

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

Groupe 5
Blu-Ray Projekt
C1

CloudStudio

Dossier 3e itération - Projet Semestre 3

Table des matières

| | |
|---|----|
| 1. Cadrage du projet..... | 2 |
| 1.1 Description du projet..... | 2 |
| 1.2 Objectifs du projet..... | 2 |
| 1.3 Contexte du Projet..... | 3 |
| 1.4 Parties prenantes..... | 6 |
| 1.5 Contraintes..... | 6 |
| 1.6 Risques | 7 |
| 1.7 Suivi des contraintes et risques | 7 |
| 2. Expression du besoin | 7 |
| 2.1 Besoins fonctionnels et non fonctionnels..... | 7 |
| 2.2 Hiérarchisation des besoins | 9 |
| 2.3 Critères qualité | 10 |
| 2.4 Critères ergonomiques..... | 11 |
| 2.5 Conception de l'interface | 12 |
| 2.6 Spécifications | 15 |
| 2.6.1 Développement du code | 15 |
| 2.6.2 Développement de la base de données | 17 |
| 3. Solutions | 20 |
| 3.1 Technologies envisageables | 20 |
| 3.2 Finalisation des choix techniques..... | 22 |
| 3.3 Organisation du travail | 22 |
| 3.4 Suivi de l'avancement | 23 |
| 3.5 Tests..... | 24 |
| 3.6 Et après ? | 24 |
| Glossaire..... | 25 |
| Annexe | 25 |

1. Cadrage du projet

1.1 Description du projet

Le projet vise à aider des musiciens à créer leur musique en groupe à distance. La création d'un morceau implique plusieurs personnes, des musiciens, des producteurs, des ingénieurs, avec des tâches différentes, mais chacune dépendante d'une ou plusieurs autres tâches. Pour travailler correctement, les créateurs travaillent ensemble, dans la même pièce. Lors du confinement d'avril 2020, 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 fichiers 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.

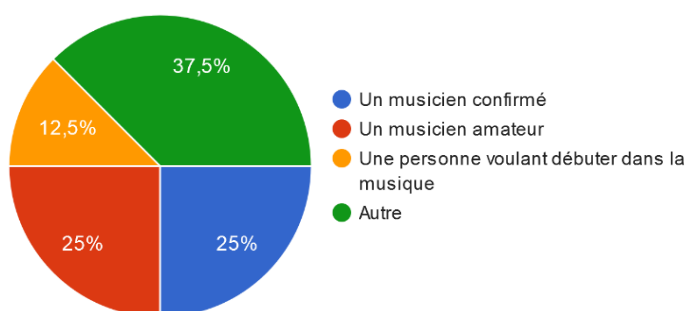
⇒ Nous sommes un groupe de 6 étudiants, et nous connaissons déjà suffisamment les langages de programmation nécessaire pour commencer le projet, voir même bien l'avancer sans énormément de difficultés au départ, il y aura néanmoins un certain challenge, car nous avons beaucoup de choses à découvrir,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

⇒ 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.

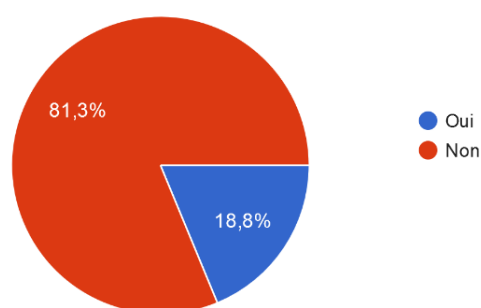
1.3 Contexte du Projet

Analyse du terrain

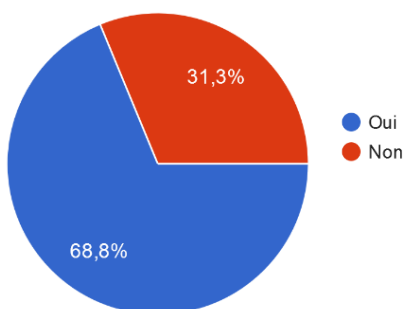
Pour mieux comprendre les besoins des musiciens, nous avons réalisé une analyse de terrain, notre analyse de terrain, nous avons procédé de deux manières différentes. La première chose que nous avons faite a été de créer un Google Form (un questionnaire) pour interroger les utilisateurs sur l'intérêt que ceux-ci pourraient porter au projet ainsi que sur leurs habitudes de travail en ce qui concerne la production musicale. Nous avons obtenu 87 réponses actuellement. Voici quelques-unes des réponses que nous avons récoltées.



