# **FuzeScrum**

# **Besoins et conception**

## 1. Présentation du projet

*.*

FuzeScrum a une longue tradition d'encouragement du mentorat interne et a mis au point un programme interne qui permet aux personnes de l'entreprise de se porter volontaires pour être mentor ou mentoré. Les résultats positifs de ce programme permettent d’envisager la création d’un projet plus large, répondant à nos objectifs d’investissements dans l’innovation. Ceci nous permettrait d'appliquer notre savoir-faire dans le domaine pour mettre sur le marché un nouveau produit de mentorat. Nous sommes convaincus que ce produit recevra un bon accueil de la communauté et ajoutera de la valeur à notre marque.

**1.1 Objectifs du projet**

Le programme doit être capable :

* d’attribuer un utilisateur mentor à un utilisateur mentoré. Chacun ayant renseigné leur objectif personnel pour le mentorat, ainsi que quelques informations personnelles telle que nom, prénom etc.
* d’établir un contrat liant les 2 utilisateurs. Il comprend la durée de celui-ci, la fréquence des sessions de mentorat et un indicateur d’efficacité des dîtes sessions.
* De définir des sessions de mentorats entre les deux parties qui renseigne la date et l’heure de celui-ci, le sujet qui sera abordé et des notes.

*Les problèmes rencontrés :*

○ planification des sujets pour la prochaine session de mentorat ;

○ association du bon mentor au bon mentoré ;

○ enregistrement des retours d'expérience d'une session de mentorat ;

○ mesure de l'efficacité de la relation.

● Des solutions techniques préliminaires ciblant ces problèmes ;

● Les profils utilisateurs.

**1.2 Éléments hors périmètre**

Vidéoconférence : dans le cadre du projet, cette fonctionnalité ne sera pas inclus à cette version.

Messagerie : dans le cadre du projet, cette fonctionnalité ne sera pas inclus à cette version.

Cours : dans le cadre du projet, cette fonctionnalité ne sera pas inclus à cette version.

**1.3 Métriques du projet**

Un système de notation des sessions de mentorats par les mentorés.

Les mentorés attribuent une note entre 1 et 5.

1 étant la note la plus basse et 5 la plus haute, ils jugent l’efficacité de la session, si leur mentor a pu répondre à leur interrogation, si les explications fournies sont satisfaisantes.

De plus, on peut considérer que le projet est une réussite :

* Si les notes que les Mentorés attribuent aux sessions de mentorat atteignent une moyenne de 75% d’efficacité.
* Si au moins 80% des membres de l’entreprise ont un compte sur l’application.
* Si on note moins de 10% de perte de d’utilisateurs après un an.

2. Fonctionnalités

**2.1 Fonctionnalités du projet**

Le programme doit être capable :

* d’attribuer un utilisateur mentor à un utilisateur mentoré. Chacun ayant renseigné leur objectif personnel pour le mentorat, ainsi que quelques informations personnelles telle que nom, prénom etc.
* d’établir un contrat liant les 2 utilisateurs. Il comprend la durée de celui-ci, la fréquence des sessions de mentorat et un indicateur d’efficacité des dîtes sessions.
* De définir des sessions de mentorats entre les deux parties qui renseigne la date et l’heure de celui-ci, le sujet qui sera abordé et des notes.
* De permettre à un mentoré d’attribuer une note quant à l’efficacité de ses sessions de mentorats
  1. **User stories et critères d’acceptation**

En tant qu’Utilisateur, je veux que le programme m’autorise à m’**inscrire ou me connecter** afin que je puisse participer au programme de mentoring.

**Étant donné**que je suis un utilisateur système déconnecté et que je suis sur la page de connexion,

**Lorsque** je remplis les champs «Email» et «Password» avec mes informations d'authentification et que je clique sur le bouton Connexion,

**Alors**le système me connecte.

En tant que Administrateur, je veux que le programme m’autorise à **valider les inscriptions** d’autres Administrateur, Mentor et Mentoré afin de garder un contrôle sur les inscris. Il doit également me permettre d’**assigner un Mentor** à un Mentoré selon sa spécialité (Attribuer le bon Mentor au bon Mentoré). Il faut aussi que je puisse **gérer les contrats** Mentors et Mentorés.

**Étant donné**que je suis un Administrateur connecté et que je suis sur la page « Home »,

**Lorsque** je clique sur l’icône « Validé » près d’un nouvel inscrit,

**Alors**le système valide l’inscription de ce nouvel utilisateur.

**Étant donné**que je suis un Administrateur connecté et que je suis sur la page « Mentors » (ou « Mentorés »,

**Lorsque** je clique sur l’icône « Add Mentoré » (ou respectivement « Add Mentors ») après avoir sélectionné un Mentor (ou Mentoré),

**Alors**le système m’affiche la liste des Mentorés (ou Mentors), avec leur spécialités afin que les assignements soient pertinents.

**Étant donné**que je suis un Administrateur connecté et que je suis sur la page « Contrats »,

**Lorsque** je clique sur l’icône « Définir Contrat » ou « Voir Contrat » ou « Supprimer Contrat » près d’un Mentor ou d’un Mentoré,

**Alors**le système permet de définir les termes d’un contrat, de voir les détails d’un contrat existant ou de supprimer un contrat.

En tant que Mentor, je veux que le programme me m’autorise à **gérer une session de mentorat** avec un Mentoré (créer ou annuler **,** définir la date, l’heure et le sujet) afin que l’on puisse échanger, dialoguer dans le cadre de son apprentissage.

**Étant donné**que je suis un Mentor connecté et que je suis sur la page « Mentorés »,

**Lorsque** je clique sur l’icône « Plan Next Session » après avoir sélectionné un Mentoré,

**Alors**le système affiche un calendrier, sur lequel je dois définir date et heure de la session.

En tant que Mentoré, je veux que le programme m’autorise à **noter mes Sessions de mentorat** afin que les Mentors et les Administrateur aient un retour sur l’efficacité de celle-ci (mesurer si la relation fonctionne bien). Je dois également pouvoir **consulter les notes prises durant la session** afin d’en avoir un récapitulatif.

