BERRUYER William
PORTE Eden
GROS Antoine
LARGUIER Luka
MEASSON Emerik
BAPTISTA Alain

Groupe 5
C1

CloudStudio

Dossier 2e itération - Projet Semestre 3

Table des matières

1. Cadrage du projet

1.1 Description du projet

Le projet vise à aider des musiciens à créer leur musique en groupe. La création d'un morceau implique plusieurs personnes, des musiciens, des producteurs, des ingénieurs, avec des tâches différentes, mais chacune dépendant d'une ou plusieurs autres taches. Pour travailler correctement, les créateurs travaillent ensemble, dans la même pièce. Lors du confinement, c'était complètement impossible pour eux de se voir, ce qui fait que leurs projets étaient souvent en attente, et pour le peu de projets qui ont continué, la communication entre les différents participants était compliquée. Ils devaient tous envoyer leur partie du travail aux autres avec des outils qui ne sont pas optimisés pour cela, ce qui est assez pénible et contraignant. En effet, chaque personne se retrouvait avec beaucoup de versions du projet différentes, et il leur était alors compliqué d'avancer avec des versions différentes ou à des moments différents.
Le but de notre projet est de palier aux problèmes des différentes versions, de la sauvegarde et de l'optimisation d'échange de fichier pour ce public.
Le projet se fera sous la forme d'un site web, car les musiciens travaillent sur ordinateur ce qui rend un ce choix « logique », car il correspond à l'environnement de travail des utilisateurs.
De plus cela permettra qu'il soit accessible à tous facilement, peu importe la plateforme sur laquelle l'utilisateur travaille, et peu importe le matériel des utilisateurs.

1.2 Objectifs du projet

⇒ Le projet permet la coordination et le transfert de fichier audio entre tous les participants d'un projet musical, mais également la création de versions d'un même projet afin de pouvoir revenir sur des versions précédentes si nécessaire.
De manière plus détaillée, un musicien pourra poser un fichier sur le site et n'importe quel autre collaborateur du projet pourra récupérer ce fichier, pour le modifier, par exemple, mais les dernières versions restent disponibles en cas d'erreur de l'utilisateur ou de mécontentement de son travail. Le mode de participation sera asynchrone, chacun pourra créer une version et importer ou exporter son travail comme il le souhaite à n'importe quel moment.

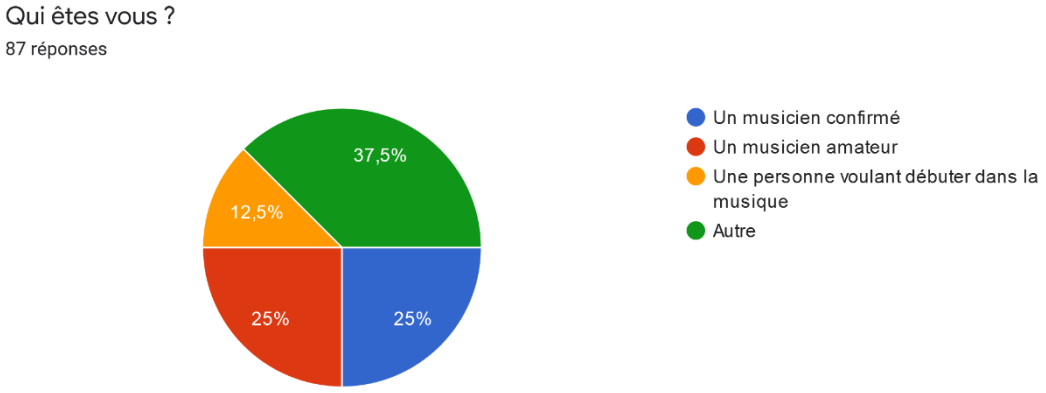
⇒ L'objectif principal est de faire une interface facile à prendre en main mais néanmoins très complète permettant aux utilisateurs de se transmettre des fichiers audios de manière sécurisée à travers un groupe et d'organiser l'avancée de leur projet musical.

⇒ Nous sommes un groupe de 6 étudiants, et nous connaissons déjà suffisamment les langages de programmation nécessaire pour commencer le projet, voir même bien l'avancer sans énormément de difficultés au départ, il y aura néanmoins un certain challenge, car nous avons beaucoup de choses à découvrir, surtout pour ce qui est de l'hébergement du site web, mais aussi pour les fonctions plus poussées du site. Ce projet à donc des objectifs atteignables et réalisables dans les contraintes temporelles imposés.

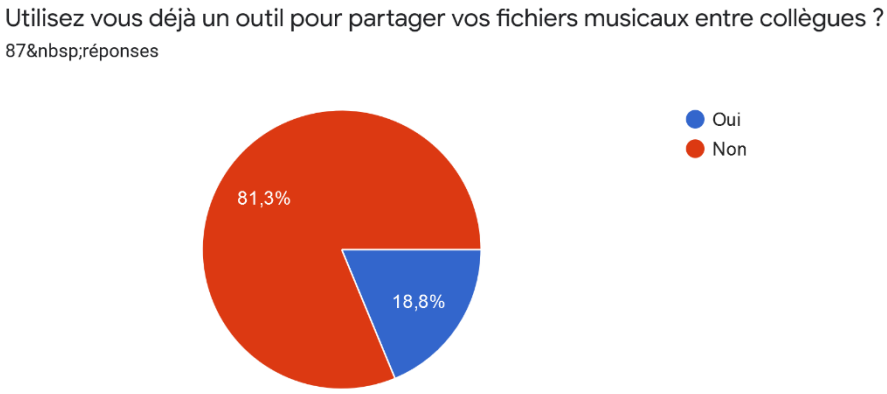
⇒ Au niveau du temps, nous avons une date butoir qui nous laisse 17 semaines de projet, mais le projet sera découpé en plusieurs parties pour une meilleure organisation. Chaque élève aura son travail et une durée limitée pour le faire. S'il est en difficulté, un autre l'aidera pour ne pas perdre trop de temps et avoir fini les fonctionnalités principales au minimum pour la fin du projet.

Analyse du terrain

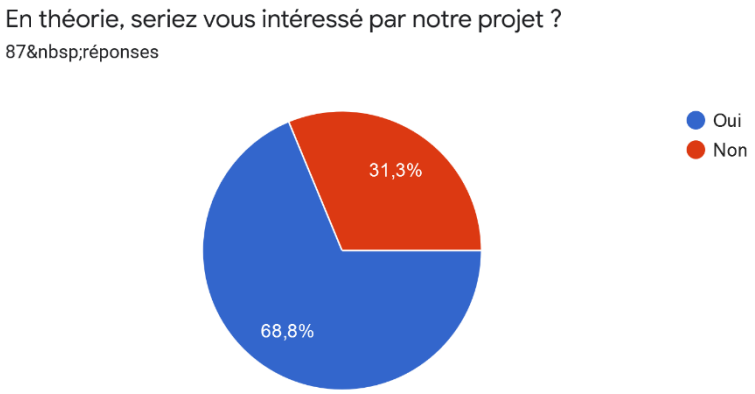
Pour notre analyse de terrain, nous avons procédé de deux manières différentes. La première chose que nous avons faite a été de créer un Google Form (un questionnaire) pour questionner les utilisateurs sur l'intérêt que ceux-ci pourraient porter au projet ainsi que sur leurs habitudes de travail en ce qui concerne la production musicale. Voici quelques-unes des réponses que nous avons récoltées.



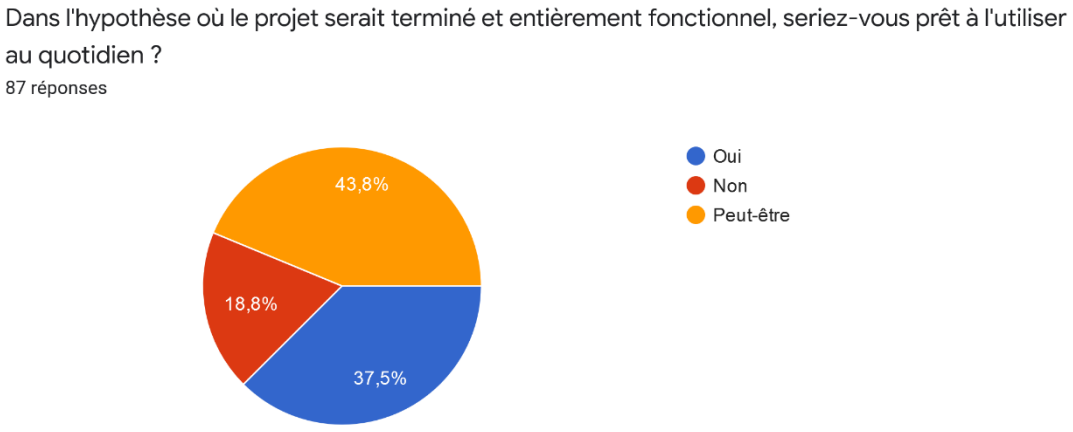
Graphique 1 Réponses obtenues au questionnaire : Profils utilisateurs



Graphique 2 Réponses obtenues au questionnaire



Graphique 3 Réponses obtenues au questionnaire



Graphique 4 Réponses obtenues au questionnaire

La seconde méthode que nous avons utilisée est l'interview d'utilisateurs potentiels (notamment grâce à l'outil zoom en ces temps de confinement). Nous avons interviewé des personnes représentant différents profils d'utilisateurs potentiels :

- Des musiciens confirmés
- Des musiciens amateurs
- Des personnes ne pratiquant pas d'activité liée à la musique
- Des musiciens amateurs, développeurs séniors de métier

Ces deux méthodes nous ont permis de discerner que les habitudes des musiciens en termes de production musicale en groupe se résument souvent à l'utilisation de matériel physique tel que des disques durs fautes de plateformes en ligne permettant d'avoir un workflow suffisamment efficace. De plus nous pouvons souligner certains points importants auprès des utilisateurs que nous avons sondés. Tout d'abord, beaucoup nous ont expliqué qu'ils apportés une grande importance à l'interface de la plateforme d'un point de vue visuel, mais également d'un point de vue d'efficacité par rapport

à l'utilisation des fonctionnalités de celle-ci. De plus, la vitesse de transfert des fichiers en entrée et en sortie de la plateforme est un autre point qui a été beaucoup mentionné. Enfin, la polyvalence du site a beaucoup été mentionnée, c'est-à-dire une forte accessibilité pour les musiciens débutants, il faut donc que la plateforme soit facile à prendre en main rapidement pour des personnes novices. Mais il faut aussi qu'elle offre des fonctionnalités suffisamment poussées pour avoir un intérêt auprès des musiciens plus expérimentés. Cela nous permettrait donc de pouvoir viser à la fois des utilisateurs pratiquant la musique de façon amateur ainsi que les professionnels.