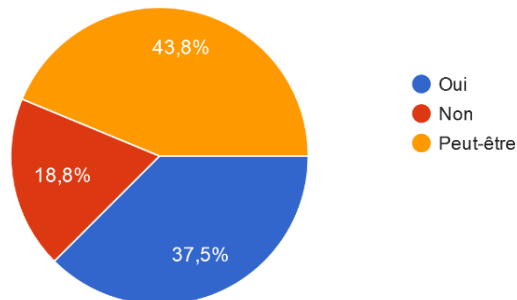
Graphique 1 Profil utilisateurs : Qui êtes-vous ?



Graphique 2 Utilisez-vous déjà un outil pour partager vos fichiers musicaux entre collègues ?



Graphique 3 En théorie seriez-vous intéressé par notre projet ?



Graphique 4 Dans l'hypothèse où le projet serait terminé et entièrement fonctionnel, seriez-vous prêt à l'utiliser ?

Le *Graphique 1*, qui répond à la question Qui êtes-vous ? nous permet de savoir à quel public nous nous adressons et donc de pouvoir adapter notre plateforme selon les besoins des utilisateurs. Les *Graphiques 2 et 3*, nous montre qu'actuellement le manque d'un outil efficace pour travailler en groupe à distance. Le *Graphique 4* nous montre qu'en considérant le nombre de musicien (*Graphique 1*) ayant répondu au sondage un nombre important de personne serait prête à utiliser la plateforme.

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 20 personnes représentant différents profils d'utilisateurs potentiels :

- Des musiciens confirmés (6 personnes)
- Des musiciens amateurs (5 personnes)
- Des personnes ne pratiquant pas d'activité liée à la musique (8 personnes)
- Un musicien confirmé, développeur sénior de métier

Ces interviews nous ont permis de confirmer les tendances observées grâce au sondage google Form, mais nous ont surtout permis de préciser en profondeur les fonctionnalités nécessaires à la plateforme. Nous avons donc pu déterminer plus précisément l'ordre d'implémentation et l'importance des fonctionnalités. Ces interviews nous ont aussi permis de trouver de nouvelles idées de fonctionnalités qui pourraient être implémentées dans le futur, grâce aux idées données par les personnes interviewées.

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 ou 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 apportent 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 fichiers 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 travail 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 fichiers et la gestion de projet, mais consiste plus à une grosse banque de sons libres 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. Un fichier 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'autres emplacements choisis 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 crée une nouvelle version sur la plateforme en ligne automatiquement. Donc 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 différences 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 collaboration 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 prévues pour le stockage en ligne de fichiers, mais aussi le partage de documents, mais qui ne sont pas prévues 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 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.

| Fonctionnalités | Solution(s) Existante(s) |
|---|--------------------------------------|
| Transfert de fichier et stockage de fichier | Clé Usb, Dropbox, Google Drive, etc. |
| Gestion de projet (musical) | Splice Studio, Git |
| Chat et messagerie | Reddit |

Tableau 1 Résumé des solutions existantes

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.

Le projet CloudStudio est dirigé par Eden Porte qui a été attribué Chef de projet, chargé de la prise de décision au sein de notre équipe son rôle a été de nous partager dans nos différents postes afin que chacun fournisse un travail plus efficace, tout au long du projet il organise le projet en nous attribuant une nouvelle tâche à effectuer en priorité.

Durant ce projet la maîtrise d'œuvre a un éventail de compétences qui doit englober plusieurs domaines, tout d'abord elle doit faire un site web cohérent et moderne, de plus il doit être facile à utiliser, ce qui fait qu'il doit avoir un design "responsive", mais également adaptée au public visé, nous ajoutons à cela la compétence de création et gestion de base de données, que nous devons mettre en place sur notre plateforme, et ce de manière sécurisée. Pour cela nous avons une équipe de 6 développeurs, Eden Porte, William Berruyer, Antoine Gros, Luka Languier, Emerik Measson, Alain Baptista.