**Étant donné**que je suis un Mentoré connecté et que je suis sur la page « Session »,

**Lorsque** j’attribue un nombre d’étoile (1 à 5) à une session,

**Alors**le système valide cette note.

**Étant donné**que je suis un Mentoré connecté et que je suis sur la page « Session »,

**Lorsque** je clique sur l’icône « Résumé » correspondant à une session,

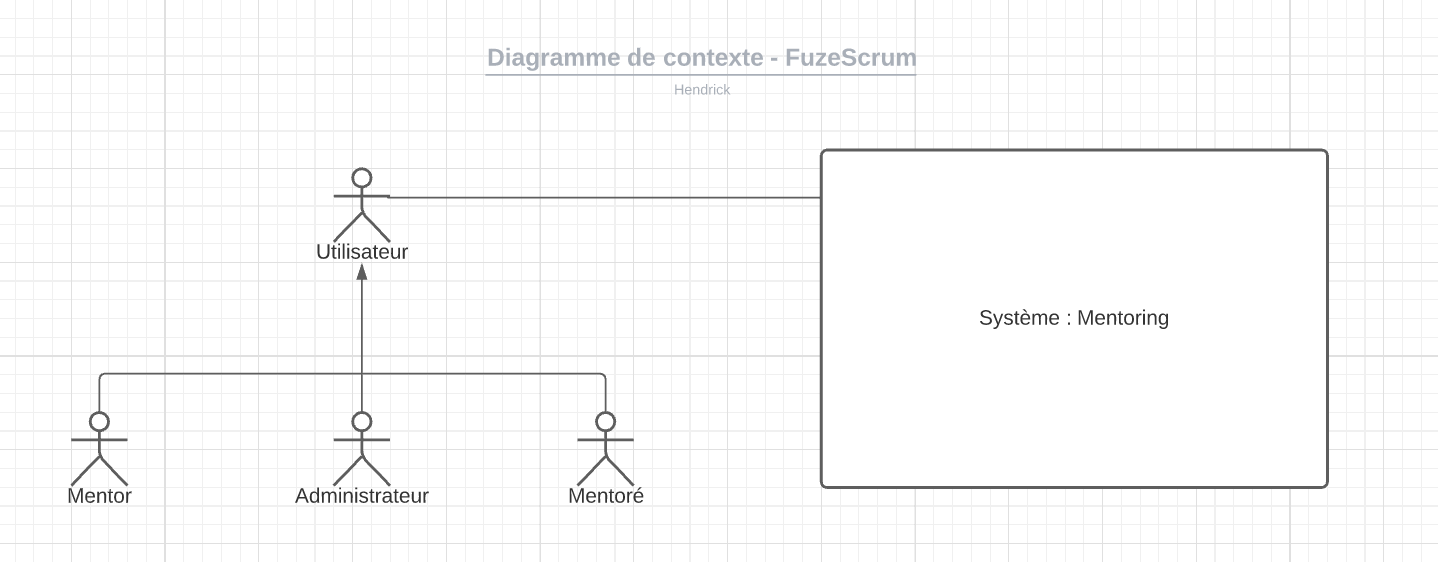
**Alors**le système m’affiche les notes prises durant cette session.

3. Solution proposée

**3.1 Diagrammes de conception technique**

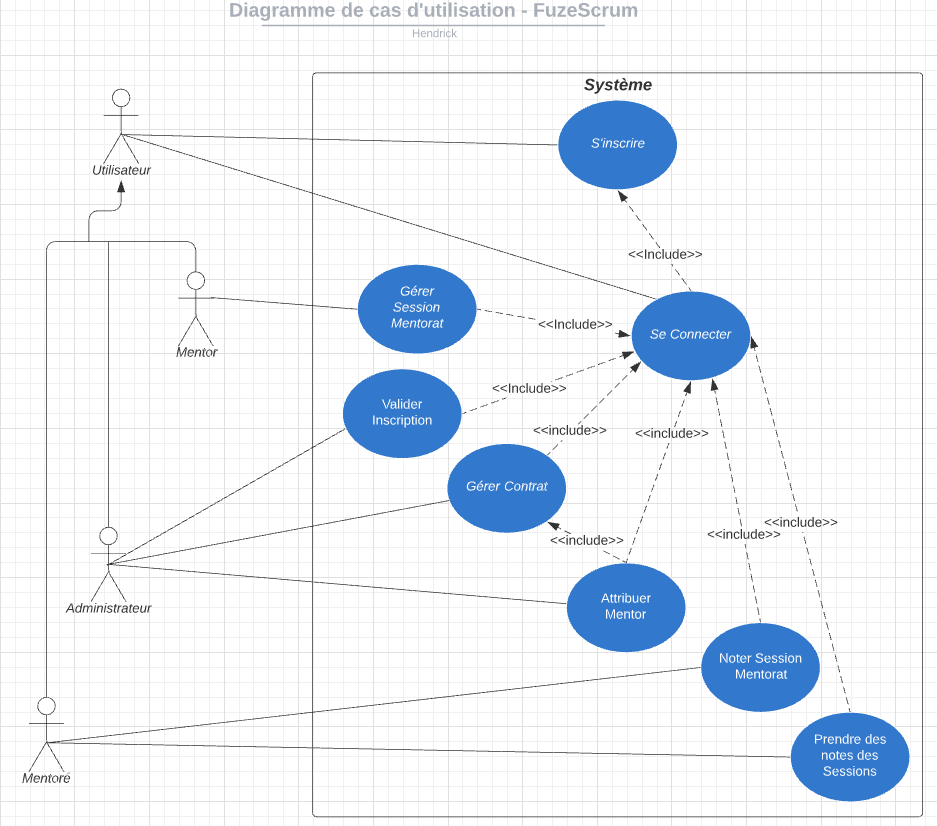
**3.1.1 Diagramme de contexte**

Le diagramme de contexte est l’un des premiers diagrammes pouvant être réalisé afin de décrire à haut niveau le futur système en mode boîte noire. Il permet de se concentrer sur l’identification des différents acteurs interagissant avec lui. Il est aussi possible de représenter les interactions entre acteurs si cela semble pertinent pour la compréhension globale du système. Lors de sa réalisation on ne parle ni d’implémentation, ni de solution, mais seulement de son contexte d’utilisation.

****

**3.1.2 Diagramme de cas d’utilisation**

Le diagramme Use-Case, appelé diagramme de cas d’utilisation en français, fait partie des diagrammes de comportement du langage Unified Modeling Language abrégé, avec les systèmes et processus de programmation objet ou encore les processus métier. UML n’est donc pas un langage de programmation, mais un langage de modélisation. C’est une méthode standardisée qui décrit un système en cours de conception, ou déjà existant.