Analyse de l'existant

Actuellement, nous avons observé que dans la réalité le partage et l'organisation de fichier dans des projets musicaux se font majoritairement à l'aide de disque dur externe ou de support physique (clé USB …). Ces disques durs sont transmis et déplacés sur les lieux de travails communs aux différents acteurs impliqués dans le projet (en Studio par exemple). Un autre outil qui est fréquemment utilisé est le stockage cloud, avec des outils comme google drive ou Dropbox.

Il existe actuellement une plateforme qui se rapproche de la nôtre dans le concept. Cette plateforme s'appelle Splice. Cette plateforme propose différentes fonctionnalités séparées les unes des autres. La principale n'étant pas l'échange de fichier et la gestion de projet, mais consiste plus à une grosse banque de son libre de droits accessibles via un abonnement mensuel ou annuel. Il y a également une partie appelée *Studio* sur la plateforme qui est la partie se rapprochant de notre idée. Cette partie de la plateforme permet de créer des projets musicaux et d'en créer des versions. Splice utilise un logiciel qui sert d'extension et qui est téléchargeable sur Windows et macOS et qui fait le lien avec leur plateforme en ligne. La gestion des versions sur leur plateforme fonctionne de la manière suivante :

Le dossier contenant le projet doit être contenu dans les dossiers prévus à cet effet par le logiciel. C'est-à-dire dans un emplacement prévu à cet effet sur l'ordinateur de l'utilisateur (de base : C:\Users\NomUtilisateur\Documents\Splice), le logiciel permet également dans ses options de choisir de sauvegarder depuis d'autre emplacement choisi par l'utilisateur.

À chaque fois que le fichier de projet est écrasé par l'utilisateur, c'est-à-dire qu'à chaque fois que l'utilisateur fait une nouvelle sauvegarde de son projet dans son logiciel de production de musique cela créer une nouvelle version sur la plateforme en ligne automatiquement. C'est-à-dire qu'à chaque fois que le fichier est écrasé sur l'ordinateur de l'utilisateur (à chaque fois qu'on sauvegarde le projet, il est automatiquement écrasé et remplacé par la nouvelle version) le logiciel associe cette action à la création d'une nouvelle version.

De plus, quand Splice crée une nouvelle version le fichier de projet est automatiquement uploadé dans la version et le nom de la version est également donné automatiquement (et sera forcément du format : Vx, x étant le numéro de la version).

Cette fonctionnalité est loin d'être optimale d'un point de vue organisation, car lors de la création d'une musique il est très fréquent de sauvegarder son fichier de projet, en réalité on sauvegarde souvent plusieurs fois par minute, ce qui fait que les versions s'accumulent très vite sur la plateforme. Le projet est donc rapidement saturé de versions qui ont toutes très peu de différence les unes des autres si on sauvegarde le projet musical à des intervalles fréquents.

Cette plateforme permet également d'ajouter des collaborateurs au projet, nous nous sommes inspirés de cette fonctionnalité pour notre site, néanmoins sur Splice Studio les fonctions de collaborations sont rapidement limitées. La plateforme permet de rechercher des personnes sur le site pour les ajouter comme collaborateurs, soit via adresse mail soit via pseudonyme.

Pour le service de stockage dans le cloud, il existe des plateformes telles que Google Drive ou Dropbox qui sont prévus pour le stockage en ligne de fichier, mais aussi le partage de document, mais qui ne sont pas prévus pour gérer des projets. Nous voulons utiliser le même principe que ces plateformes, c'est-à-dire que les fichiers déposés sur la plateforme par l'utilisateur sont stockés sur un serveur distant à l'utilisateur. Sur ces plateformes si un utilisateur dépose deux fichiers ayant exactement le même nom le deuxième fichier est automatiquement renommé en rajoutant un (1) derrière son nom original. C'est une fonctionnalité que nous pourrions potentiellement exploiter pour notre projet.

Pour la partie chat du site, il existe bien évidemment de nombreuses plateformes incorporant des fonctionnalités de chat, Reddit est un exemple, des plateformes comme Git permettent également de communiquer à travers des chats. Dans Splice Studio il n'y a pas de fonctionnalités de chat général par rapport au projet. Néanmoins Splice Studio permet la possibilité d'ajouter des commentaires pour chaque version, c'est une fonctionnalité dont nous allons nous inspirer pour notre plateforme.

Pour la partie permettant de créer des versions de chaque projet il existe différentes plateformes actuellement, on peut citer Git pour le code par exemple. Splice Studio propose également de faire des versions de chaque projet. Dans Splice Studio il est possible d'ajouter des fichiers audios à la version (les stems ou piste par piste) pour que les autres collaborateurs puissent y accéder.

1.4 Parties prenantes

Le domaine d'activité ciblé par notre projet est assez vaste. Nous souhaitons que notre application web puisse être utilisée aussi bien par des professionnels du milieu de la musique (artistes, groupes, ingénieurs) que par des amateurs voulant créer leur propre projet tout seul ou avec des partenaires sans qu'ils aient de difficultés par rapport à la gestion des nouvelles versions de leur projet.
C'est pour cela que notre application doit être facile d'accès à n'importe qui, qu'il soit ou non familier avec l'outil informatique et les outils de musique numérique.

Notre application finale se doit d'être collaborative, afin de permettre à chaque utilisateur, à chaque membre d'un même projet de pouvoir communiquer ensemble durant le suivi de leur projet et ainsi permettre une meilleure organisation et un risque plus faible d'avoir des informations différentes entre chaque membre du groupe, car leurs fichiers seront centralisés sur une même plateforme, et ils pourront commenter chaque nouvelle version pour savoir laquelle est la bonne et les modifications ayant été apportés dessus. Les utilisateurs attendent de notre plateforme un site fiable et sécurisé afin d'avoir les bonnes informations au bon moment sans pour autant risquer une perte de donnée ou que d'autres utilisateurs puissent avoir accès à leurs données.

Durant ce projet nos compétences doivent englober plusieurs domaines, tout d'abord nous devons faire un site web cohérent et moderne, de plus il doit être facile à utilise, ce qui fait qu'il doit avoir un design "responsive", mais également adaptée au public visé, nous ajoutons à cela la compétence de gestion de base de données, que nous devrons créer et mettre en place sur notre plateforme, et ce de manière sécurisée.

Durant ce projet nous ne nécessitons pas l'intervention de sous-traitants, car notre équipe comporte des membres spécialisés dans chaque domaines ou langages que nous utiliserons.

1.5 Contraintes

La première contrainte qui se pose à nous est la gestion de la synchronisation des fichiers et version du projet. En effet, chaque membre d'un projet doit avoir accès aux mêmes versions, nommées de la même façon, et il ne faut pas qu'il y ait de "décalage", entre les utilisateurs. Pour gérer cela, il faut que chaque fois qu'un fichier est mis à jour, la base de données actualise automatiquement la version mise précédemment sur la plateforme.
La seconde contrainte est celle du nombre de fichier et donc de projets que nous pouvons stocker. En effet, pour ce projet nous travaillons avec un Raspberry Pi équipé de 32Go de stockage en tant que serveur qui hébergera notre base de données. Nous sommes donc bien évidemment limités en ce qui concerne le stockage des projets. Pour ce prototype nous avons donc décidé de limiter le nombre de version d'un projet qu'un utilisateur pourra créer ainsi que le nombre de projets que celui-ci pourra créer.
La troisième contrainte est la capacité de vitesse des transferts de fichier sur la plateforme, encore une fois le fait de travailler avec un Raspberry pi nous limite d'un point de vue des vitesses de transferts. Il faut donc que nous optimisions notre plateforme afin que les temps de transferts soient suffisamment rapides pour ne pas gêner l'utilisateur.
Il faut également que le serveur sur lequel est hébergé la plateforme tourne 24h/24, pour permettre aux usagers de travailler à toute heure.
Une contrainte supplémentaire est la contrainte temporelle, nous n'avons que 17 semaines pour réaliser notre projet nous devons donc être efficaces afin d'atteindre nos objectifs.
De plus, pour anticiper de potentiels dégâts physiques sur le serveur hébergeant la base de données, il faut réaliser régulièrement une sauvegarde de cette dernière.
Enfin, une des contraintes majeures est une contrainte d'ordre légale. En effet, notre projet va contenir de nombre projet musicaux, nous devons donc définir parfaitement comment les droits d'auteurs sont répartis entre les collaborateurs d'un projet, quels droits possède la plateforme sur les projets, etc…
Nous devons donc répondre à toutes ces interrogations afin que les droits d'auteur soient bien respectés.

1.6 Risques

Le premier risque, le plus important, est celui de la perte de données. Il est en effet possible que, si la plateforme ne fonctionne pas correctement, les données uploadées par l'utilisateur ne soient pas correctement sauvegardées.
Un autre risque majeur est celui de la mise à jour des versions, qui, si elle n'est pas faite correctement, peut entraîner au minimum une confusion entre les utilisateurs, et donc du temps perdu, au pire, des pertes de données, etc. Nous devons donc faire en sorte que notre base de données gère tous les problèmes liés au stockage des fichiers.
Un autre risque, soit qu'une tierce personne ait accès au projet, par le biais des identifiants d'un utilisateur présent sur le projet, ou de toute autres façons, et que cette dernière supprime, intentionnellement ou non, des fichiers du projet. C'est pour cela qu'une sécurité va être mise en place, nécessitant l'approbation de plusieurs utilisateurs, participant au projet, avant la suppression définitive d'un fichier.
Les risques temporel et humain doivent également être pris en compte, en effet il est possible que nous n'arrivions pas à atteindre les objectifs que nous nous sommes fixés dans le temps impartis.
Enfin, il existe le risque que notre serveur rencontre des problèmes de stockage supprimant le contenu stocké sur celui-ci. Ce risque est bien évident inacceptable, nous devons donc faire en sorte que cela n'arrive jamais, en utilisant des backups de nos données au cas où nous aurions un problème technique sur notre serveur principal.

1.7 Suivi des contraintes et risques

Actuellement, nous n'avons pas eu de problème lié aux contraintes ou aux risques que nous anticipions. Nous n'avons pas encore assez avancé dans le développement de notre plateforme pour identifier certains problèmes, techniques notamment.

2. Expression du besoin

2.1 Besoins fonctionnels et non fonctionnels

Le site doit être entièrement rédigé en anglais, car l'anglais est la langue de référence en matière de production de musique, en particulier assistée par ordinateur. De plus l'anglais permettra de toucher un maximum de public.

Le site doit permettre à l'utilisateur de créer un compte ou de se connecter à son compte si celui -ci en possède déjà un. L'utilisateur doit également pouvoir modifier les informations de son compte telles que son nom, son nom d'artiste, sa photo de profil, son adresse mail, son mot de passe, ses préférences en termes de notification.
L'utilisateur doit pouvoir se déconnecter de son compte.

