparé en trois parties: une partie dédiée au salon présentateur, une partie dédiée au salon lecteur et une partie commune aux deux.

La partie commune permet de gérer les fonctionnalités principales telles que l'affichage d'une page ou le changement de page. Ce script commence par mettre en place certaines informations qui seront nécessaires par la suite. Le script décrit une fonction qui permet de dessiner une page sur le canvas. La fonction suivante est celle qui sera appelée par les suivantes afin de garantir qu'il n'y ait pas de problèmes de rendu grâce à un système de file d'attente. Il y a ensuite des fonctions dédiées au changement de page qui appelleront la fonction précédente.

Le script lecteur s'occupe de récupérer les fonctions communes, se connecte au salon et ajoute les fonctionnalités "rejoindre" et "suivre". Le script s'occupe à chaque mise à jour du socket de changer la page du lecteur. Les fonctions sont ensuite ajoutées aux boutons.

Le script présentateur commence par importer les fonctions globales puis se connecte au socket. Une fonction est ensuite déclarée pour ajouter aux différents boutons la synchronisation aux différentes vues lecteur. Les boutons sont ensuite associés aux différentes fonctions.

Du côté des annotations, les différentes parties seront elles aussi séparées en 3 scripts. La partie commune s'occupera de l'affichage des données enregistrées sur chaque slide. La partie présentateur se chargera de détecter les différentes annotations qui seront enregistrées et envoyées à travers le socket. Les données seront ainsi récupérées par le script lecteur à travers le socket et affichées par le script commun. Le script lecteur s'occupera aussi au démarrage de récupérer l'ensemble des annotations précédentes.

## 4.5. Bouton Lever la Main

Mettons en situation qu'un salon de cours est ouvert avec plusieurs lecteurs. Si pendant le cours, un des lecteurs souhaite poser une question, le bouton lever la main va être la solution. Comme on peut le voir dans le diagramme de séquence (cf [Annexe 7](#)), le lecteur doit remplir le formulaire. Dans ce formulaire on a 2 champs à remplir.

Le premier, est le champ où le lecteur devra mettre le sujet de la question à poser. De cette façon les clients auront après les questions affichées, chacune avec un sujet clair.

Le deuxième champ à remplir c'est la question. Pour ce champ on a une zone de texte où le lecteur pourra écrire une courte question à envoyer au présentateur.

Suite au remplissage du formulaire, le lecteur doit cliquer sur le bouton envoyer pour envoyer la question au présentateur.

L'information du formulaire est récupérée et elle est envoyée au serveur. Ensuite le serveur, à l'aide de websocket, va envoyer la question posée à tous les clients dans le salon. Cette question va s'afficher avec les autres questions déjà posées et on aura une page similaire à la vue fournie ([Annexe 2](#) pour le lecteur, et [Annexe 3](#) pour le présentateur).

Le lecteur se trouvera avec le petit formulaire pour envoyer la question où il devra compléter de manière impérative le sujet de la question. Si, le lecteur essaye d'envoyer une question avec un sujet vide, une exception sera levée indiquant au lecteur de remplir le sujet avant d'envoyer. De plus, si le sujet est complet, mais le champ de la question est vide, une exception va se lever indiquant à la personne de remplir le champ de la question. De plus, si rien est rempli, et que le lecteur veut envoyer la question vide, une autre exception va se lever indiquant au lecteur de compléter les différents champs.

Finalement, le bouton envoyer, va faire que le contrôleur va récupérer le sujet et le texte du champ question pour ensuite pouvoir l'envoyer à tous les clients.

## 4.6. Fonctionnalité Quizz

Mettons en situation que l'enseignant prépare son prochain cours en distanciel, celui-ci souhaite préparer également les questions du quizz qu'il pourra lancer à n'importe quel moment durant la présentation. Ainsi, l'enseignant peut créer des questions avant (même pendant) son cours en allant sur la page dédiée au quizz de l'enseignant. Une fois la page chargée, l'enseignant se retrouve devant une page similaire à la vue fournie ([Annexe 5](#)). Celui-ci aura à sa gauche un récapitulatif des questions qui ont déjà été préparées au préalable par l'enseignant (s'il y'en a), récupérées à partir du serveur. Tandis qu'à sa droite, se trouvera le champ de création de question qui se distingue par défaut en 3 champs de texte - dont 2 accompagnés de boutons radios - et 2 boutons.

Le premier champ de texte correspond à l'intitulé de la question (par exemple : "Lequel de ces items est faux ?" ou "Selon le Code de la Propriété Intellectuelle, le droit au titre est-il un droit moral ou patrimonial ?").

Les deux autres champs de textes correspondent aux items du QCM que les étudiants sélectionneront en songeant qu'il s'agit de la réponse correcte à la question.

Les boutons radios suivant les champs de texte permettent à l'enseignant de distinguer la bonne réponse parmi tous les items.



Enfin, les deux boutons en bas de la section correspondent respectivement à la fonction d'ajouter un item de réponse (la question peut aller jusqu'à 4 items) ce qui rajoutera un champ de texte en dessous du dernier item pour en créer un nouveau; et à la fonction de créer la question ce qui mettra à jour la liste de questions stockée dans le serveur avec la nouvelle question créée.

La liste des questions à gauche sera donc rafraîchie et fera apparaître la nouvelle question accompagnée des boutons représentés par une flèche, une loupe et une croix.

La flèche permet de lancer la question côté étudiant, ce qui affichera une pop-up avec l'intitulé de la question et les items à sélectionner accompagnés d'un bouton radio puis un bouton "valider" pour envoyer la réponse. Ainsi, le serveur incrémentera le compteur de l'item choisi et le nombre de réponses total (passé en variable globale).