Notre projet est suivi de bout en bout par deux professeurs référents présents pour nous conseiller et suivre notre avancée. Des rendus sont également effectués à chaque fin d'itération lors d'un oral devant deux autres professeurs qui nous évaluent.

1.5 Contraintes

La première contrainte est celle du nombre de fichiers 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 la plateforme. 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 versions d'un projet qu'un utilisateur pourra créer ainsi que le nombre de projets que celui-ci pourra créer.

La seconde 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 téléversées par l'utilisateur ne soient pas correctement sauvegardées. L'utilisateur risque donc de ne pas retrouver ses fichiers.

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 autre façon, 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 imparti. Il y aurait alors une réduction de la couverture fonctionnelle.

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, le risque majeur est de ne pas pouvoir finir l'ensemble des fonctionnalités que nous aurions souhaité implémenter à temps. De plus nous faisons face à un risque sanitaire avéré qui nous a forcé à travailler à distance et qui a donc nécessité que nous mettions en place des outils de travail collaboratif à distance tel que discord.

Les contraintes de stockages et de débit (techniques) ne nous ont pas posé particulièrement problème pour le moment.

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 postproduction.
- 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 Masterisée 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.

Ces notions sont définies plus en détail dans l'annexe.

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 (dont l'extension de fichier dépendra du logiciel avec lequel l'utilisateur travaille) 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 échangés 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 travaille afin que les potentiels autres collaborateurs qui rejoindraient le projet en cours de route en soit informés et pour qu'une vérification 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 ne choisir aucune version précédente. Cette fonctionnalité permet aux autres collaborateurs d'être informés sur quelle version est basée 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 téléchargent 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 projet, 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.

Du côté des besoins non fonctionnels, il faut tout d'abord que la plateforme soit ergonomique pour qu'elle soit utilisable facilement par tout type d'utilisateur, le site doit également être responsive pour s'adapter à tout type d'appareil. Les bases de données doivent être sécurisées, ainsi que leurs accès sur le site pour que seul l'utilisateur à qui appartient les fichiers puissent y accéder. Le site doit se mettre à jour dès qu'une action est effectuée, en particulier lors de la téléversions de fichier.

La plateforme doit toujours être disponible via l'url du site, ce qui implique que le Raspberry doit être allumé en permanence. Le site doit aussi disposer d'une sauvegarde (backup) des données des utilisateurs au cas où une panne endommagerait les données.

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 | Avoir accès aux différentes sections de son projet (Raw Session, Mix, Master) |
| 2 | Import / Export les fichiers et pistes nécessaires à la conduite du projet |
| 2 | Créer / Supprimer une version du morceau |
| 2 | Rédiger une description pour chaque version du projet créé |
| 3 | Ajouter / Supprimer des collaborateurs à un morceau (Recherche par mail) |
| 3 | Download et écouter la démo d'un morceau et les pistes dans une version du projet |
| 4 | Modifier son profil |
| 4 | Envoyer/recevoir des messages dans un chat principal |
| 4 | Créer des groupes de morceaux (albums) |
| 4 | Modifier les paramètres d'un morceau/album |
| 4 | Définir des rôles pour les collaborateurs d'un morceau |
| 4 | Tutoriel pour les nouveaux utilisateurs |
| 4 | Envoyer / Lire des commentaires pour chaque version d'un même morceau |

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

| BESOINS NON FONCTIONNELS | |
|---------------------------------|---|
| Priorité | Définition du besoin |
| 1 | Site ergonomique (responsive 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 | Facilité d'apprentissage |
| 1 | Facilité de compréhension |
| 1 | Sécurité |
| 1 | Exactitude |
| 1 | Aptitude |
| 1 | Site toujours disponible et avec une sauvegarde (backup) sur un disque dur en cas de panne (Fiabilité) |
| 2 | Code bien réparti et clair (avec des commentaires pour aider à la compréhension) afin de faciliter la testabilité et la maintenance du site |
| 2 | Service gratuit pour profiter à tous |

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 personnes puissent 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 aident 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.