Le site doit permettre à l'utilisateur de créer des projets, un projet correspond au travail sur un morceau de musique, mais doit aussi pouvoir créer des groupes de projets (l'équivalent d'un album). L'utilisateur doit pouvoir accéder à ses projets depuis le site une fois qu'il est connecté à son compte. Dans chaque projet l'utilisateur doit pouvoir accéder à différentes sections correspondant à l'étape d'avancement de son projet, ces étapes sont :

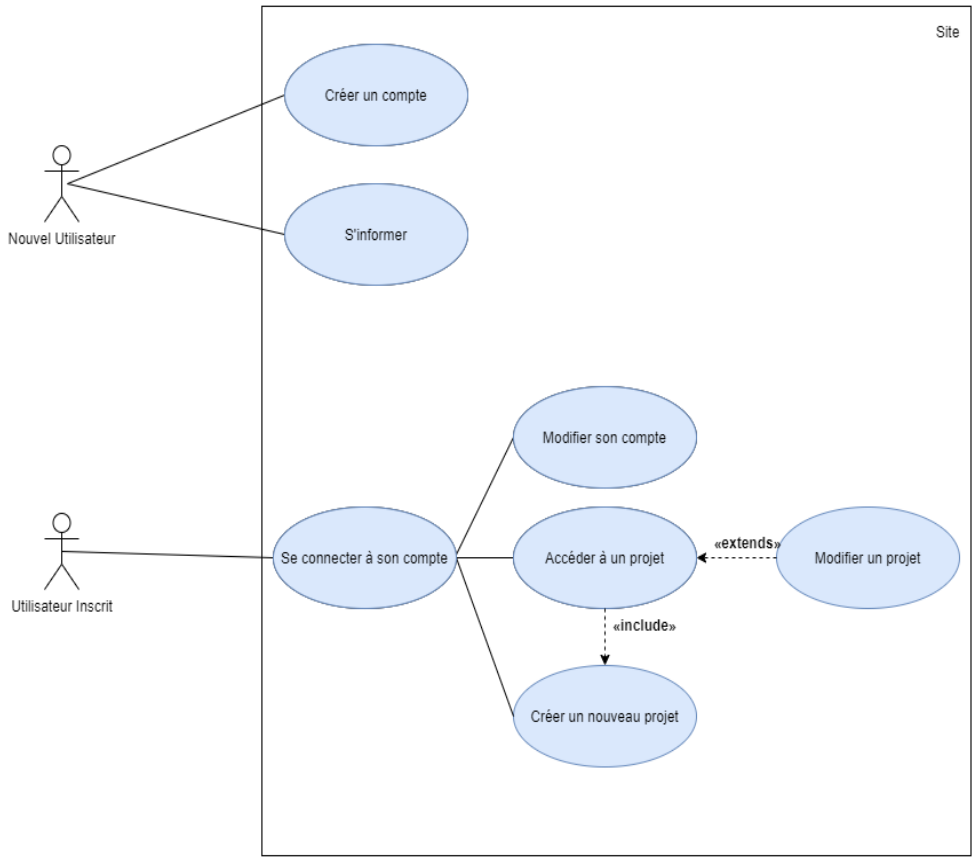
- Raw Session qui correspond à une version brute du morceau sans effets ni post-production.
- Mix qui correspond à une version mixée du morceau (c'est-à-dire une version travaillée par l'ingénieur son).
- Master qui correspond à une version Masterisé du morceau c'est-à-dire une version travaillée par l'ingénieur Master pour correspondre aux standards et aux normes de l'industrie. À la fin de cette dernière étape, le morceau est prêt à être publié

À l'intérieur de chaque section, l'utilisateur doit pouvoir créer des versions de cette section afin d'avoir des backups du projet au cas où l'utilisateur voudrait revenir sur une version antérieure à la version sur laquelle il travaille. Dans chaque version de son projet l'utilisateur doit pouvoir importer sur le site le fichier de projet correspondant à son morceau, ainsi que chaque piste du morceau s'il le souhaite, mais aussi une version de démonstration du morceau permettant d'écouter le rendu de la version directement sur le site. Pour chaque version l'utilisateur doit pouvoir rédiger une description de la version afin de mieux l'identifier et de la distinguer parmi les autres. L'utilisateur doit également pouvoir laisser un commentaire pour chaque version s'il veut s'exprimer sur le contenu de celle-ci. L'utilisateur doit pouvoir écouter les pistes du morceau une par une s'il le souhaite, néanmoins il faut que celle-ci ait été importée à l'avance. L'utilisateur doit pouvoir télécharger chaque piste d'une version, une par une, plusieurs à la fois ou toute d'un coup ; il peut également télécharger le fichier de projet de la version. Chaque version doit afficher sa date de création.
Un utilisateur qui crée un projet en est le propriétaire est donc l'administrateur de celui-ci, cet utilisateur doit pouvoir ajouter des collaborateurs à son projet, mais aussi leur attribuer des rôles (Guitariste/Batteur/etc.).
L'utilisateur doit pouvoir, dans chacun de ses projets, accéder à un chat général accessible par tous les collaborateurs du projet. Ce chat permet de discuter à propos du projet dans sa globalité. L'utilisateur doit être informé par mail des messages échangé dans le chat.

L'utilisateur doit modifier les paramètres de ses projets, tels que l'image du projet ou son nom, il doit également pouvoir supprimer ses projets s'il le souhaite.
Lors de la création d'un projet l'utilisateur doit pouvoir choisir s'il veut créer un projet correspondant à un projet simple (Morceau) ou à un projet composé (Groupe de morceau/Album), l'utilisateur doit également pouvoir choisir sur quel logiciel il travail afin que les potentiels autres collaborateurs qui rejoindraient le projet en cours de route en soit informé et pour qu'une vérifications du format de fichier de projet se fasse lors de l'import de fichier sur la plateforme (grâce à l'extension de fichier qui est unique à chaque logiciel). Il doit également pouvoir choisir le nom de son projet.

Lors de la création d'une version d'un projet, si cette version n'est pas la première du projet, un utilisateur doit pouvoir choisir de quelle version celui-ci est parti pour concevoir la version qu'il est en train de créer. C'est-à-dire qu'il pourra choisir dans une liste de quelle version il est parti, il pourra toutefois également choisir aucune version précédente. Cette fonctionnalité permet aux autres collaborateurs d'être informé sur quelle version est basé la version qu'ils sont en train de consulter. Ce qui permet d'éviter les problèmes de confusions ou de doublons dans les fichiers personnels des utilisateurs. Par exemple, si deux collaborateurs télécharge la version 2 d'un projet et que ces deux collaborateurs font des modifications sur cette version chacun de leur côté et qu'ils veulent chacun créer une nouvelle version du projets, ils devront chacun expliciter qu'ils sont partis de la version 2, le premier collaborateur à créer sa version créera donc la version 3 et le second la version 4, les autres collaborateurs pourront alors distinguer les deux versions grâce au nom du créateur de la version ainsi qu'à la description des versions.

Lors de son inscription, l'utilisateur doit renseigner son nom, son prénom, sa date de naissance, son nom d'artiste, son adresse mail, et créer un mot de passe. Les informations servant à l'accès à son compte sont son adresse mail et son mot de passe.



Digramme de cas d'utilisation

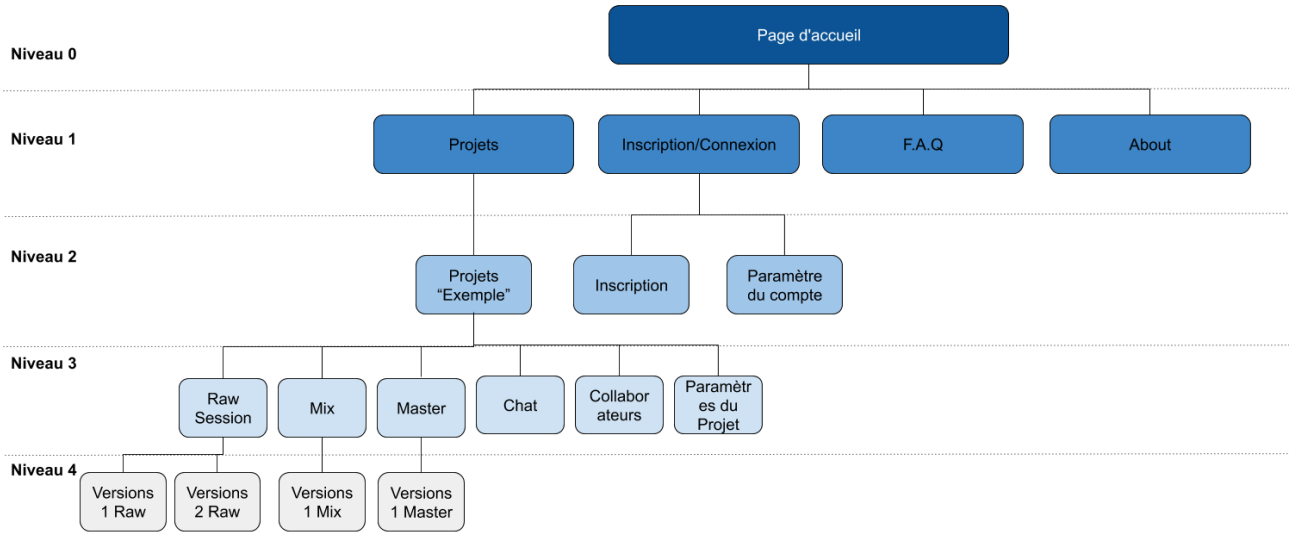
2.2 Hiérarchisation des besoins

Toutes ces fonctions n'ont pas la même importance dans le développement du site, certains sont essentiels à son fonctionnement alors que d'autres ne sont que des ajouts apportant de petites améliorations au site. Nous avons donc hiérarchisé ces besoins afin de visualiser efficacement lesquels sont les plus importants. Cette hiérarchisation des besoins va nous permettre de nous focaliser sur les fonctionnalités principales afin de respecter les contraintes temporelles qui nous sont imposées.

BESOINS FONCTIONNELS	
Priorité	**Définition du besoin**
1	Créer un compte utilisateur
1	Créer / supprimer un projet pour un(des) morceau(x)
2	Diviser son projet en différentes étapes (Raw Session, Mix, Master)
2	Import / Export les fichiers et pistes nécessaires à la conduite du projet
2	Créer / Supprimer une version du projet
2	Rédiger une description pour chaque version du projet créé
2	Envoyer / Lire des messages dans un chat global à chaque projet
3	Ajouter / Supprimer des collaborateurs à un projet (Recherche par pseudonyme ou mail)
3	Download et écouter la démo d'un morceau et les pistes dans une version du projet
4	Modifier son profil
4	Créer des groupes de projet (album)
4	Modifier les paramètres d'un projet
4	Définir des rôles pour les collaborateurs d'un projet
4	Tutoriel pour les nouveaux utilisateurs
4	Envoyer / Lire des commentaires pour chaque version d'un même projet

Les besoins surlignés en bleu doivent impérativement être développés pour le prototype final.