La loupe permet à l'enseignant de voir les résultats reçus des étudiants, affichant une pop-up avec un pourcentage des réponses rentrées jusqu'à l'instant du clic sur le bouton. L'affichage n'est pour l'instant pas mis à jour régulièrement mais reste un détail sur lequel nous pourrons nous pencher plus tard.

Enfin, le dernier bouton correspond au fait de supprimer la question : un bouton en forme de X. Lors de la pression sur ce bouton, le contrôleur supprime la question de la liste de questions du serveur. Il ne fait donc pas qu'un simple `pop_back`, mais cherche l'index sélectionné pour le supprimer et renuméroter d'un index en dessous toutes les questions suivant celle choisie pour être supprimée. Suite à la suppression de la question dans le contrôleur, la vue ré-affiche toutes les questions restantes accompagnées de leur nouvelles numérotations pour celles où il y'a eu modification.

Un diagramme de séquence est disponible en fin de document pour résumer la fonctionnalité Quizz (cf [Annexe 15](#))

## 4.7. Communication Client - Serveur

Une fois le salon lancé sur le client, la communication client serveur est effectuée via le protocole websocket. Ce protocole permet une communication en temps réel dans les deux sens très facilement. Cette communication bidirectionnelle est indispensable dans notre projet car le serveur peut envoyer des données à tout moment. Pour utiliser ce protocole nous utilisons la librairie très populaire `sockets.io` qui utilise un protocole basé sur les websockets tout en ajoutant des fonctionnalités.

### 4.7.1. Synchronisation page

Afin que les lecteurs soient toujours à jour par rapport à la page du présentateur, une succession d'échange de paquets est déclenché lorsque le présentateur change de page. Cet événement est modélisé par le diagramme de séquence en [annexe 13](#).

### 4.7.2. Connexion lecteur

Pour que le lecteur soit informé de l'avancée du présentateur dès qu'il rejoint le salon, un certain nombre d'informations lui est envoyé dès sa connexion. La connexion est modélisée par le diagramme de séquence en [annexe 14](#).

## 4.8. Droit vis-à-vis de code

Afin de coder efficacement et lisiblement, il est primordial d'utiliser des fonctions. Pour plus de facilité, certains bout de code sont stockés dans des librairies publiques. Cependant, ils sont parfois protégés par des licences, cela veut dire que l'on ne peut pas forcément l'utiliser sans autorisation du propriétaire.

Pour le projet, seulement deux licences sont utilisées : principalement du code de la licence MIT mais aussi de la licence Apache-2.0. Elles sont toutes les deux des licences de logiciel libre et open source. Utiliser, modifier et publier est donc autorisé.

Si nous utilisons du code de quelqu'un d'autre, nous encourons jusqu'à cinq ans d'emprisonnement ainsi que 150 000 euros d'amende, voire 300 000 en cas de récidive légale.

Suite à la création de notre site, si nous décidons de le publier, les droits d'auteur reviendront à l'IUT car la demande de ce site est en partie de leur part.

## 5. Réalisation

La réalisation du projet n'en est qu'à l'état de prototype. Plusieurs parties marchent séparément mais le logiciel n'est pas fonctionnel. Vous pouvez déjà tester le coeur de l'application, c'est à dire la synchronisation de pdf, à l'adresse

<http://yamazouki.freeboxos.fr/présentateur> pour avoir accès à un test du côté présentateur,

et à <http://yamazouki.freeboxos.fr/lecteur> pour le côté lecteur. La phase de connexion du

serveur marche aussi séparément vous pouvez avoir accès à l'accueil ici

<http://yamazouki.freeboxos.fr/> ainsi qu'au prototype de l'invitation par lien ici

<http://yamazouki.freeboxos.fr/room/2587> (ici code du salon = 2587, changez le code à la fin pour changer le code du salon).

# Annexe

## Annexe 1

Pseudonyme :

Admin

Créer un nouveau salon

Rejoindre un salon existant

The image shows a user interface for a chat application. It features a large rectangular frame containing a login screen. At the top, there is a label 'Pseudonyme :'. Below it is a text input field containing the word 'Admin'. Underneath the input field are two buttons: 'Créer un nouveau salon' on the left and 'Rejoindre un salon existant' on the right.

## Annexe 2

<p><b>Administrateur :</b></p> <p><input type="radio"/> Admin</p> <p><b>Invités :</b></p> <p><input type="radio"/> Invité1</p> <p><input type="radio"/> Invité2</p> <p><input type="radio"/> Invité3</p> <p><input type="radio"/> Invité4</p> <p><input type="radio"/> Invité5</p> <p><input type="radio"/> Invité6</p> <p><input type="radio"/> Invité7</p> <p><input type="radio"/> Invité8</p> <p><input type="radio"/> Invité9</p> <p><input type="radio"/> Invité10</p> <p><input type="radio"/> Invité11</p> <p><input type="radio"/> Invité12</p>	<p style="text-align: center;"><b>Titre</b></p> <p>Chapitre I</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p>*****</p> <p style="text-align: right;">1</p>	<div><div>Question Cours</div><div><div></div><div></div><div></div></div></div> <div><div>Sujet</div><div>Posez votre question...</div><div>Envoyer</div></div> <div><div>Rejoindre</div><div>Suivre <input type="checkbox"/></div></div>
--	---	--











## Annexe 3

The screenshot shows a presentation interface. On the left, there is a sidebar with the title 'Administrateur :'. Below it, there is a list of users, each preceded by a circular icon. The first user is 'Admin', and the others are 'Invité1' through 'Invité12'. The main area of the slide is titled 'Chapitre IV' and contains ten horizontal lines of asterisks. On the right side, there is a box labeled 'Invité1' with four horizontal lines for text entry. At the bottom right, there is a green button labeled 'Lancer un quizz' with a plus sign icon.