****

**3.1.3 Diagramme de classe**

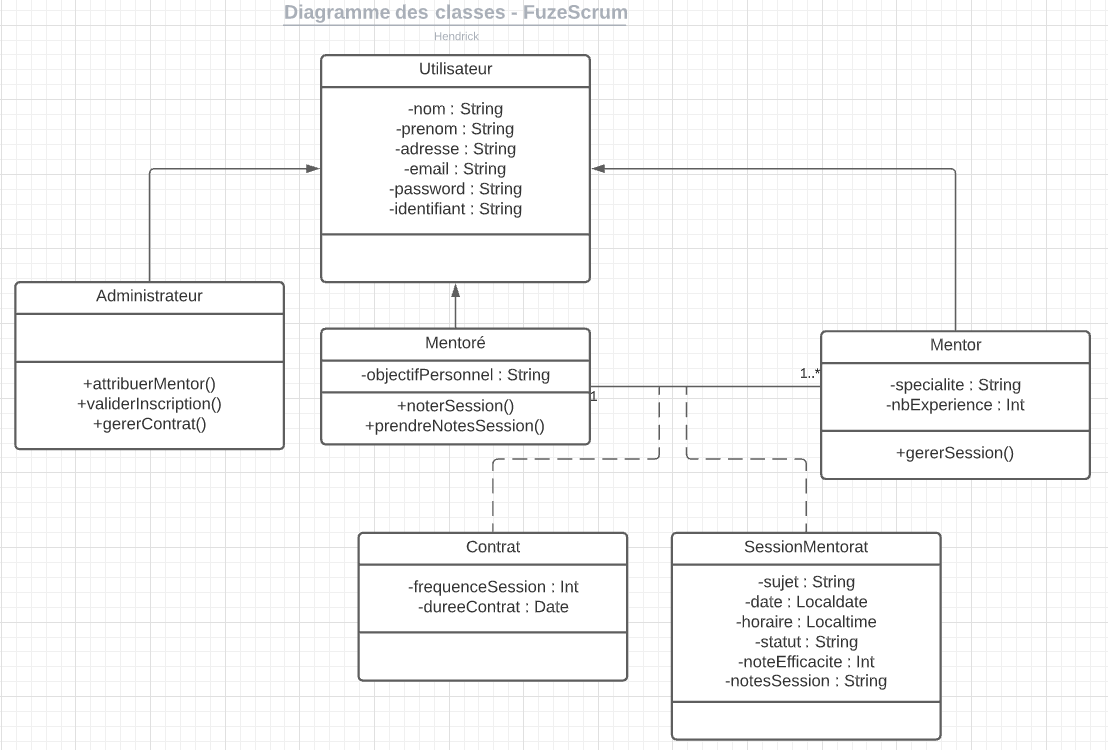
Le **diagramme de classes** est un schéma utilisé en génie logiciel présenter les classes et les interfaces des systèmes ainsi que leurs relations. Ce diagramme fait partie de la partie statique d’UML, ne s'intéressant pas aux aspects temporels et dynamiques.

Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe.

Une classe est un ensemble de fonctions et de données (attributs) qui sont liées ensemble par un champ sémantique. Les classes sont utilisées dans la programmation orientée objet. Elles permettent de modéliser un programme et ainsi de découper une tâche complexe en plusieurs petits travaux simples.

Les classes peuvent être reliées grâce au mécanisme d'héritage qui permet de mettre en évidence des relations de parenté. D'autres relations sont possibles entre des classes, représentées par un arc spécifique dans le diagramme de classes.

Elles sont finalement instanciées pour créer des objets (une classe est un *moule à objet* : elle décrit les caractéristiques des objets, les objets contiennent leurs valeurs propres pour chacune de ces caractéristiques lorsqu'ils sont instanciés).

****

**3.2 Glossaire**

**Mentor :** Conseiller avisé qui sert de guide à quelqu'un.

**Mentoré :** Personne qui est encadrée par un **mentor** dans le but de s’ améliorer.

Le mentor guide le mentoré dans ses apprentissages, il favorise le partage d'expériences et le développement du savoir, du savoir-faire et du savoir-être du mentoré.

**Session de mentorat :** Rendez-vous entre un Mentor et son Mentoré afin qu’il puisse discuter, le mentor en profitera pour analyser et s’assurer de la progression de son mentoré sur ses projets.

**Efficacité :** Capacité d' obtenir le résultat souhaité ou attendu, d'atteindre l'objectif fixé. Les objectifs peuvent se décliner selon un ou plusieurs critères. Degré de réalisation des objectifs d'un programme.

**Rôle :** Fonction remplie par quelqu’un (Administrateur, Mentor ou Mentoré).

**Année d’expérience :** Nombre d’année à pratiquer dans un domaine spécifique (Médecine, comptabilité ou employé de la tech)

**Fréquence :** Nombre de fois qu'un phénomène périodique se reproduit par unité de temps (exemple : une session par semaine).

**Accord de Mentorat :** **Aide apportée par une personne d'expérience à une personne moins expérimentée**,le plus souvent sur une base volontaire, en fonction d'objectifs liés au développement personnel ou professionnel de celle-ci.

**Objectif (ou Spécialité) :** Secteur d’études, d’activité déterminé.

**3.3 Spécifications techniques**

***Langage de programmation JAVA :***

Java est un langage de programmation créé par James Gosling et Patrick Naughton et lancé le 23 mai 1995.

Java est l’un des deux langages de programmation les plus employés à l'heure actuelle, notamment pour des programmes de gestion des entreprises ou de création de jeu vidéo. Du fait du développement de l’informatique, ce langage est de plus en plus utilisé par les entreprises. Le langage Java n'est pas seulement un langage de programmation, il inclut aussi des outils permettant de développer des interfaces graphiques via des logiciels comme Android Studio pour le développement d'applications mobiles.

***Langage de programmation PHP :***

PHP : HyperText Préprocesseur plus connu sous son sigle PHP, est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP, mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.

PHP est un langage de script utilisé le plus souvent côté serveur : dans cette architecture, le serveur interprète le code PHP des pages web demandées et génère du code (HTML, XHTML, CSS par exemple) et des données (JPEG, GIF, PNG par exemple) pouvant être interprétés et rendus par un navigateur web.

Il a été conçu pour permettre la création d'applications dynamiques, le plus souvent développées pour le Web. PHP est le plus souvent couplé à un serveur Apache bien qu'il puisse être installé sur la plupart des serveurs HTTP. Ce couplage permet de récupérer des informations issues d'une base de données, d'un système de fichiers (contenu de fichiers et de l'arborescence) ou plus simplement des données envoyées par le navigateur afin d'être interprétées ou stockées pour une utilisation ultérieure.

***Appliquer la méthode de gestion de projet Scrum :***

La méthode SCRUM se base sur l’approche Agile de la gestion de projet. En effet, tout comme la méthode Agile, il s’agit ici de découper le projet en plusieurs lots. La validation du client est nécessaire à chaque étape pour avancer dans le projet. Mais la méthode SCRUM va encore plus loin. En effet, son objectif principal est d’améliorer la productivité des équipes. Pour y parvenir, une analyse de chaque étape est prévue. Cela permet notamment à l’équipe projet d’améliorer sa façon de travailler. La rétrospective est essentielle pour se rendre compte des écueils et de la meilleure façon de les éviter pour la suite du projet.

Cette méthodologie est idéale pour bénéficier d’une vue d’ensemble du projet. Et ça, c’est essentiel pour ne pas perdre de vue l’objectif final. L’approche SCRUM permet surtout d’améliorer la qualité du produit final. En effet, le produit est constamment amélioré grâce aux nombreux retours du client. Et l’équipe en charge du projet se perfectionne au fur et à mesure de l’avancée du dossier, ce qui permet à terme, d’augmenter sa productivité.

***Appliquer le principe du Domain-Driven Design :***

Le DDD n’est pas une méthode de modélisation mais plutôt une façon de penser ou encore une philosophie de conception. Son objectif est de définir une vision et un langage partagés par toutes les personnes impliquées dans l’élaboration d’une application, afin de mieux en appréhender la complexité.