BESOINS NON FONCTIONNELS	
Priorité	**Définition du besoin**
1	Menu ergonomique (sur ordinateur / tablette / téléphone) et utilisation d'un lexique compréhensible pour les débutants
1	Sécurité des bases de données (utilisation de vues, gestion des permissions en lien au site, et à chaque projet)
1	Avoir accès à un fichier à jour dès qu'il est déposé
1	Site toujours allumé et avec une sauvegarde sur un disque dur en cas de panne
2	Un compte administrateur pour avoir la liste des utilisateurs et leur(s) projet(s)
2	Code bien réparti et clair (avec des commentaires pour aider à la compréhension) afin de faciliter la testabilité et la maintenance du site
3	Service gratuit pour profiter à tous



Arborescence du site

2.3 Critères qualité

La facilité d'apprentissage : Notre plateforme doit permettre aux utilisateurs visés (les musiciens) de prendre rapidement en main son interface ainsi que la signification des différents menus et des différentes fonctions.
La facilité de compréhension : Notre plateforme doit être facilement compréhensibles pour tout utilisateur qui voudrait s'en servir qu'il fasse partie du public visé ou non. Nous devons limiter l'utilisation de terme trop technique et implémenter une page permettant d'aider un utilisateur à comprendre l'interface si celui-ci est en difficulté.
La sécurité : Notre plateforme doit empêcher tout accès non autorisé, avec l'utilisation d'identifiant et de mot de passe pour se connecter à son compte, mais il doit également empêcher tout accès non autorisé aux données stockées sur notre plateforme.
L'exactitude : La plateforme doit être précise du stockage des données afin d'être fonctionnel.
L'aptitude : Les fonctionnalités principales de la plateforme doivent toutes être parfaitement fonctionnelles pour que la plateforme fonctionne.
L'efficacité des temps de réalisation : L'import et l'export de fichier doivent être assez rapides pour maintenir une bonne productivité du point de vue de l'utilisateur, l'utilisation des autres fonctionnalités telle que la création ou l'ouverture d'un projet doivent également être rapides.

2.4 Critères ergonomiques

Le guidage

Notre plateforme doit pouvoir conseiller, informer l'utilisateur mais également conduire ses interactions. Elle doit donc être lisible, utiliser un vocabulaire adapté pour que le plus de personne puisse en comprendre le fonctionnement. Elle doit aussi contenir des indications visuelles, et inciter à l'action afin de guider efficacement l'utilisateur pour que la plateforme réponde aux besoins et aux objectifs de l'utilisateur.

La charge de travail

Il convient de réduire au minimum la charge de travail effectué par l'utilisateur. Plus elle est grande, plus le risque d'erreur augmente. Il faut donc limiter les étapes en rendant le parcours fluide. Nous devons réduire la densité informationnelle afin d'améliorer la perception et la mémorisation de l'utilisateur. Enfin, nous devons limiter les interactions superflues pour optimiser la navigation et éviter la perte de temps et d'attention.

Contrôle explicite de l'utilisateur et échange avec l'interface

L'interface communique avec l'utilisateur pour lui indiquer un parcours, étape par étape. Dans tous les cas, l'utilisateur dispose de sa liberté d'action, ne fait pas face aux contraintes et a toujours le choix.

Adaptabilité

Nous devons prendre en compte les habitudes, les exigences et l'expérience des utilisateurs. Pour cela, nous avons déjà réalisé une analyse de terrain et il faudrait dans l'idéal réitérer ce genre d'opération. L'utilisateur se sert de ses expériences passées et de son apprentissage externe pour utiliser l'interface. De plus nous devons adapter notre lexique à tous les publics.

Gestion des erreurs

L'interface doit être conçue d'une telle manière qu'elle protège l'utilisateur des erreurs via des moyens de prévention ou de détection. Si des erreurs sont commises, la plateforme doit informer l'utilisateur sur leur nature, et indiquer la bonne marche à suivre pour les corriger ou les éviter. Les moyens pour corriger ces erreurs sont mis à disposition et aide l'utilisateur à aller de l'avant.

Homogénéité et cohérence de l'interface

Les choix graphiques et interactifs doivent correspondre d'un contexte à l'autre. Une action réussie doit pouvoir se répéter de la même manière sur une même fonctionnalité. Les mêmes éléments doivent se retrouver au même endroit d'une page à l'autre, garder les mêmes appellations, avoir une cohérence entre les interactions. L'utilisateur reproduit et applique ce qu'il a compris ou appris d'une page à l'autre, les indications graphiques doivent l'aider en ce sens.

Signifiance des codes et dénominations

L'interface doit utiliser un vocabulaire et un langage graphique explicite et cohérent avec les actions à effectuer. Autrement dit, les éléments et les labels placés ont un rôle évident dans le parcours d'utilisation.

Compatibilité et accessibilité

L'interface, toujours centrée sur l'utilisateur, doit être conçue en fonction des caractéristiques de ce dernier. Nous devons faire en sorte que notre site respecte une certaine typographie, certaine règle en termes de mise en page et d'accessibilité au personne présentant des handicaps.

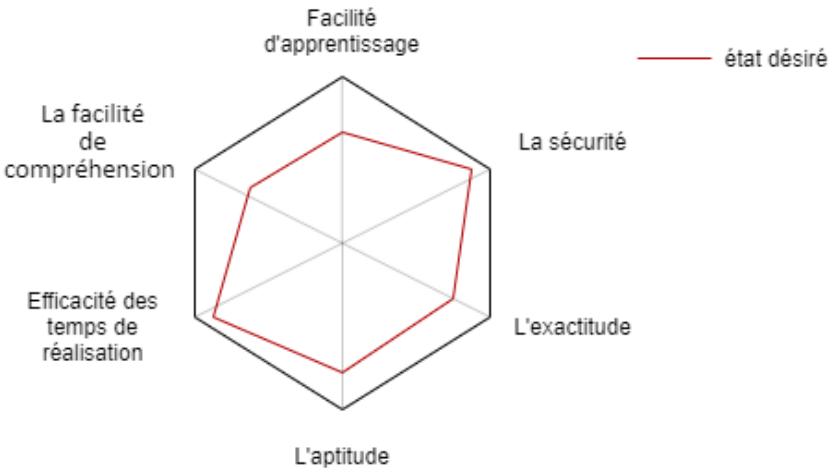
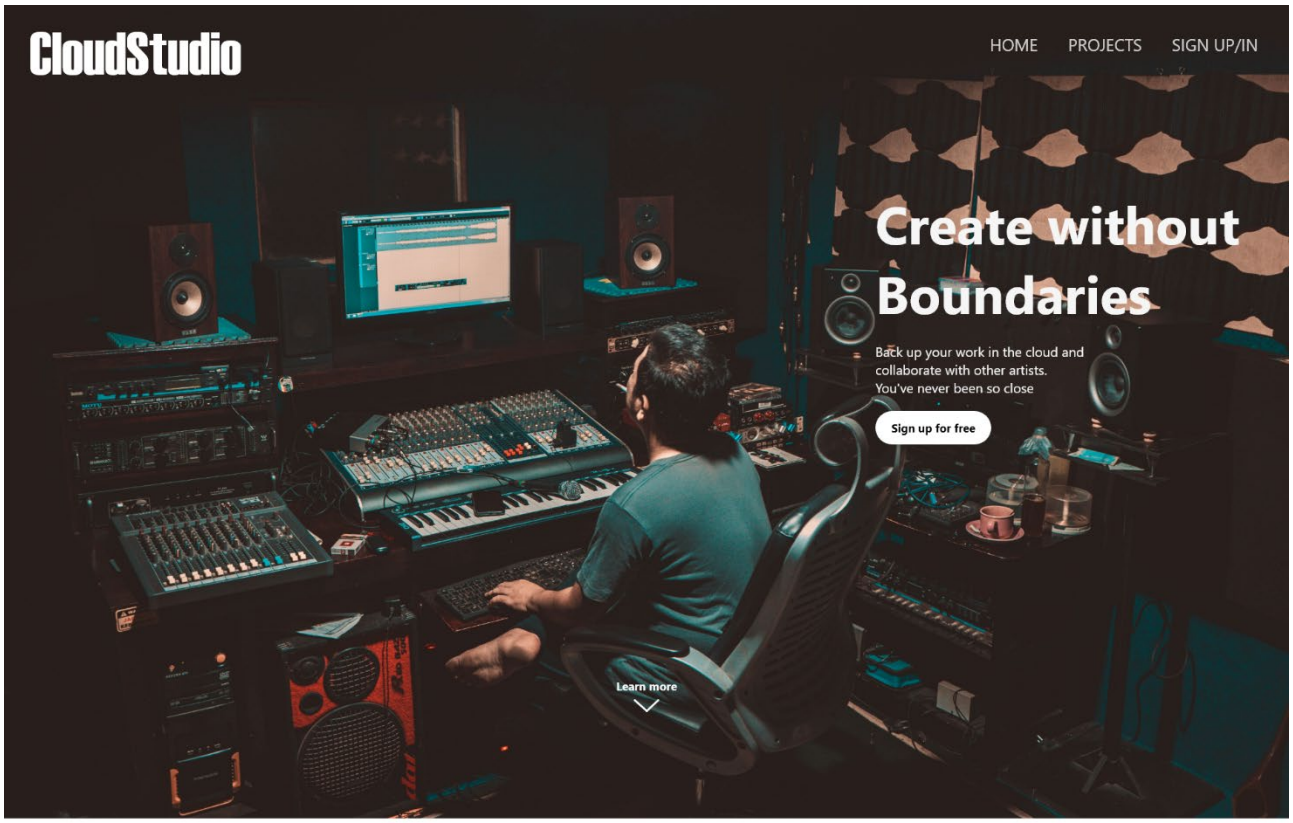