Signifiante 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, certaines règles en termes de mise en page et d'accessibilité aux personnes présentant des handicaps.

La facilité d'apprentissage et la facilité de compréhension sont des facteurs importants, néanmoins la sécurité et l'efficacité des temps de réalisation sont essentielles à l'utilisation de la plateforme afin que les utilisateurs puissent s'en servir correctement. De plus la plateforme est destinée à des utilisateurs de tout niveau, il faut donc que la plateforme soit suffisamment compréhensible pour les utilisateurs peu expérimentés, mais il faut aussi qu'elle paraisse suffisamment professionnelle pour les utilisateurs plus expérimentés.

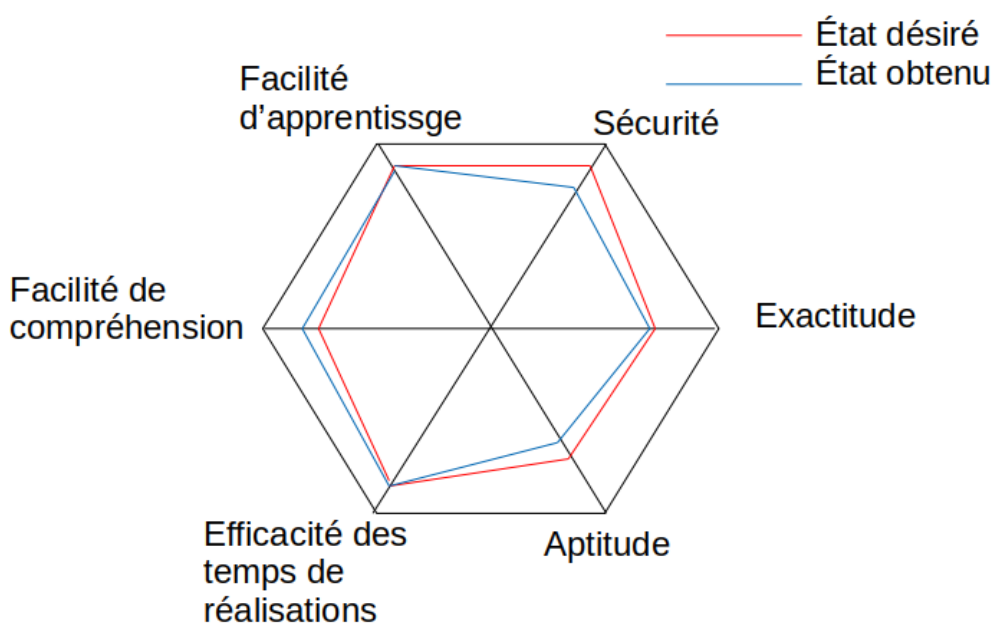


Diagramme d'indicateur de réussite

Diagramme 1 Diagramme d'indicateur de réussite

Pour les critères Facilité d'apprentissage, exactitude et efficacité des temps de réalisation nous estimons que nous avons atteints ce que nous nous étions fixés. Du côté de la sécurité nous n'avons pas pu atteindre les objectifs que nous nous étions fixés, néanmoins la sécurité de la plateforme reste acceptable. Certaines fonctionnalités n'ont pas pu être implémentées, faute de temps l'aptitude n'est donc pas au niveau que nous avons envisagé. Néanmoins toutes les fonctionnalités principales sont implémentées.

2.5 Conception de l'interface

Le schéma ci-après représente l'arborescence de la plateforme. La page d'accueil est le niveau le plus haut du site. La page projet ainsi que toutes les pages de ses niveaux inférieurs ne sont accessibles que si l'utilisateur est connecté à son compte. Un nouvel utilisateur peut se créer un compte sur la page d'accueil. Un utilisateur possédant un compte peut se connecter à son compte depuis la page d'accueil. Il peut ensuite accéder à ses projets ou à ses paramètres de compte. Depuis la page

paramètre de compte l'utilisateur peut modifier ses informations personnelles. Depuis la page Projets l'utilisateur peut créer de nouveau projet ou d'accéder à ses projets existants. Quand l'utilisateur accède à un de ses projets il a ses versions de projets.