## Annexe 4

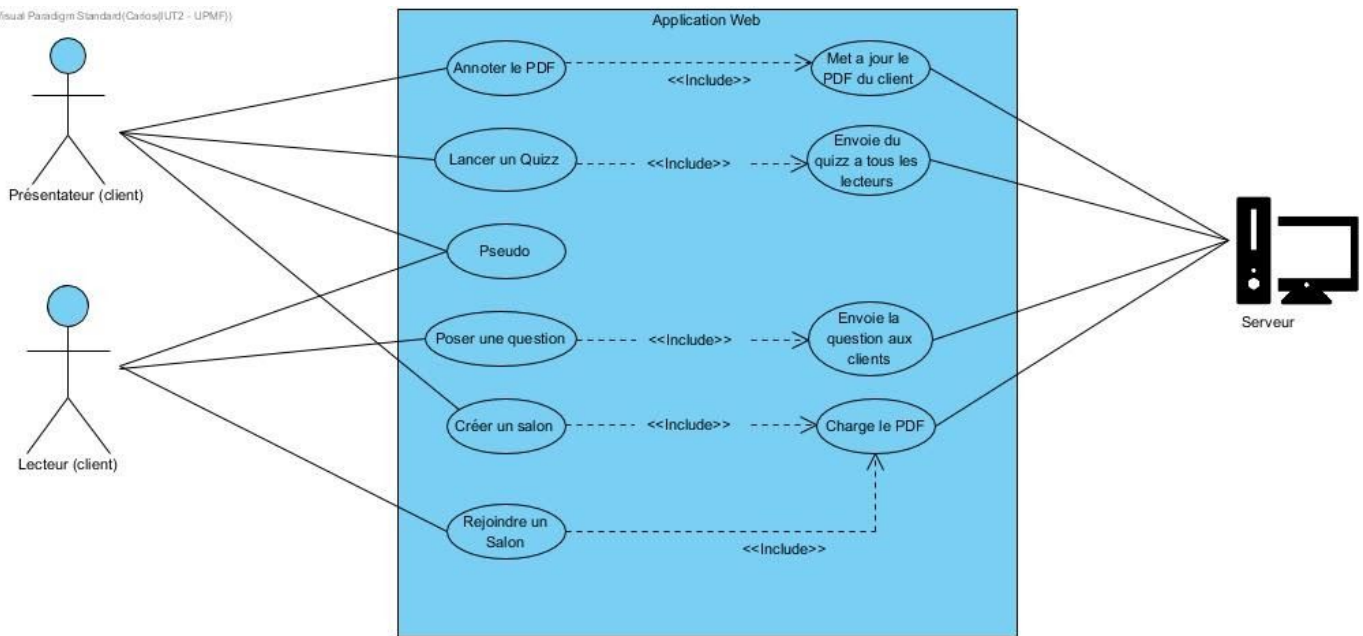
<p><b>Administrateur :</b></p> <p><input type="radio"/> Admin</p> <p><b>Invités :</b></p> <p><input type="radio"/> Invité1</p> <p><input type="radio"/> Invité2</p> <p><input type="radio"/> Invité3</p> <p><input type="radio"/> Invité4</p> <p><input type="radio"/> Invité5</p> <p><input type="radio"/> Invité6</p> <p><input type="radio"/> Invité7</p>	<div><h1>Titre</h1><h2>Chapitre 1</h2><p>*****</p><p>*****</p><p>*****</p><p>*****</p></div> <div><h3>QCM</h3><div><input type="text" value="A) Réponse A"/></div><div><input type="text" value="B) Réponse B"/></div><div><input type="text" value="C) Réponse C"/></div><div><input type="text" value="D) Réponse D"/></div><div><input type="button" value="VALIDE"/></div></div>	<div><div>Question Cours</div><div><p>.....</p><p>.....</p><p>.....</p></div></div> <div><div>Sujet</div><div>Poser votre question...</div><div><input type="button" value="Envoyer"/></div></div> <div><div>Rejoindre</div><div>Suivre <input type="checkbox"/></div></div>
--	--	--

## Annexe 5

<ul style="list-style-type: none"><li>● Question 1 : xxxxxxxxxxxxxxxxxxxx ?  </li><li>● Question 2 : xxxxxxxxxxxxxxxxxxxx ?  </li><li>● Question 3 : xxxxxxxxxxxxxxxxxxxx ?  </li><li>● Question 4 : xxxxxxxxxxxxxxxxxxxx ?  </li><li>...</li><li>● Question n : xxxxxxxxxxxxxxxxxxxx ?  </li></ul>	<p>Nouvelle question :</p> <p>Intitulé : .....</p> <p>A) ..... <input checked="" type="radio"/></p> <p>B) ..... <input type="radio"/></p> <p><a href="#">Ajouter proposition</a></p> <p><a href="#">Créer nouvelle question</a></p>
---	---