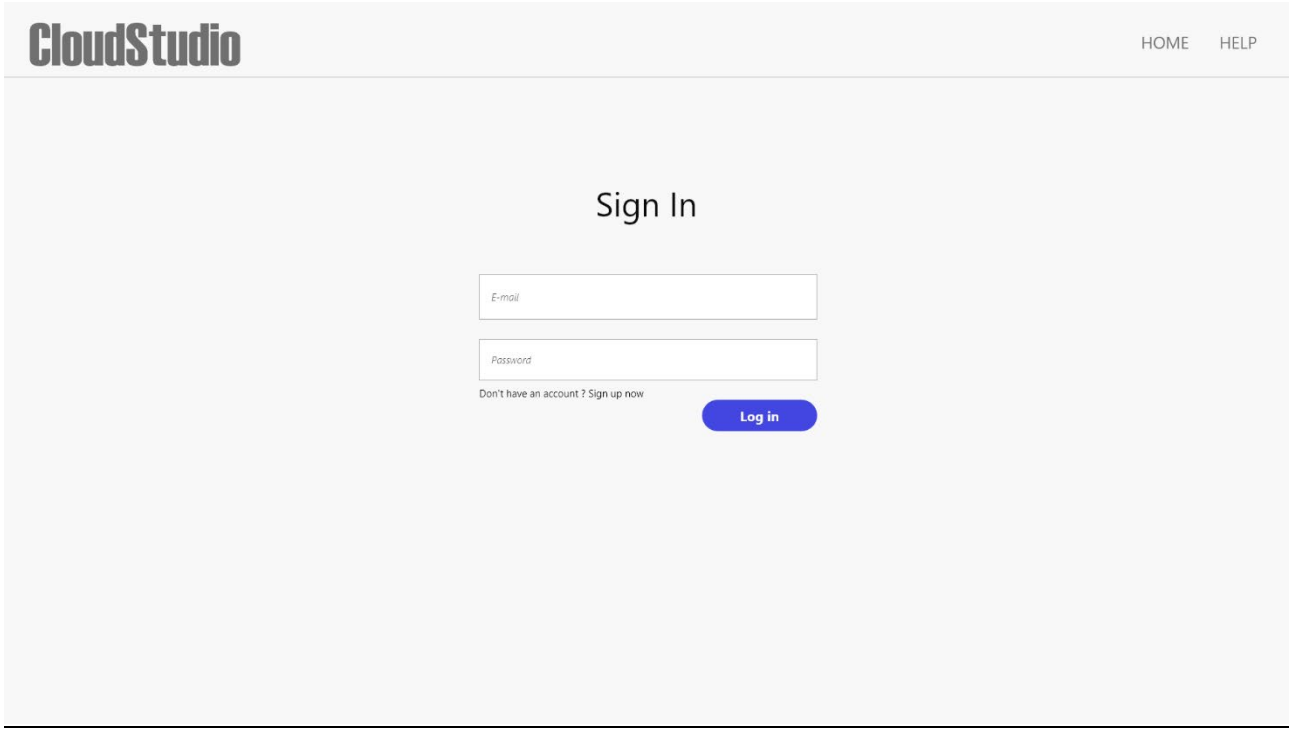
Diagramme d'indicateur de réussite

2.5 Maquettes du site

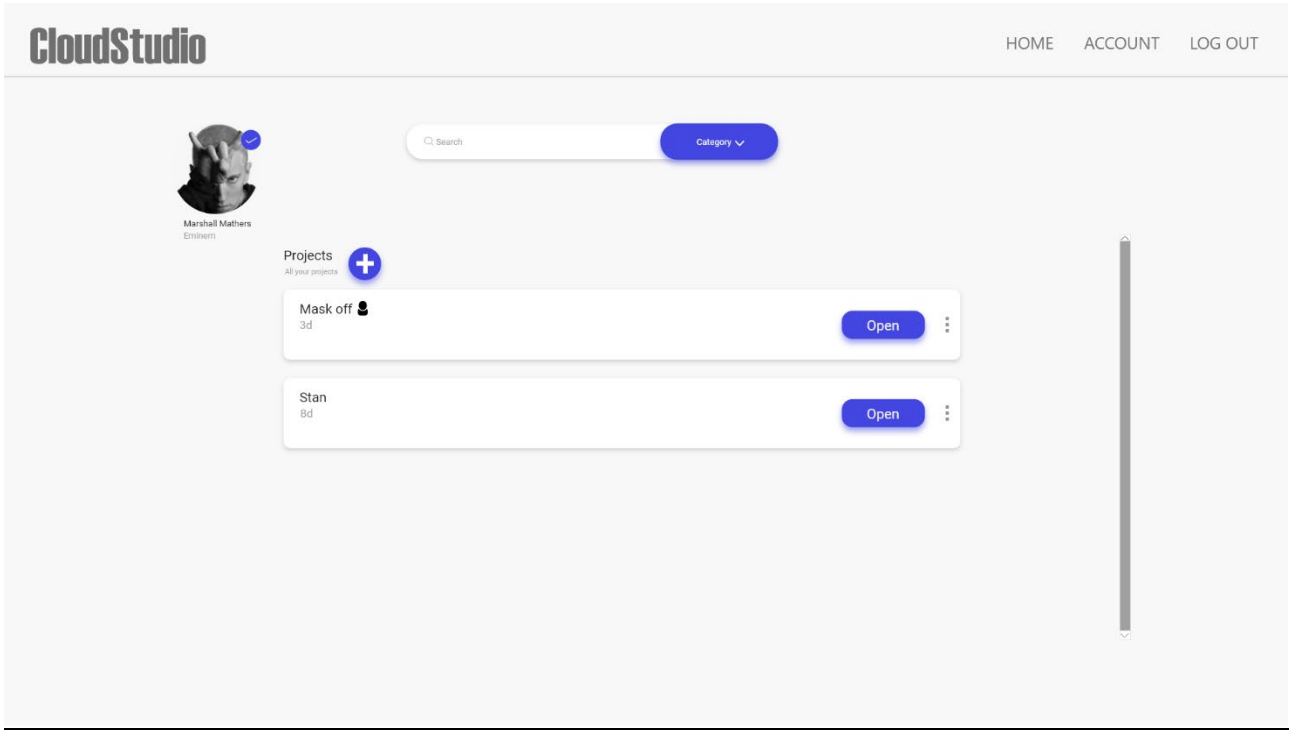
Voici tout d'abord les maquettes prototypes des différentes pages principales de notre futur site web (d'autres images sont disponibles en annexe) :



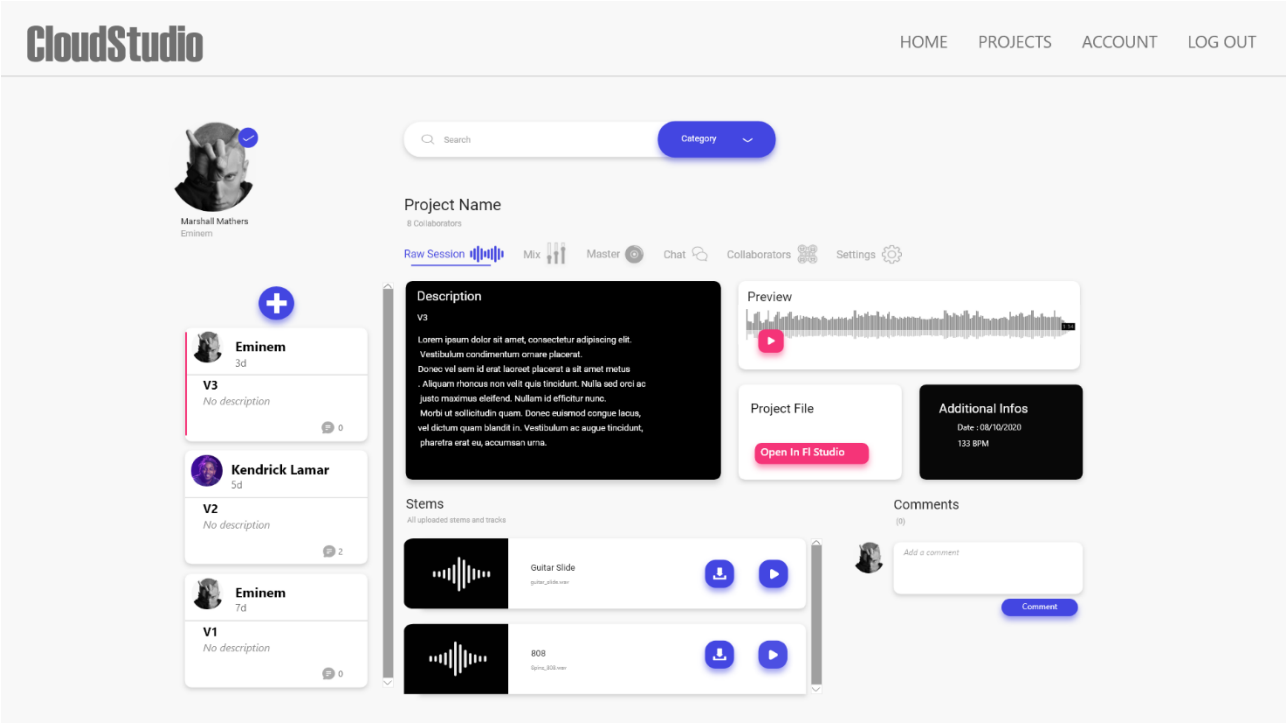
Page d'accueil



Page d'inscription



Page de sélection/création des projets



Page d'un projet

Sur notre maquette nous avons mis en avant le respect des critères ergonomiques, nous avons créer une interface épurée mais offrant de nombreuses possibilisées tout en limitant le risque d'erreur. Des sections bien distinctes correspondant à des labels distincts et cohérents. Chaque action réalisée par l'utilisateur peut être annulé en revenant en arrière. L'utilisateur est néanmoins libre de naviguer comme il veut sur l'ensemble du site, du moment que celui-ci est connecté à son compte, préalablement créer. De plus l'interface permet d'aller droit au but afin d'augmenter la productivité de l'utilisateur pour optimiser son workflow.

Les pages d'accueil et de connexion/inscription sont des pages très classiques. La page de sélection d'un projet permet d'accéder à un des projets existants de l'utilisateur ou d'en créer un nouveau en cliquant sur le bouton +. Il permet aussi d'effectuer une recherche parmi ses projets.
La page de gestion de projet est la plus complexe de notre site, elle permet de gérer tout son projet et d'accéder à la majorité des fonctionnalités proposé par notre plateforme, cela regroupe la création de version du projet, l'upload/download de fichier audio et de fichier de projet (.flp par exemple), l'ajout de collaborateur, etc.