Domain-Driven Design est préconisé pour le développement de systèmes complexes principalement axés sur des activités, des tâches, des événements, des règles métiers importantes et implique la collaboration entre experts du domaine et experts en conception/architecture. Quand on sait que l’absence d’une compréhension partagée entre eux constitue un obstacle majeur à la réussite des projets, Domain-Driven Design est la méthodologie pour remédier à cet obstacle.

Inclus l’utilisation de diagrammes tels que diagramme de classe, de cas d’utilisation et de contexte.

***Elaborer une interface cliquable :***

Un prototype cliquable est une représentation haute-fidélité d’un service sans qu’il soit développé. Il montre comment le service fonctionne, ses fonctionnalités principales et les parcours utilisateurs principaux. Pour arriver à ce résultat, vous allez faire des choix tout au long de votre travail de conception. Cela commence en amont par de la recherche utilisateur.

En tant qu’UX designer, vous devez prendre en compte les attentes et les besoins réels des utilisateurs cibles ; autrement dit, ceux qui utiliseront le service ou le produit. Et cela est vrai pour concevoir n’importe quel type de support. Cela veut donc dire que vous devez vous appuyer sur de l’information objective et fiable, collectée et observée sur le terrain ou obtenue grâce à des questionnaires ou des entretiens.

Inclus l’utilisation de wireframes.

*Exigence technique*

● Les solutions d'architecture de FuzeScrum sont bâties soit en Java avec Spring.

● Les applications web peuvent être construites avec des technologies adaptées mais doivent s'appuyer sur des services Java.

Accessibilité

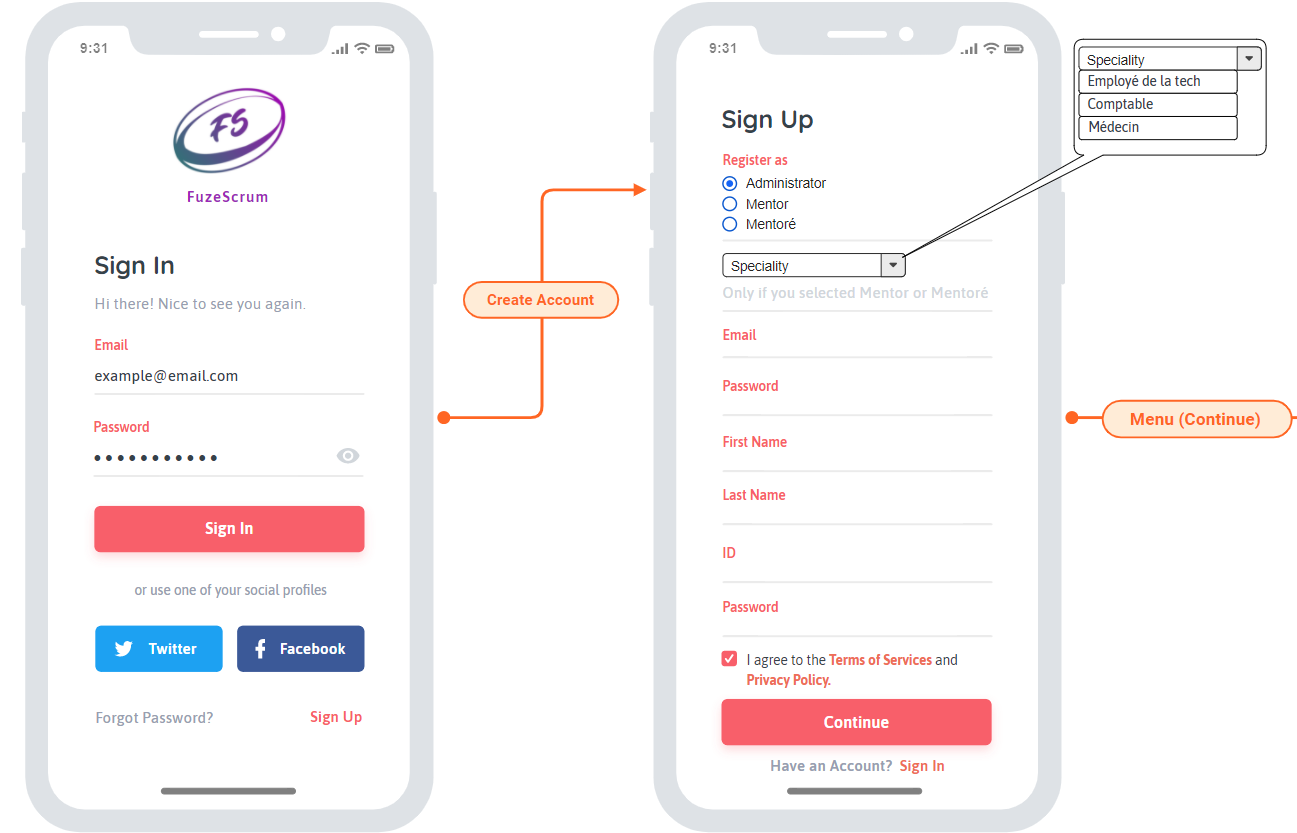
● Les applications web doivent prévoir l'évaluation de leur conformité d'accessibilité w3c avant leur déploiement final.

Java est complètement orienté objet. Il me permet et me pousse même à développer vos applications d'une façon orientée objet et donc d'avoir une application bien structurée, modulable, maintenable beaucoup plus facilement et efficace. Cela augmente ma productivité.

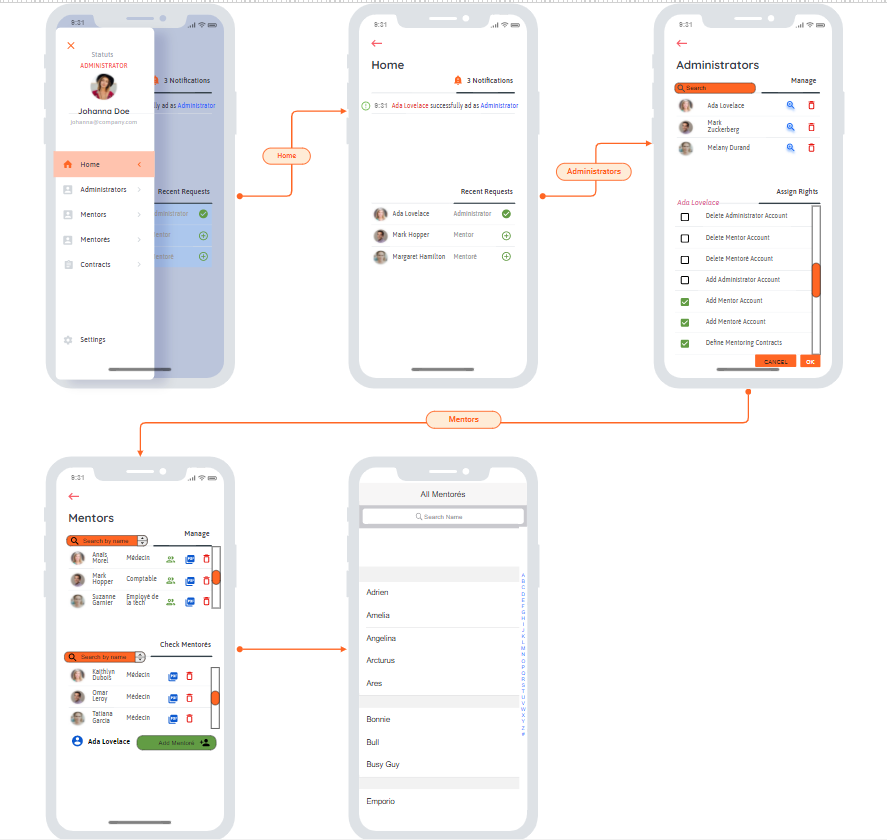
Cela respecte les exigences techniques imposées.