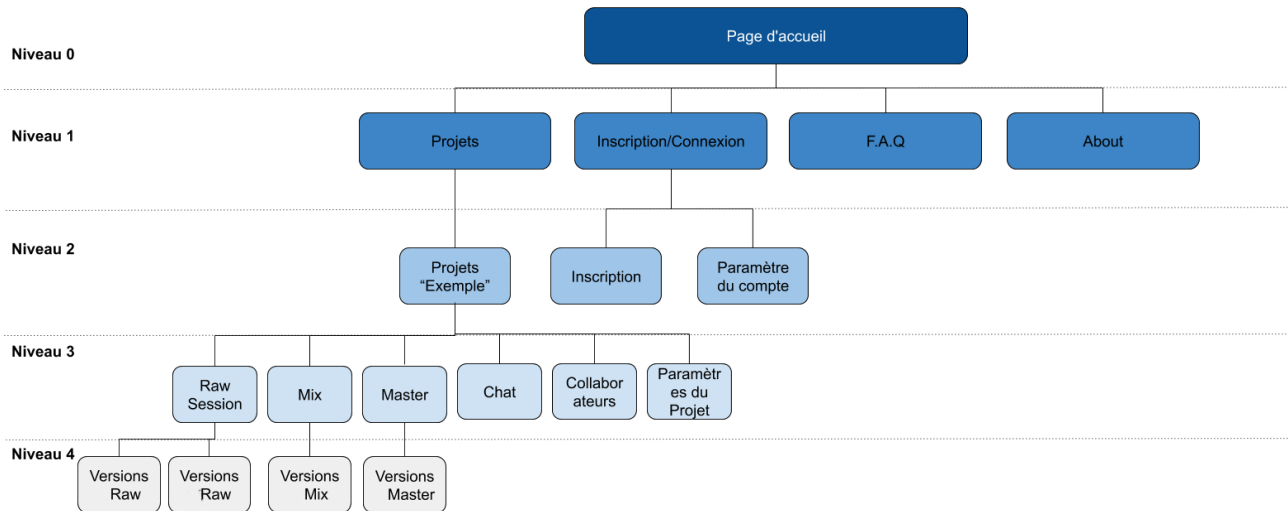
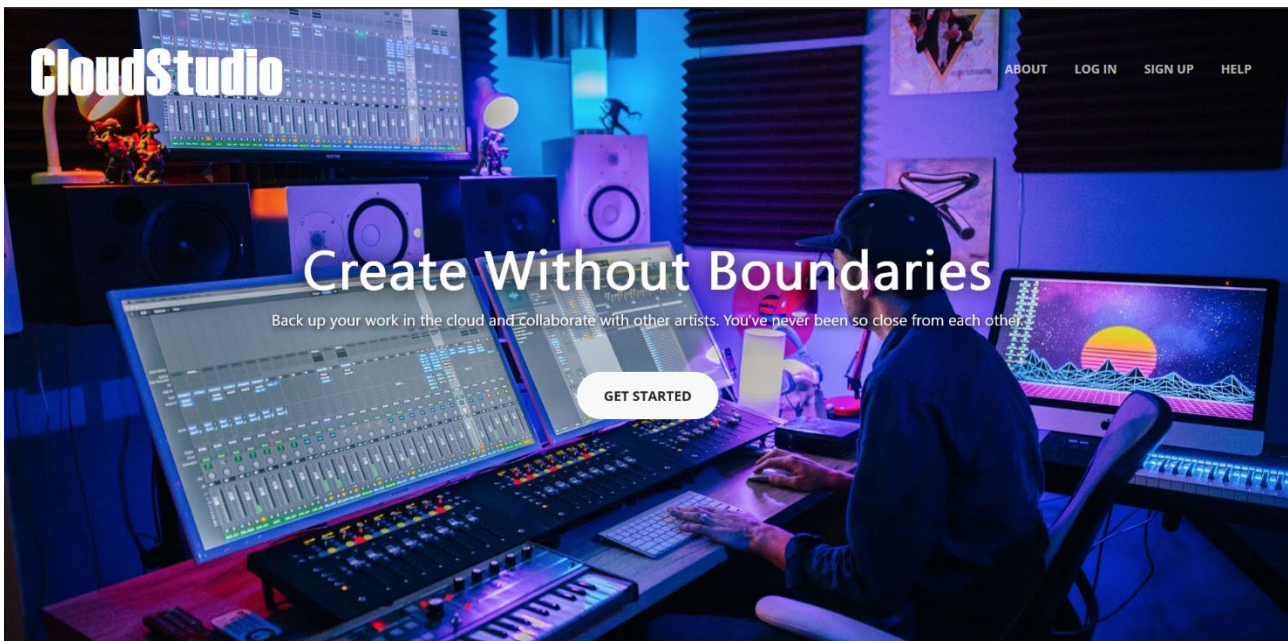