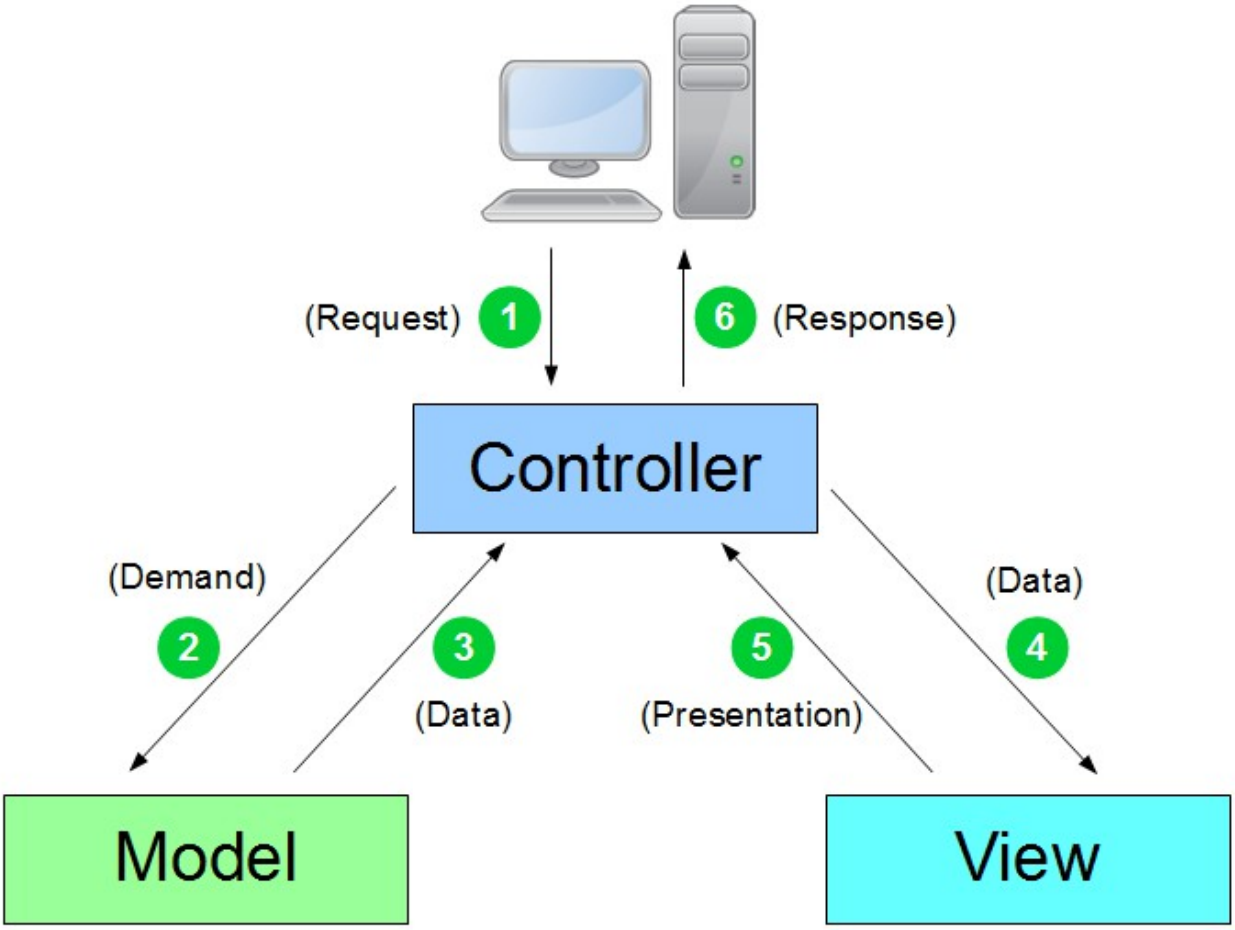
2.6 Spécifications

2.6.1 Développement du code

Pour le développement du code de notre site nous avons choisi de suivre l'architecture Modèle Vue Contrôleur (MVC)
Le pattern MVC nous permet de bien organiser notre code source. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

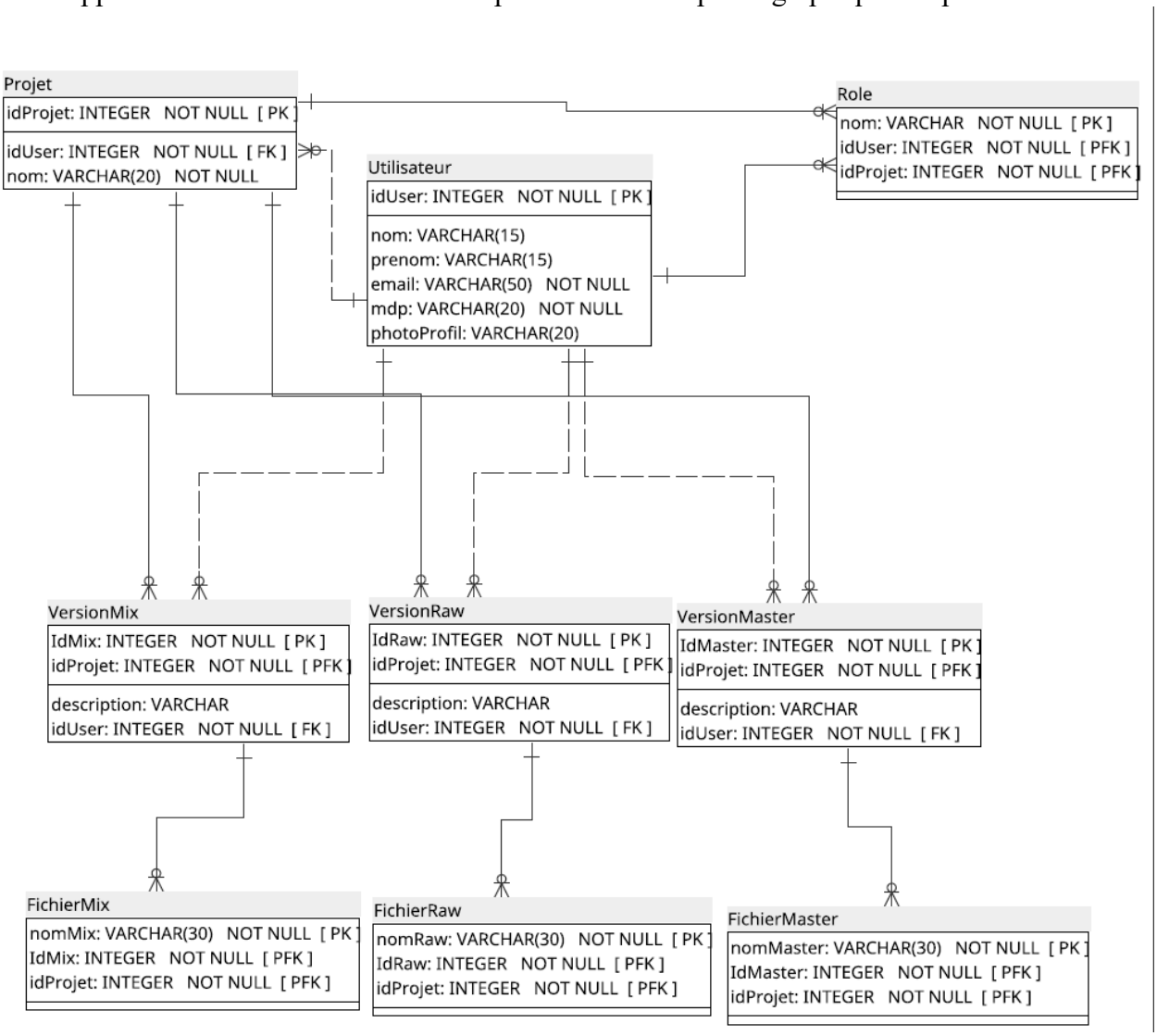
- **Modèle** : cette partie gère les *données* de notre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.
- **Vue** : cette partie se concentre sur l'*affichage*. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

Source : OpenClassRoom



2.6.2 Développement de la base de données

Le développement de notre base de données peut être résumé par le graphique ci-après :



Légende :
PK : clé primaire
FK : clé étrangère
PFK : clé primaire et étrangère
Trait plein : référence clé primaire et secondaire
Trait pointillé : référence clé secondaire mais pas primaire

Quand un utilisateur s'inscrit sur le site, on enregistre ce qu'il faut dans la table Utilisateur de la base de données. Ensuite, quand ce dernier crée un projet, il est enregistré dans la table correspondante et on enregistre son rôle de chef de projet dans la table rôle. Le projet est divisé en 3 parties (Raw, mix et master), qui correspondra à 3 onglets différents sur le site web. Chacune de ces parties contiendra une liste de versions, et chacune de ces versions contiendra une liste de fichiers. Toutes ces informations seront contenues dans les différentes tables Versions et Fichiers. Nous avons 3 tables Versions et Fichiers pour permettre de diviser les informations de la base de données en plusieurs tables, permettant de réduire le temps d'exécution de chaque recherche de versions et de fichiers dans la base de données.

3. Solutions

3.1 Technologies envisageables

Afin de réaliser ce site web, il y a un très grand nombre de solutions techniques envisageables, divisées en plusieurs types. Nous avons pour le moment des préférences sur chacun de ces types, mais nous pourrons très bien décider de changer de technologie si dans les quelques semaines à venir, nous trouvons une meilleure alternative pour résoudre nos difficultés de conception.

- Front end
Côté Client, il n'y a pas beaucoup de choix sur les langages à utiliser, qui sont HTML, CSS et JavaScript. Il y a cependant possibilité d'utiliser des CMS comme WordPress qui génèrent le code HTML, mais utiliser les langages directement permet de gagner en précision et en contrôle de l'apparence du site web pour le confort du client. Cependant il reste envisageable d'utiliser des Framework comme Bootstrap afin de faciliter le développement HTML et responsive du site.

- Backend
Côté serveur, il existe un très grand nombre de technologies différentes, PHP, nodeJS, python, JAVA, etc.
Il y a aussi les bases de données qui sont gérées côté backend, il existe donc plusieurs technologies : MySQL, PostgreSQL, SQLITE, etc.

- **Gestion/conception du projet** : Afin de gérer le projet, il existe des outils de contrôle de versions comme Git et GitLab, qui sont indispensable afin de permettre à toute l'équipe d'être à jour sur le projet et de pouvoir revenir en arrière à tout moment si nous devons annuler ou recommencer l'implémentation d'une fonctionnalité. De plus, si les données sont perdues ou si le serveur web plante, les données du site seront toujours en sécurité sur le dépôt distant.
Lors de la conception du projet, nous aurons également besoin de réaliser des diagrammes UML, nous utiliserons donc Visual Paradigm, car nous avons déjà été initiés à ce logiciel l'an dernier et ça nous évitera donc d'apprendre à utiliser un autre logiciel.

Il y a également la question de l'hébergement qui se pose. En effet, le site web devra être stocké sur un serveur, constamment allumé et relié à internet afin de pouvoir fournir à tout moment les pages web aux clients.
La première possibilité est donc d'aller sur des services d'hébergement. Cependant, afin d'avoir un bon serveur qui peut gérer correctement des fichiers assez gros et nombreux ainsi que supporter un grand nombre de personnes connectées, il faudrait se diriger vers des services d'hébergement payants, car ceux gratuits risquent de ne pas fournir un serveur d'assez bonne qualité.
Azure ou AWS sont deux solutions que nous envisageons afin d'hébergé notre site ainsi que notre base de données.
La deuxième possibilité, qui est donc celle vers laquelle nous nous dirigeons actuellement, est d'héberger notre site web nous-mêmes, sur un Raspberry PI à l'aide d'un serveur Apache de préférence, car c'est le plus utilisé.
En effet, ces nano ordinateurs consomment très peu d'énergie, ce qui leur permet de rester allumés 24h/24 facilement, et il est très facile de les contrôler à distance et d'échanger des fichiers avec eux de n'importe où, s'ils sont reliés à internet avec la fibre optique.