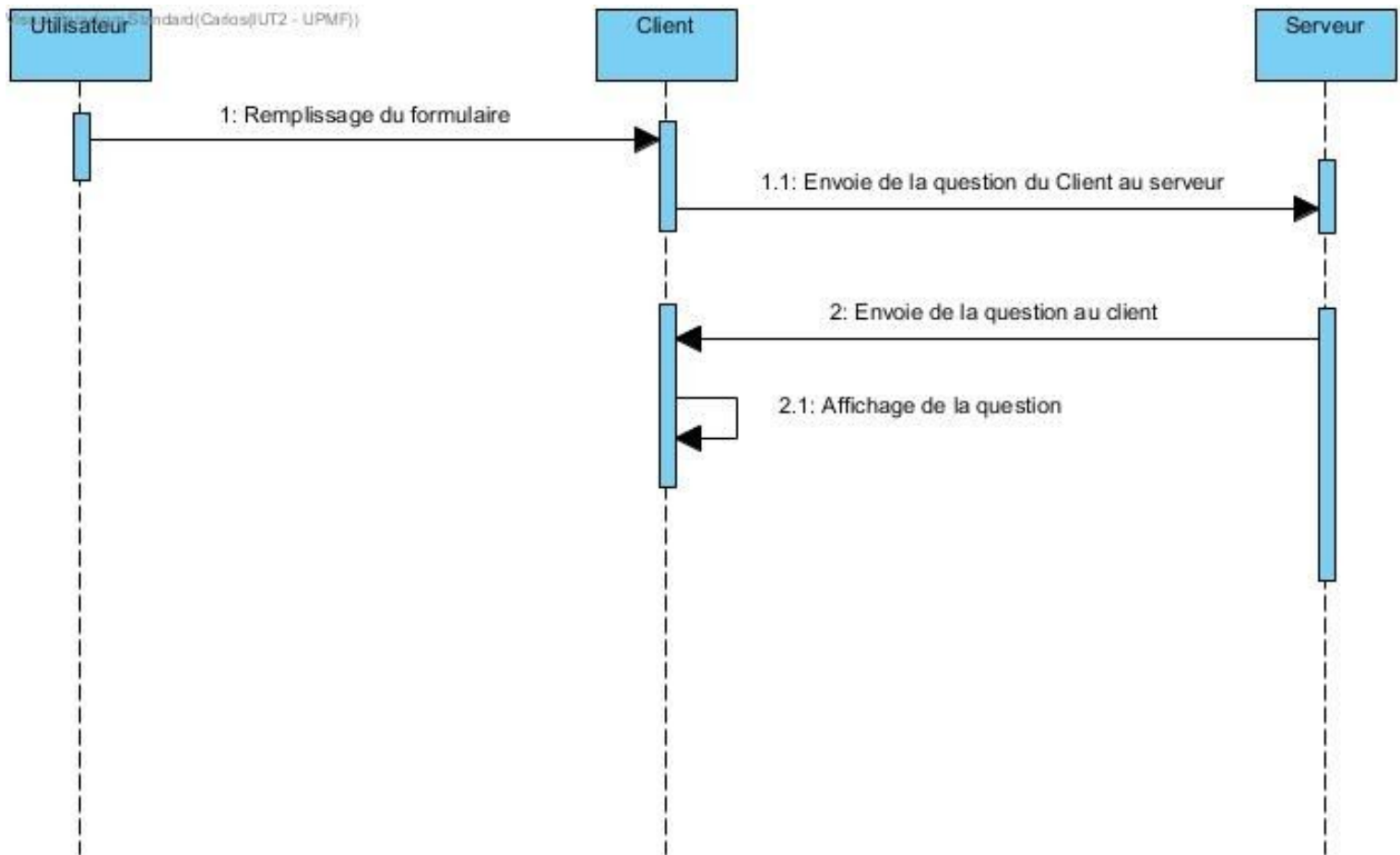
## Annexe 6

Visual Paradigm Standard(Carlos)(UT2 - UPMF)