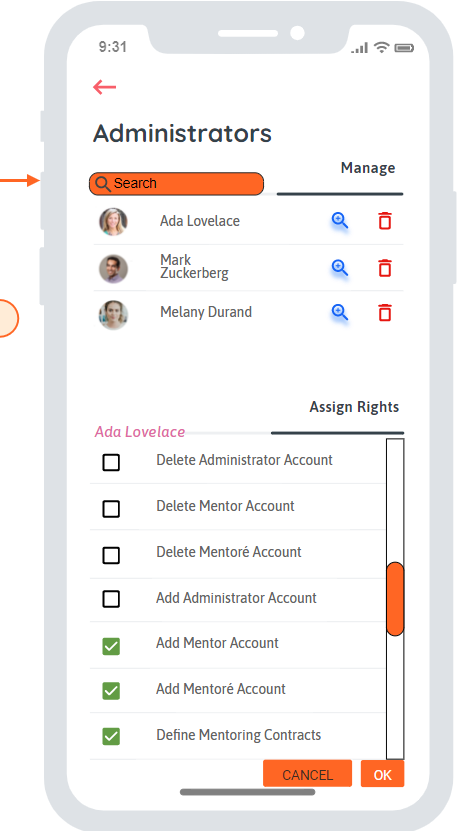
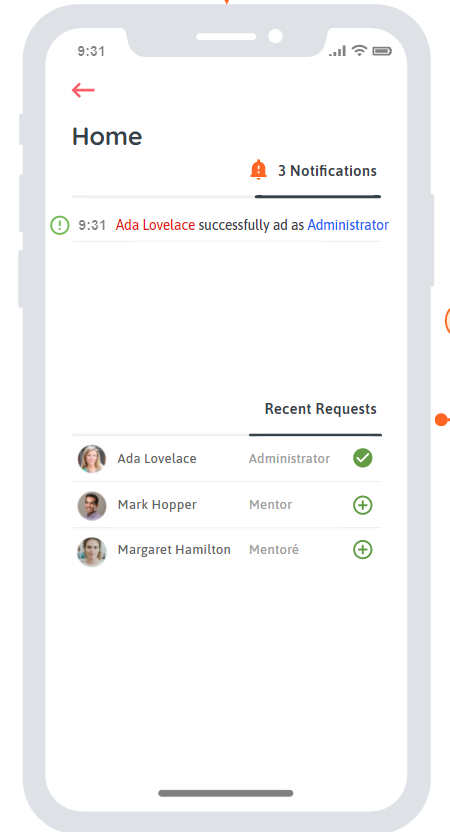
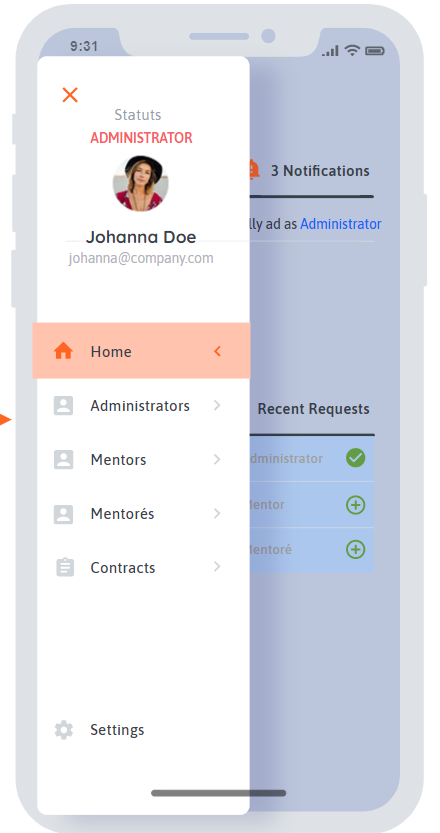
**3.4 Wireframes**