3.2 Finalisation des choix techniques

Front end

Pour le développement de la partie client nous avons donc choisi les langages HTML, CSS, JavaScript et le Framework Bootstrap. Nous avons choisi ces langages car ce sont des langages que nous maîtrisons tous, et ce sont des standards de l'industrie. De plus il y a beaucoup de documentation les concernant. Nous avons choisi l'outil Atom pour développer notre code HTML, CSS, PHP et Javascript.

Back end

Pour le développement de la partie serveur nous avons choisi PHP. PHP est le langage le plus utilisé aujourd'hui dans le développement web, il y a donc beaucoup de documentation et d'aide disponibles pour ce langage, sans oublier que nous l'apprenons tous dans nos cours en parallèle de ce projet, ce qui pourra nous permettre de nous entraider en cas de problème avec ce langage.
Pour les bases de données, nous avons choisi d'utiliser MySQL, car nous trouvons de nombreux tutoriels et cours en français sur MySQL, ce qui rendra l'utilisation de cette technologie plus simple pour nous que n'importe quelle autre technologie, de plus, cette technologie est gratuite, réputée et une des plus utilisées dans le monde pour les bases de données. De plus nous avons choisi Atom pour développer les fonctions et MariaDB pour la gestion des bases.

Gestion et conception

Pour la partie gestion nous utilisons GitLab qui est l'outil qui nous est imposé pour ce projet. Et pour la partie conception nous avons choisi Visual Paradigm car nous connaissons et savons tous utiliser ce logiciel ainsi que SQL Power Architect pour les diagrammes de Base de Donnée.

De plus nous avons opté pour l'utilisations d'un Raspberry Pi afin d'héberger notre site car c'est la solution que nous maîtrisons le mieux et c'est également la plus intéressante d'un point de vue financier.

3.3 Organisation du travail

Pour notre projet, nous utilisons un modèle proche du modèle agile. Premièrement, nous nous adaptons à tout moment. Une étude des fonctionnalités et des manières de réalisation a été faite au préalable pour savoir avec quels outils nous allons développer ce projet.
Cependant, si au cours de la réalisation du projet, nous trouvons qu'un autre outil ou une autre méthode est mieux ou plus simple, nous pouvons à tout moment changer notre manière de faire. Nous nous adaptons. Nous faisons des tâches les unes après les autres, pour permettre des rendus réguliers et faire des modifications en cours de développement si besoin.
Le projet est donc itératif, avec des itérations courtes. Une fois une tâche finie, nous faisons des tests pour vérifier que tout fonctionne bien sur cette partie du projet, bien que la plus grosse partie des tests se fera à la fin du projet.
Le projet est aussi centré utilisateur dans une certaine limite. En effet, les contacts avec les utilisateurs sont aujourd'hui assez réduits, mais tout de même existants. Nous prenons en compte les remarques des potentiels utilisateurs, pour améliorer le produit lors de sa conception et réalisation. Nous n'attendons pas la fin du projet pour voir qu'il faut faire des modifications, nous les faisons dès la prise de connaissance de l'information. Nous sommes donc réactifs aux avis des futurs utilisateurs.
Nous voulons faire un projet fonctionnel, mais simple au départ puis ajouter des fonctionnalités pour rendre le site plus complet et plus intéressant pour tous. Ce projet suit donc le modèle agile grâce à

notre adaptation, réactivité face aux problèmes et remarques des utilisateurs, avec des tâches qui permettent des livraisons fréquentes permettant de modifier le produit tout de suite si besoin grâce à des petites itérations régulières.

Pour ce projet, la répartition des tâches sera la suivante :
4 étudiants feront les pages Web en HTML et CSS (Alain, William, Luka et Antoine), mais ils devront aussi gérer le PHP pour communiquer avec la base de données. Les 2 autres élèves (Eden et Emerik) s'occuperont de la base de données et de l'hébergement. L'analyse et la conception en amont se font avec tous les élèves réunis, afin que tout le monde ait une vision et une compréhension globale du projet.

3.4 Suivi de l'avancement

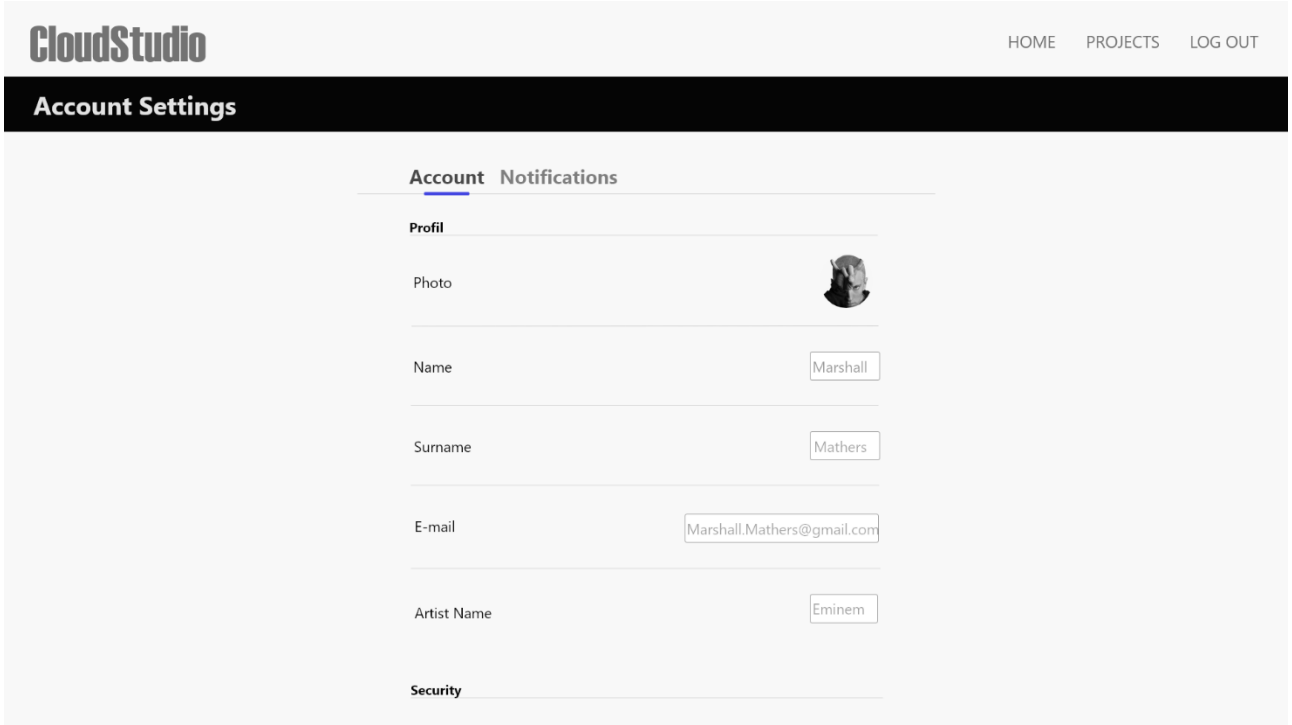
Au moment du rendu de ce second dossier voici où nous en sommes :

- Nous avons terminé le développement non fonctionnel, cela concerne la maquette notamment.
- Nous avons bien avancé la création et la mise en place de la base de données.
- Nous avons terminé la majorité des pages qui traiteront très peu d'informations et seront quasiment statique en HTML CSS et PHP. Cela regroupe la page d'accueil, la page de connexion et de création de compte notamment.
- Nous avons terminé la fonctionnalité permettant de créer un compte utilisateur et de se connecter à son compte.

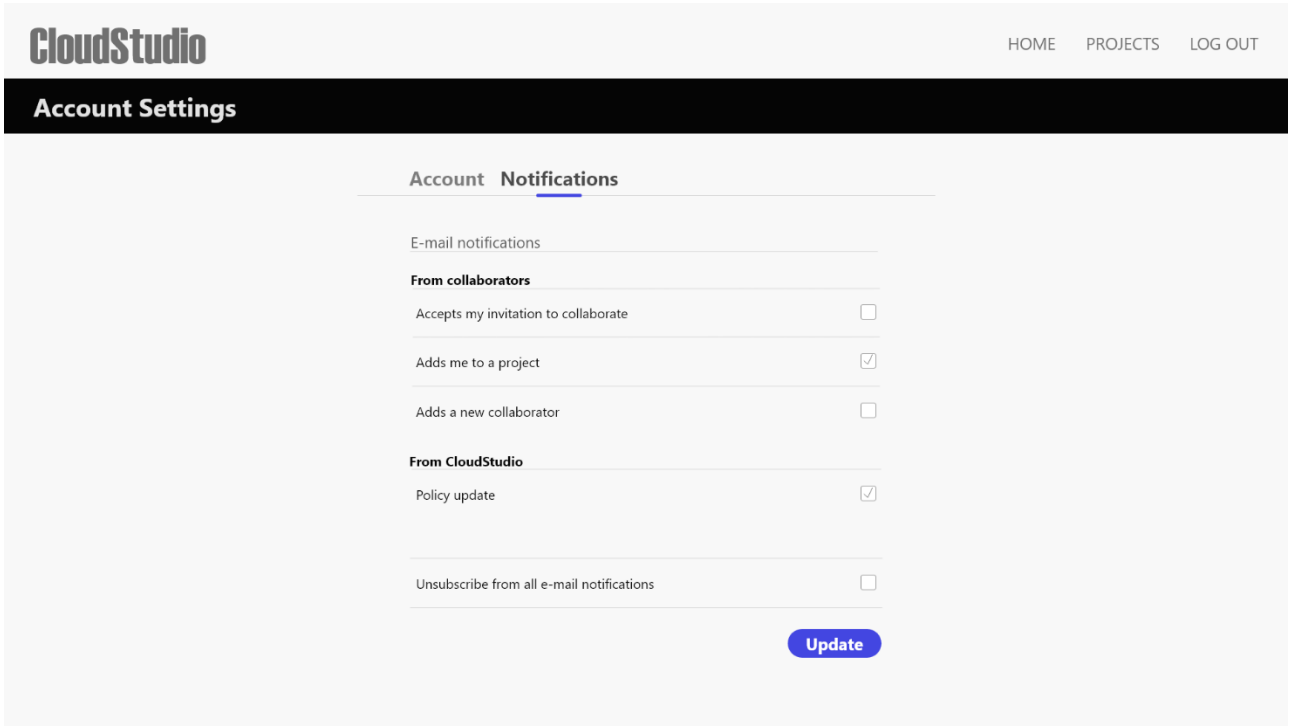
Glossaire

- **Développement front-end** : conception de l'interface graphique utilisateur (ce que le client voit)
- **Développement back end** : code exécuté par le serveur pour générer la page web (ce que le client ne voit pas)
- **Raspberry** : nano ordinateur de la taille d'une carte de crédit, très peu puissant, mais très peu cher et consommant très peu d'énergie, très utile pour faire office de serveur.
- **Mix/Mixage** : Le mixage audio est l'opération technique et artistique par laquelle, dans les domaines de la musique, du cinéma, du jeu vidéo, de la télévision et de la radio, un certain nombre de sources audio sont mélangées afin de parvenir à un équilibre cohérent, en intervenant sur le niveau, l'égalisation, la dynamique et la spatialisation.
- **Mastering** : Le Mastering est le processus consistant à transférer un ensemble d'enregistrements pour en faire un programme sur un support physique ou un fichier informatique, lequel servira à une fabrication en série ou à la diffusion. Son but premier est de rendre homogène cet ensemble. Pour l'audio, l'approche diffère suivant que l'album est un original ou une compilation de différentes œuvres originales.
- **Framework** : En programmation informatique, un Framework désigne un ensemble cohérent de composants logiciels structurels, qui sert à créer les fondations ainsi que les grandes lignes de tout ou d'une partie d'un logiciel.