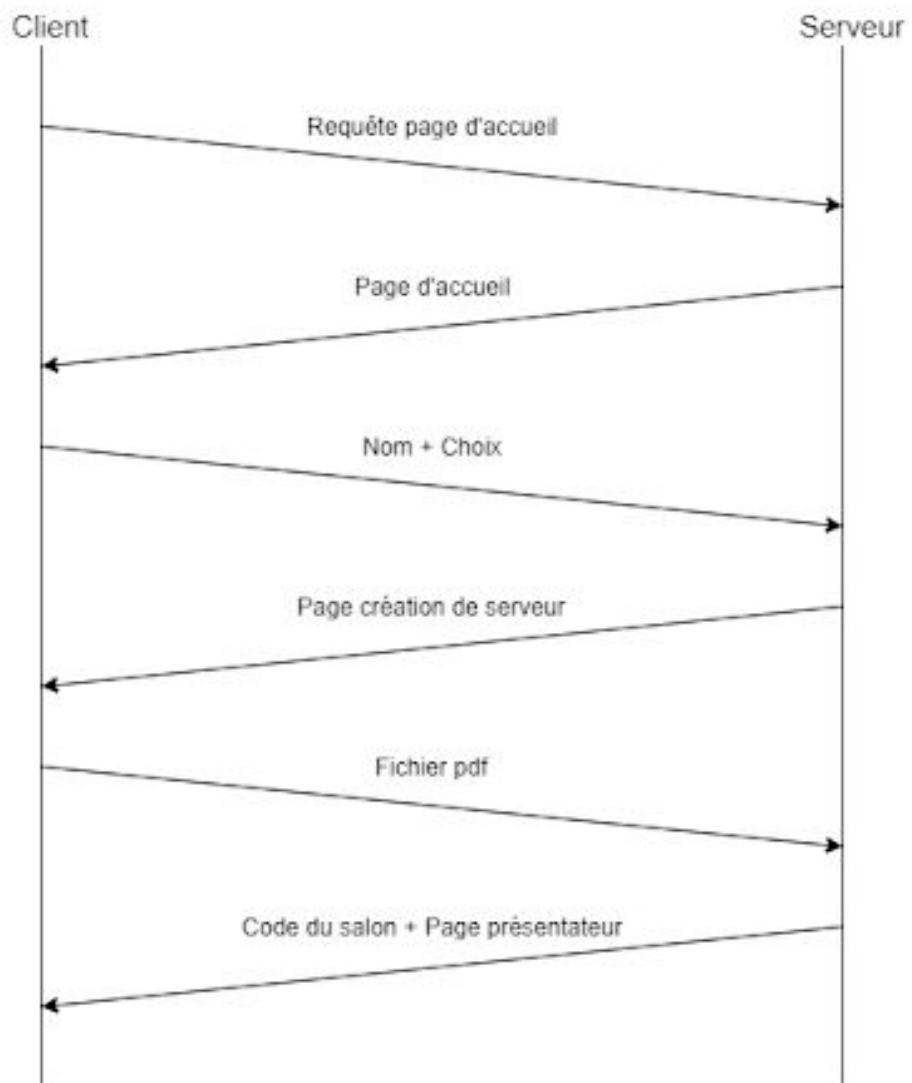


## Annexe 7



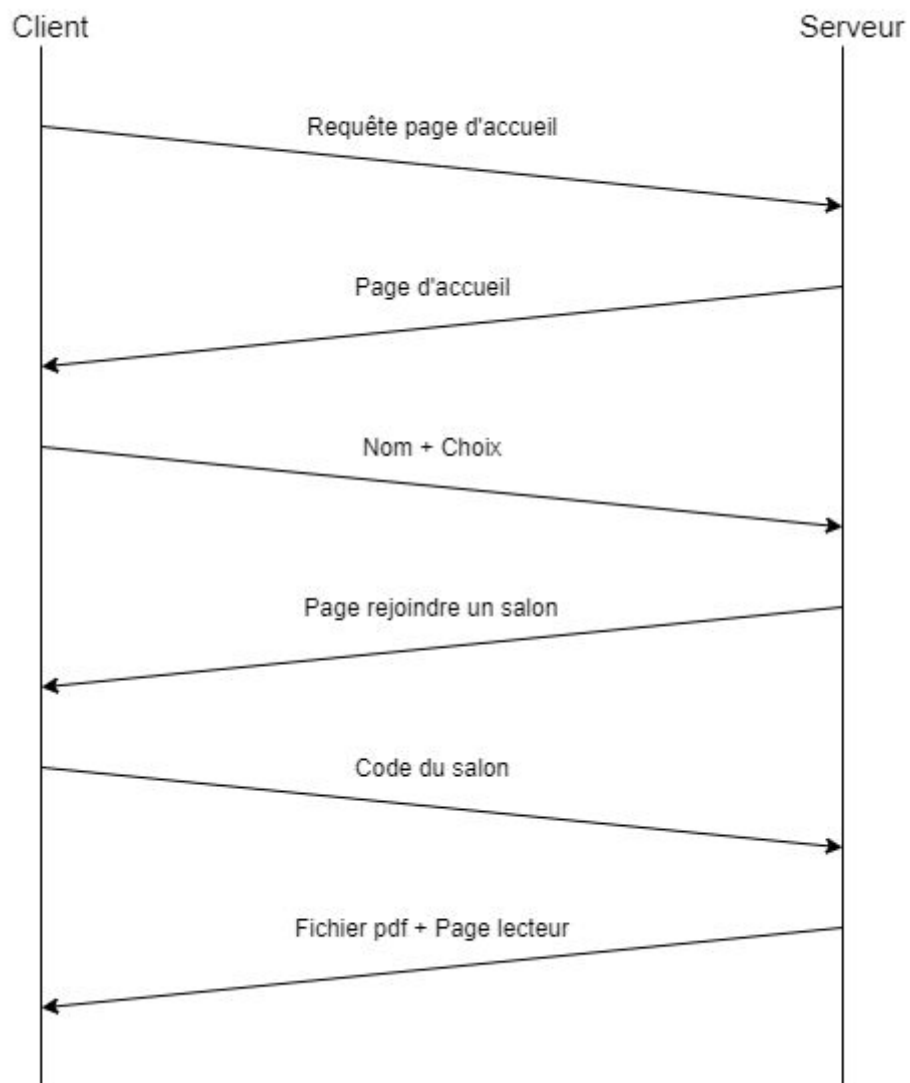
## Annexe 8

Choix = Créer un salon



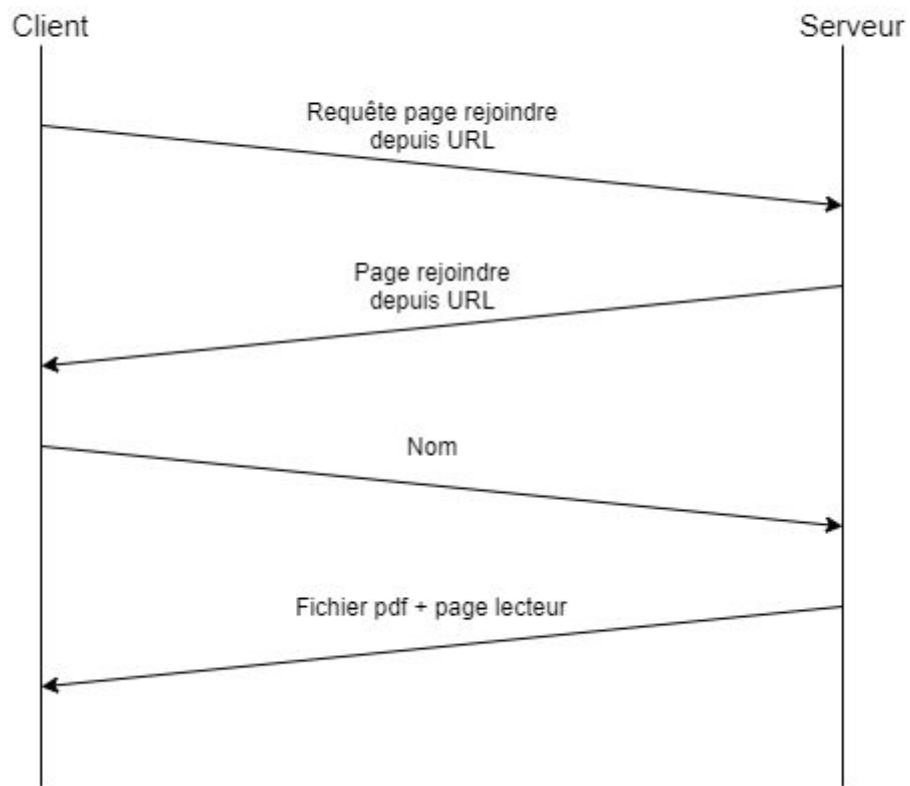
## Annexe 9

### Choix = Rejoindre un salon