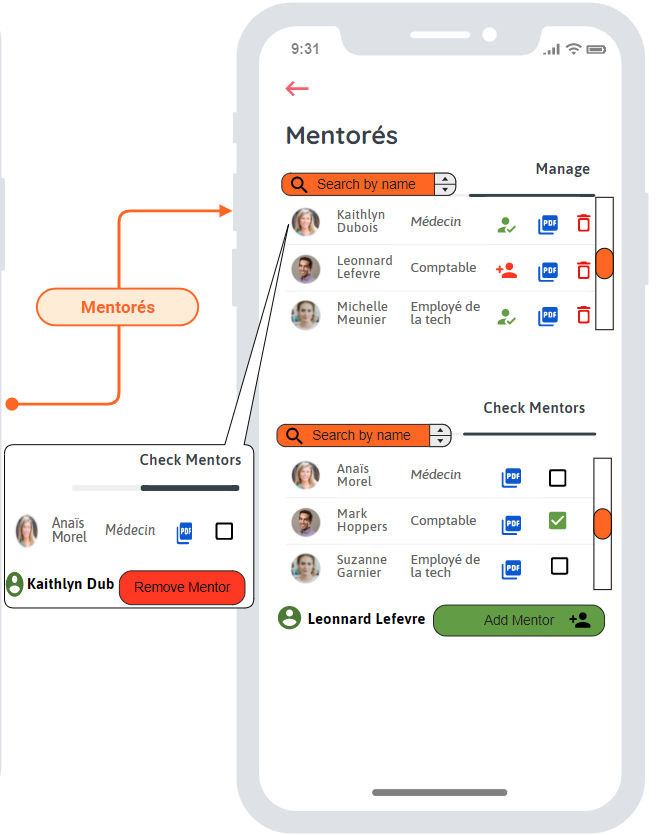
**Se connecter, s’inscrire :**

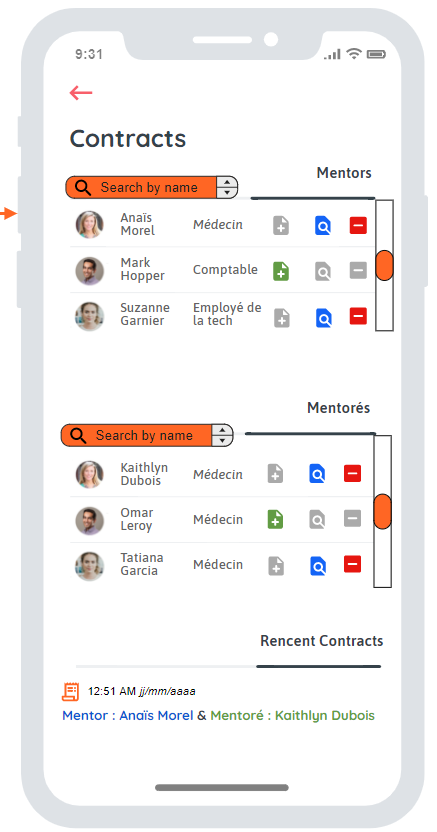
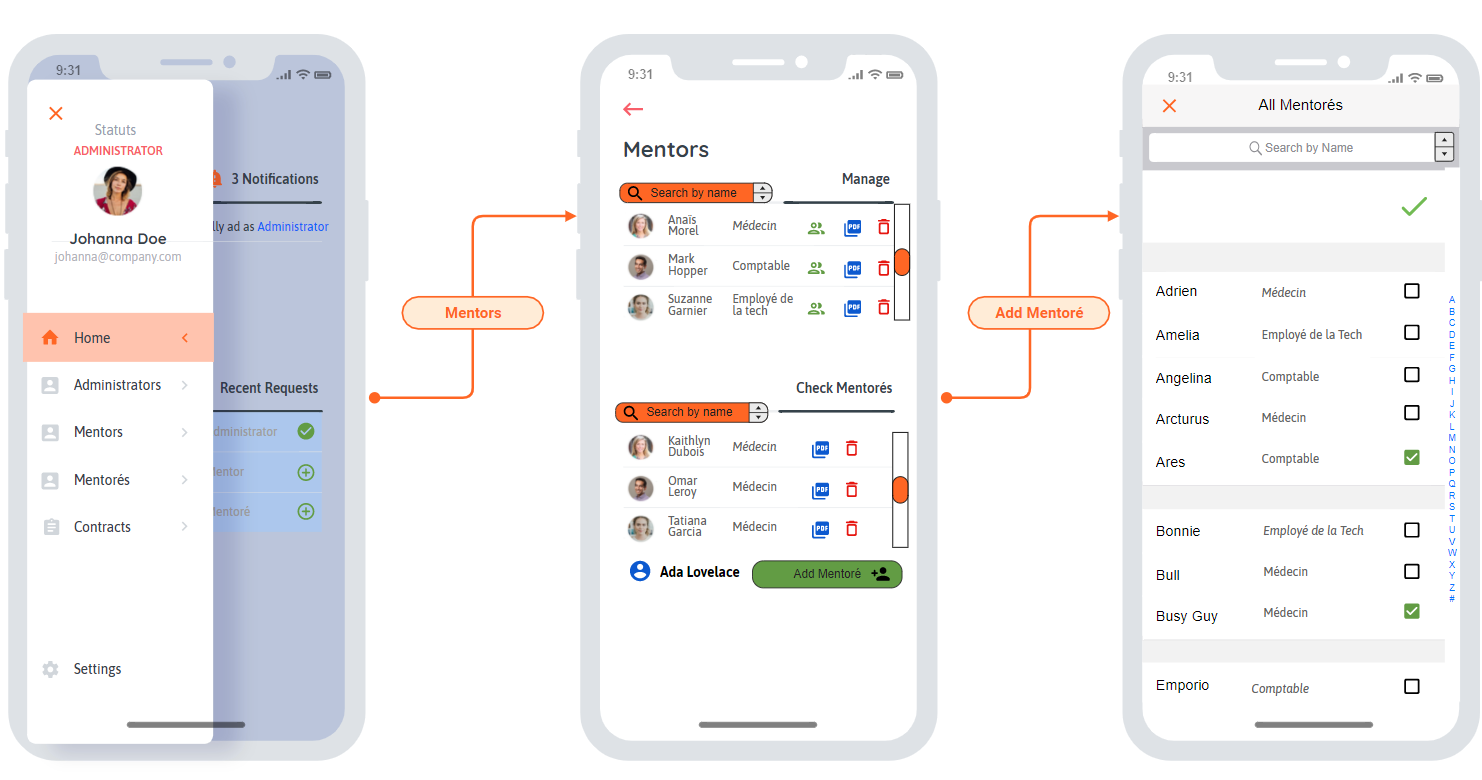