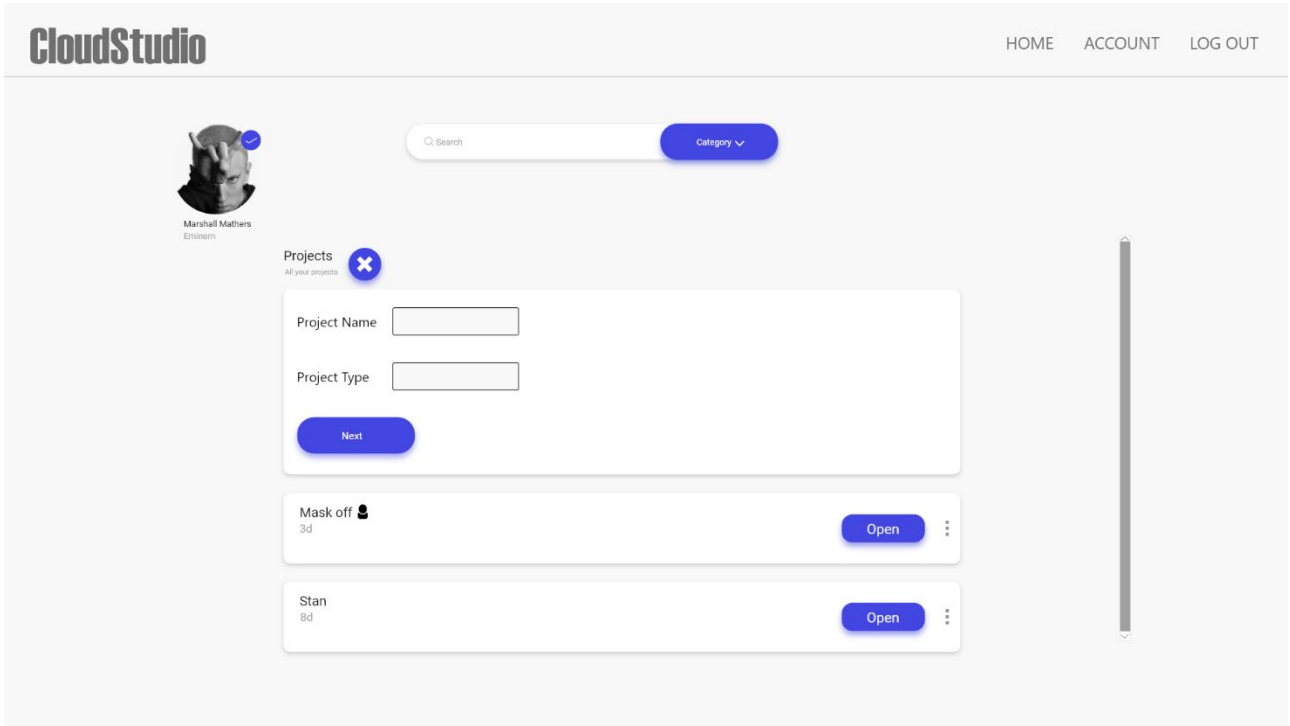
Annexe



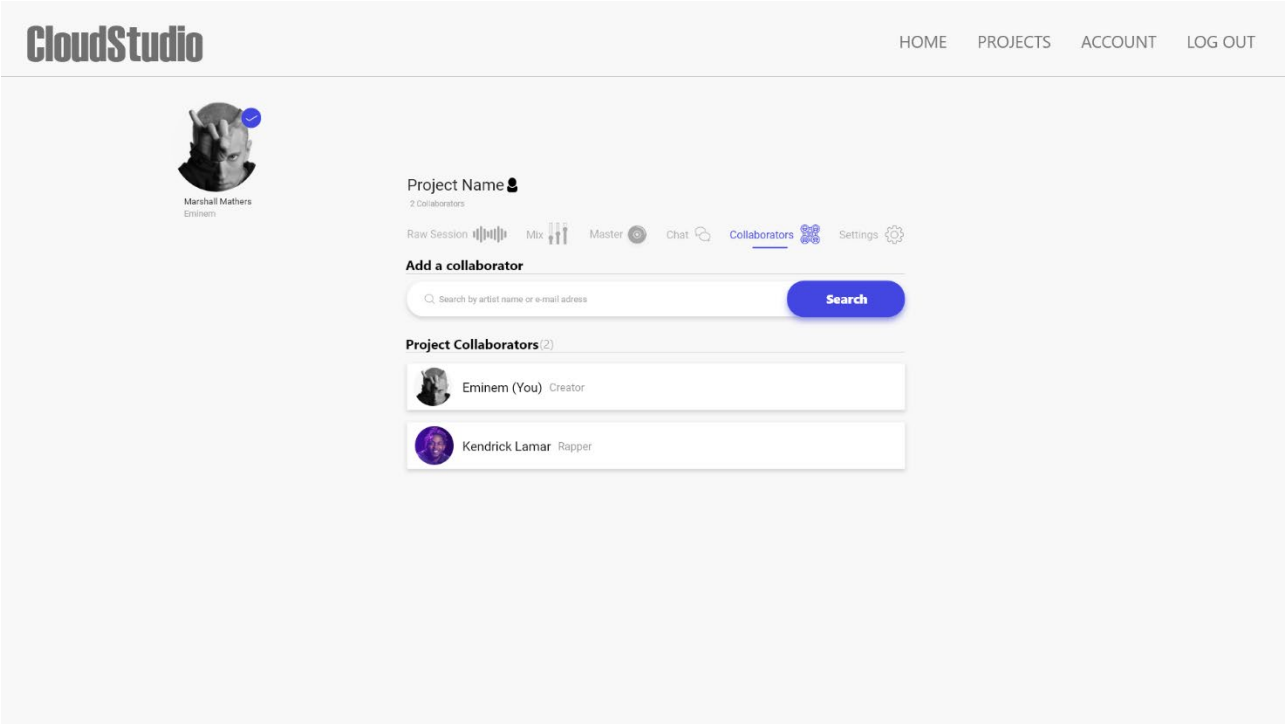
Page 1 Compte Utilisateur Paramètre 1



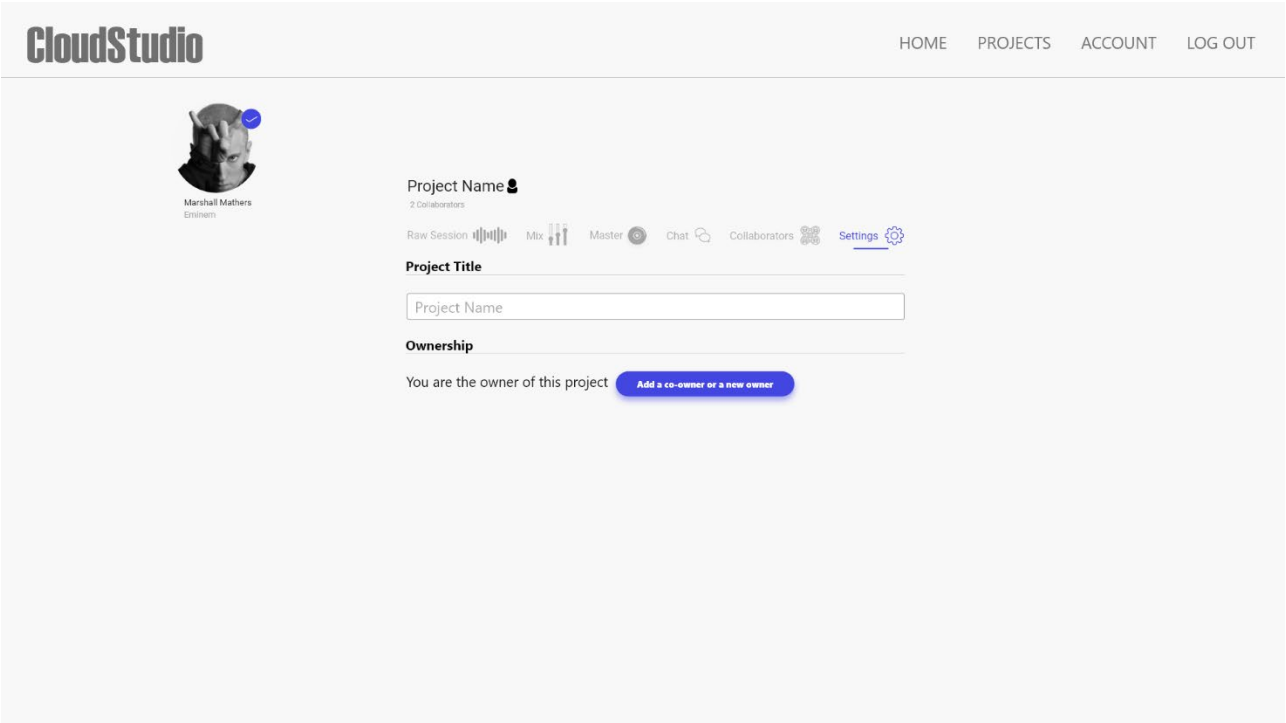
Page 2 Compte Utilisateur Paramètre 2



Page 3 Création d'un projet



Page 4 Ajout de collaborateur dans un projet



Page 5 Paramètre d'un projet