## Annexe 10

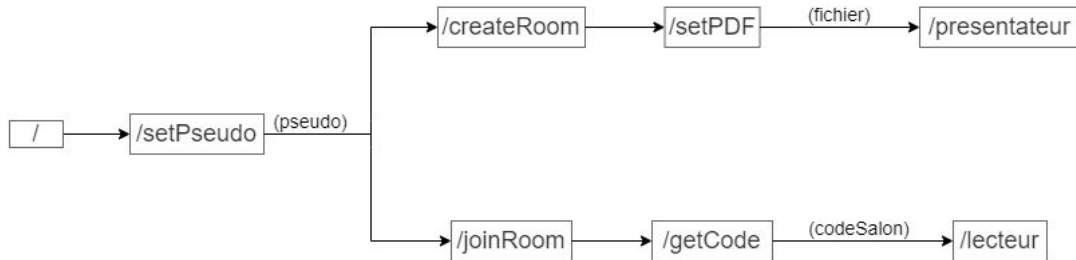
Choix = Rejoindre un salon  
depuis une URL



## Annexe 11

() : variables  
-> : redirections

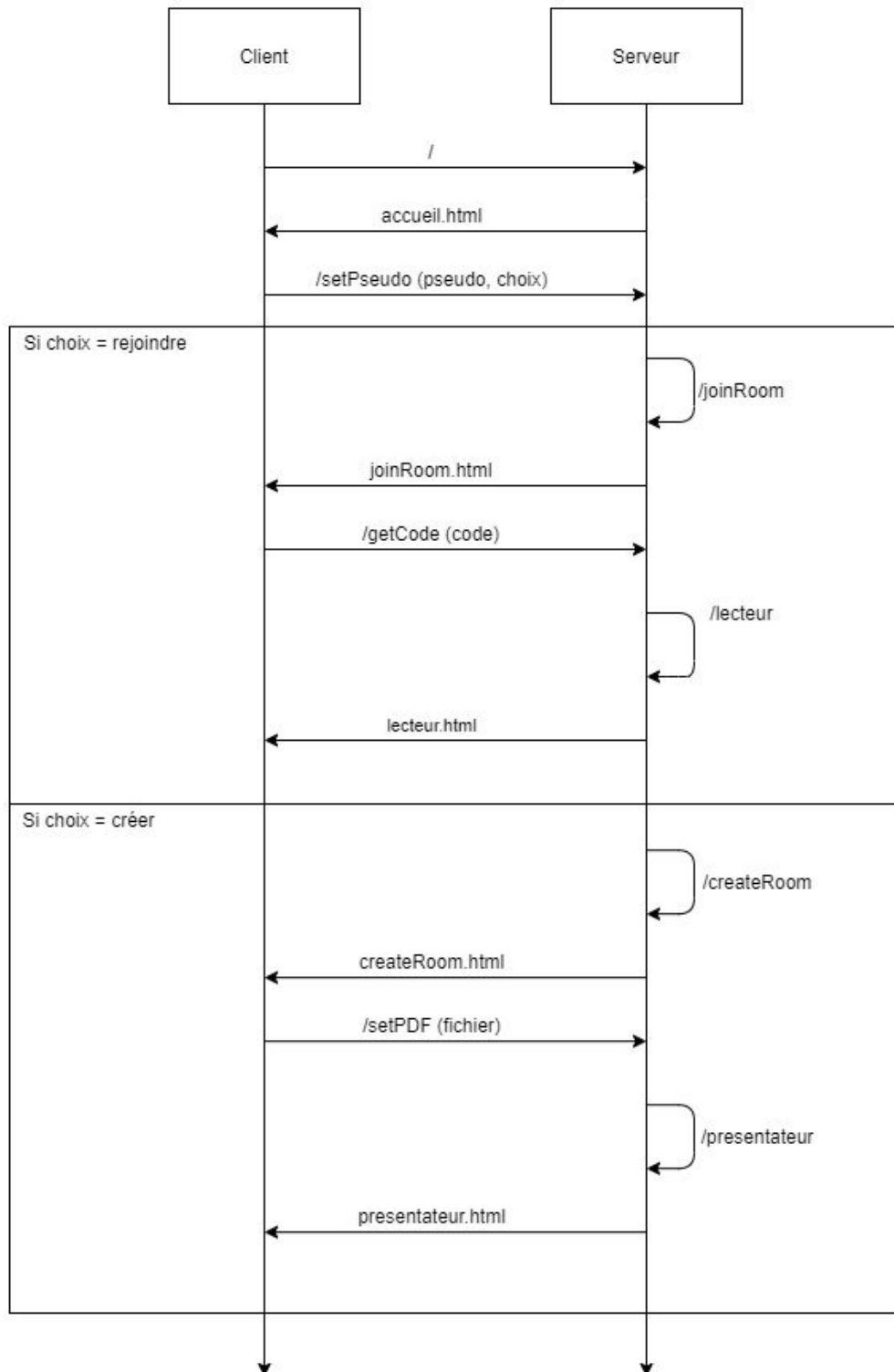
Arrivée sur la page d'accueil:



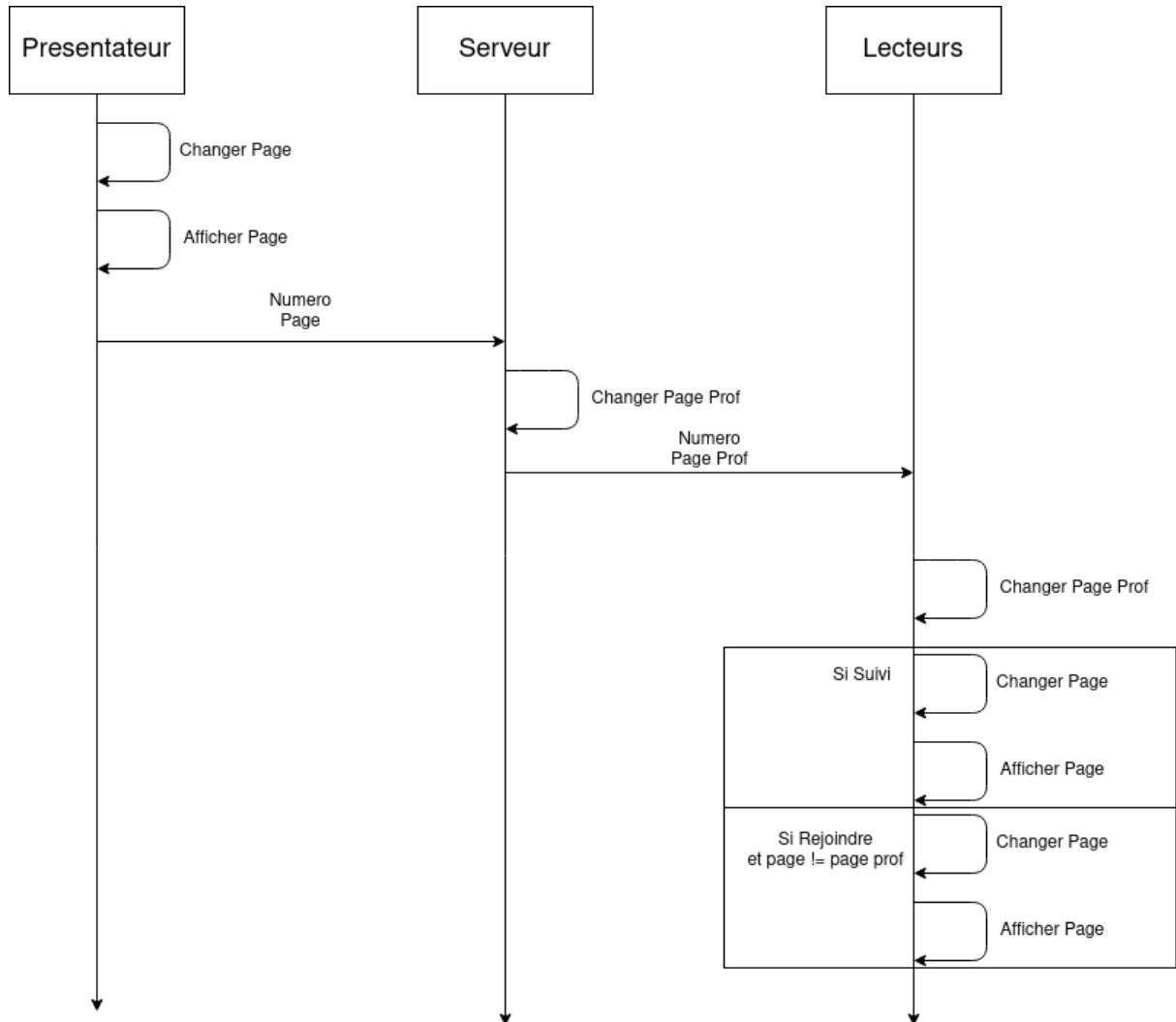
Code du salon dans l'URL:



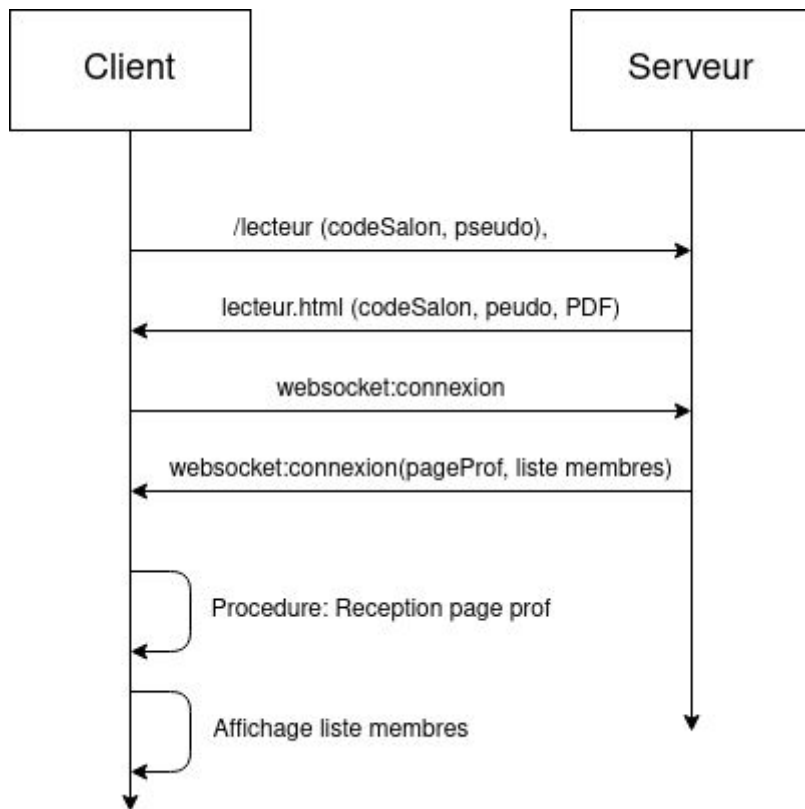
## Annexe 12



## Annexe 13

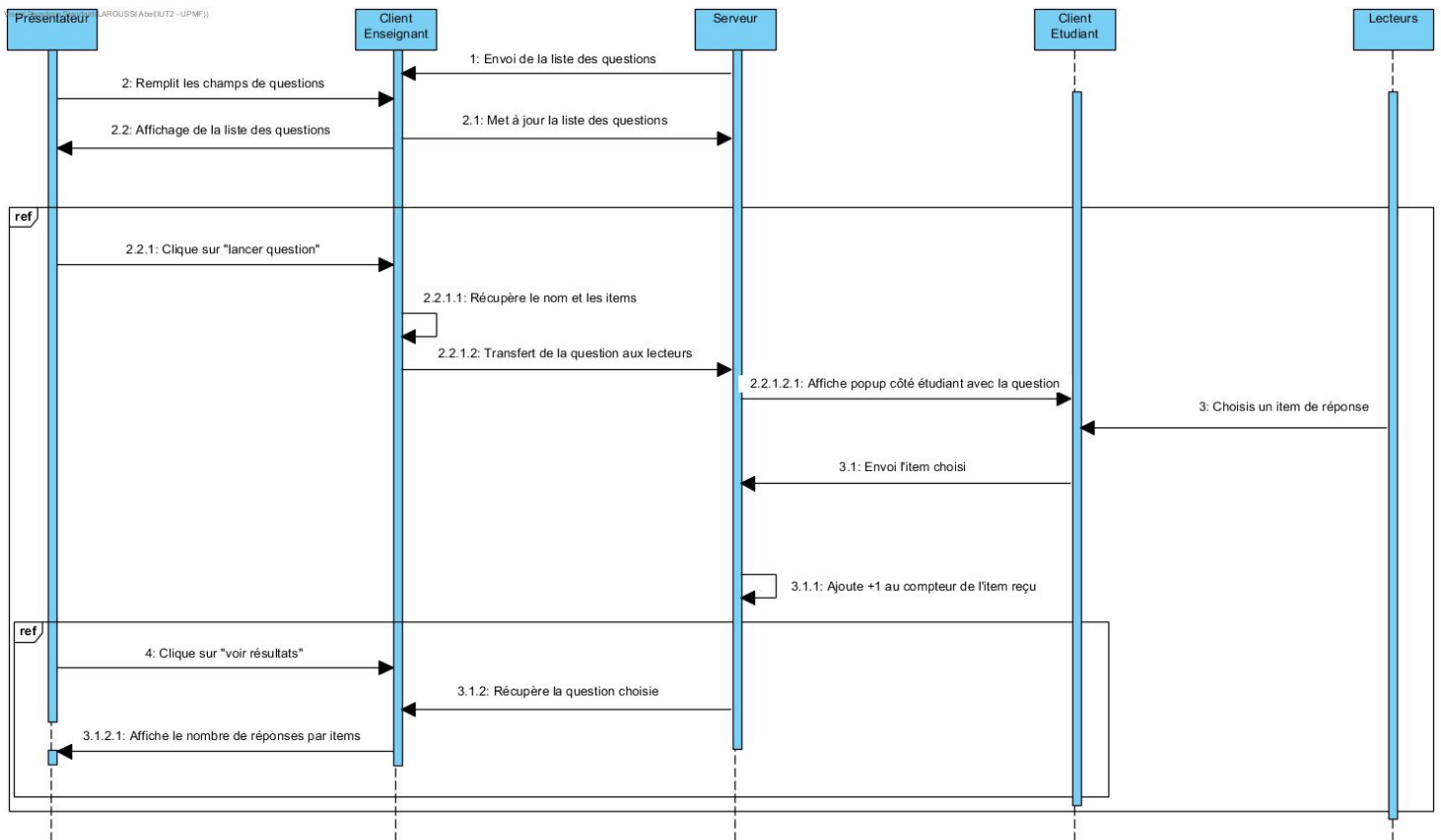


## Annexe 14





## Annexe 15



## Annexe 16