**Vue Administrateur :**

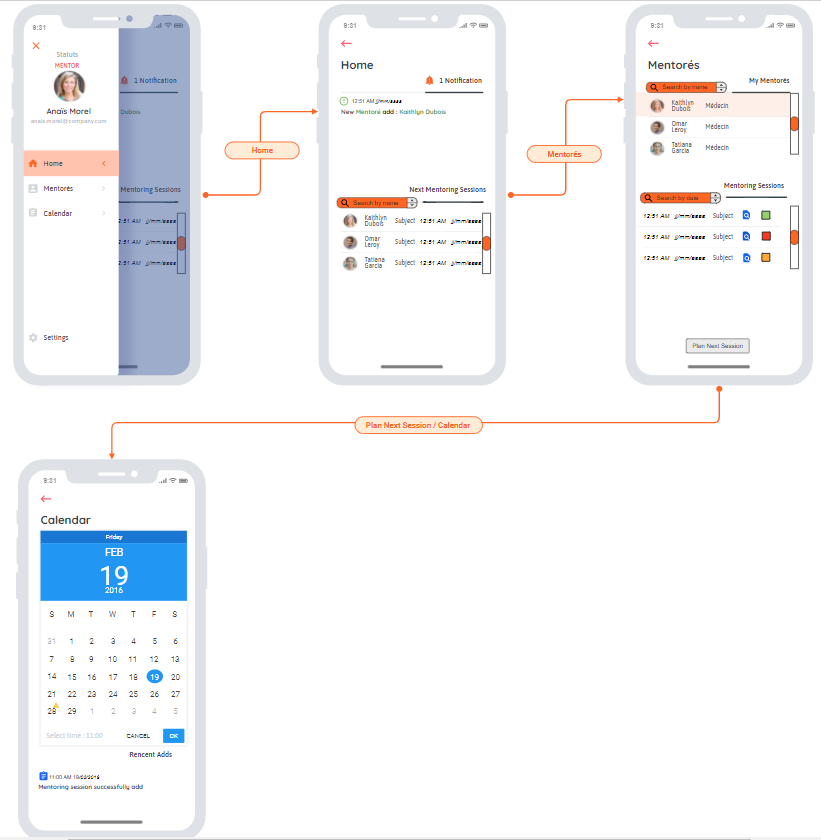
****

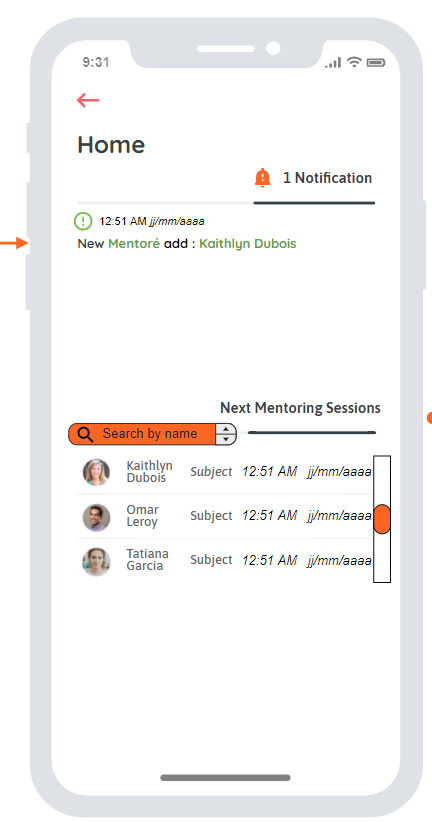
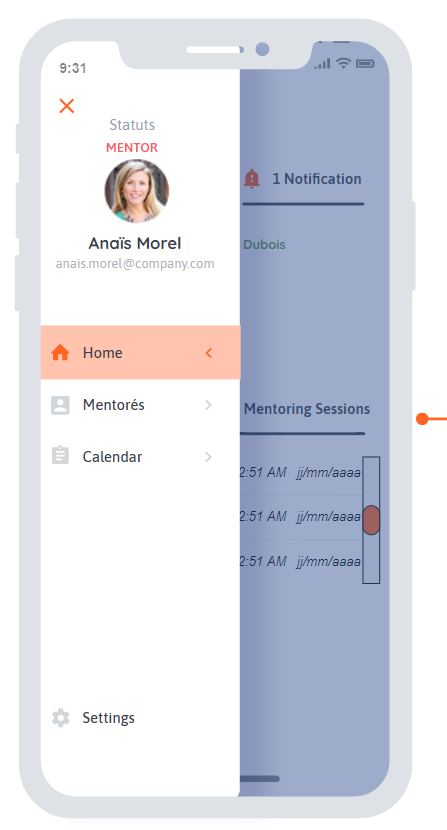
****

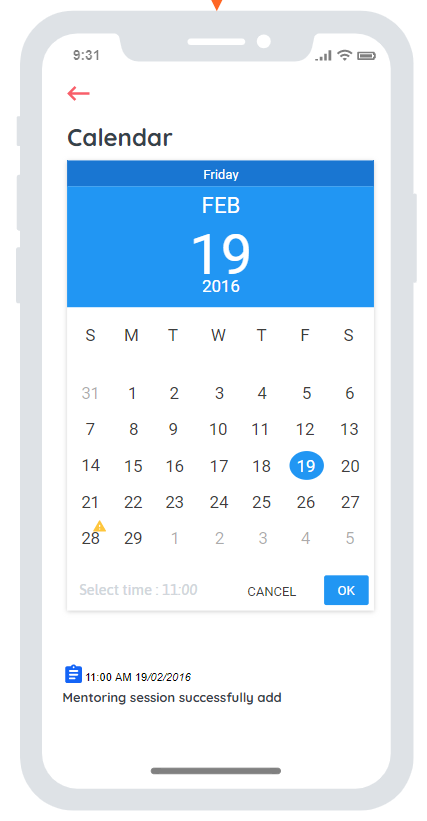
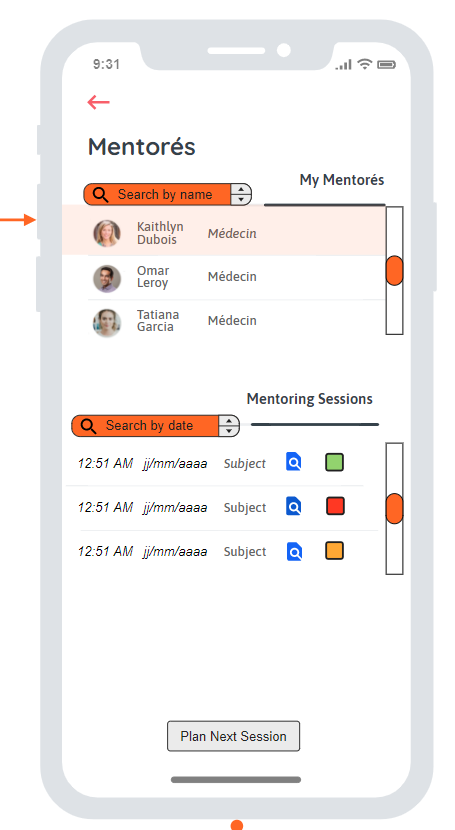
****

****

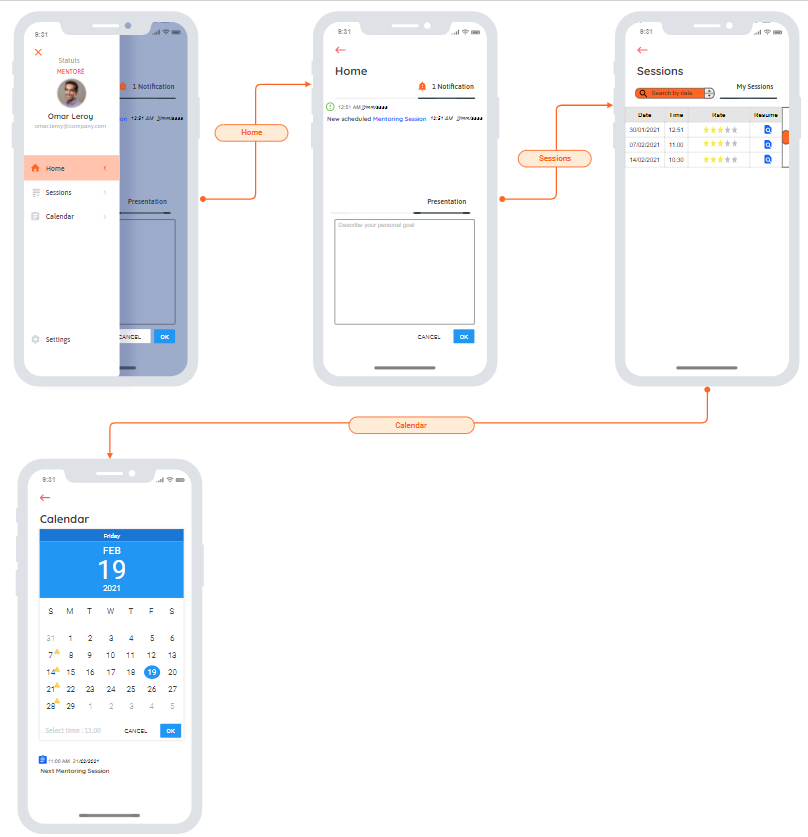
**Vue Mentor :**

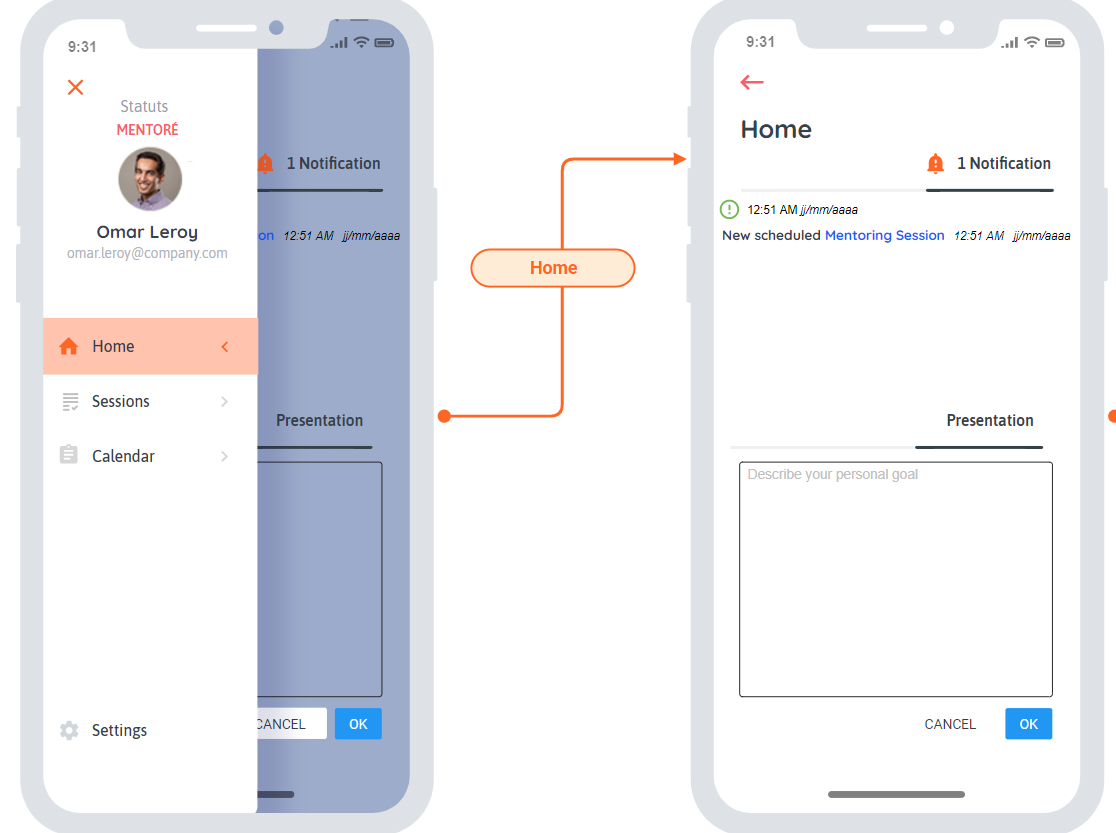
****

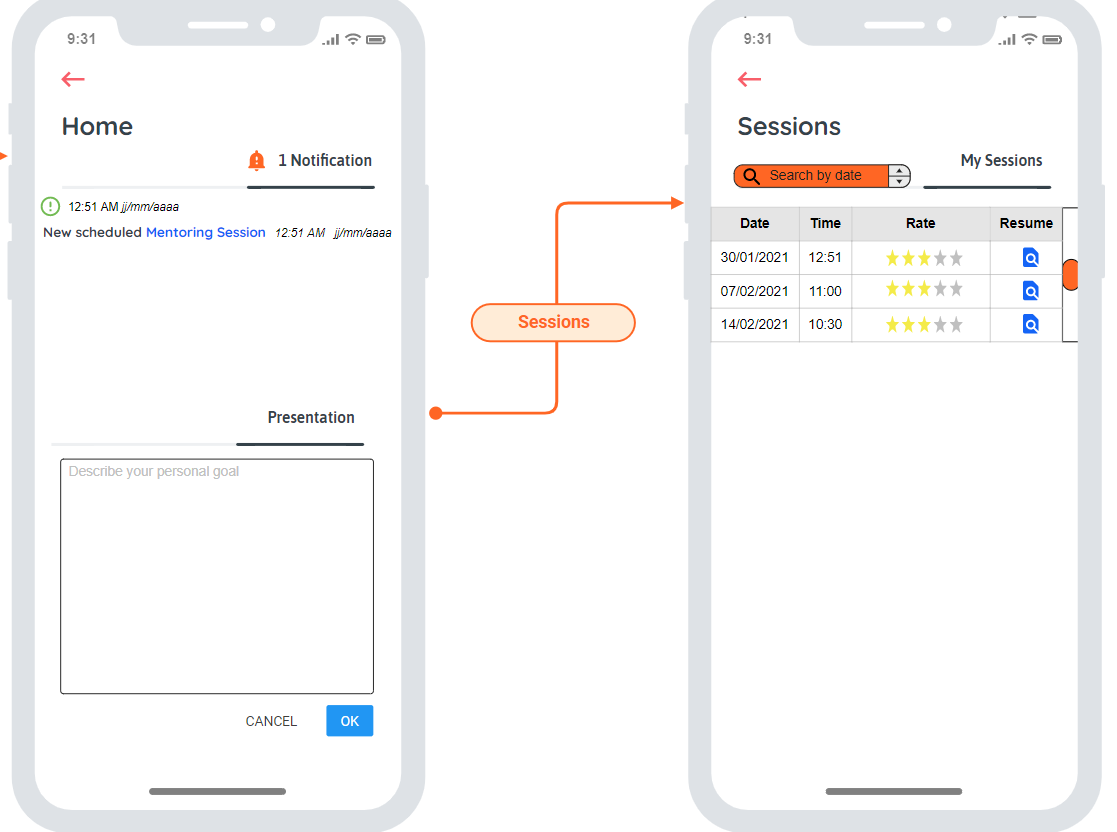
****

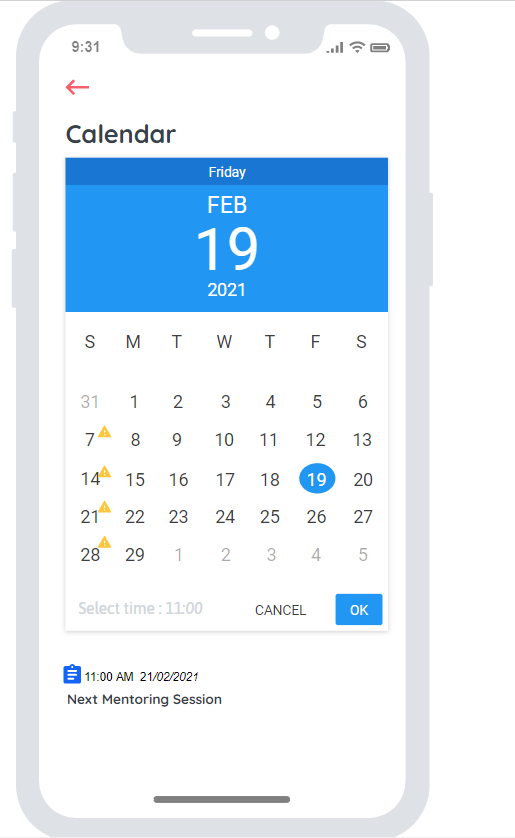
****

**Vue Mentoré :**

****

****

****

****