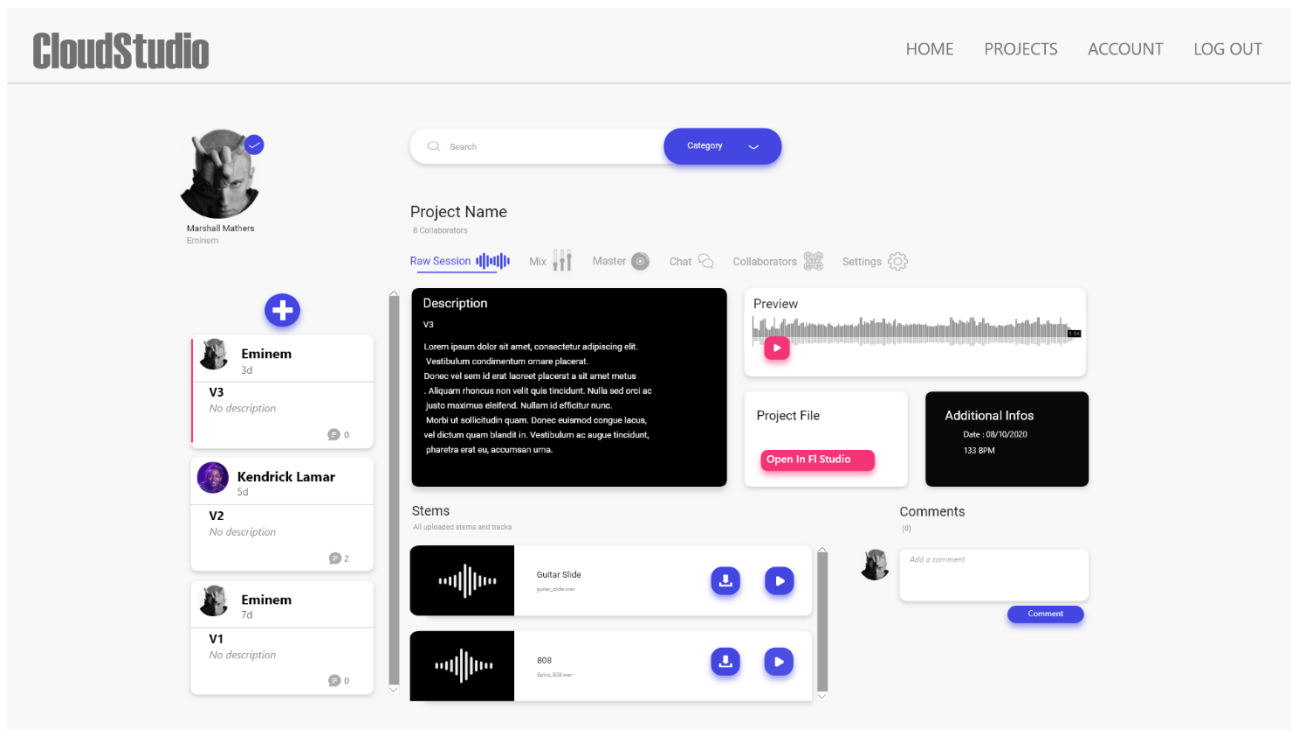


Schéma 2 Arborescence du site

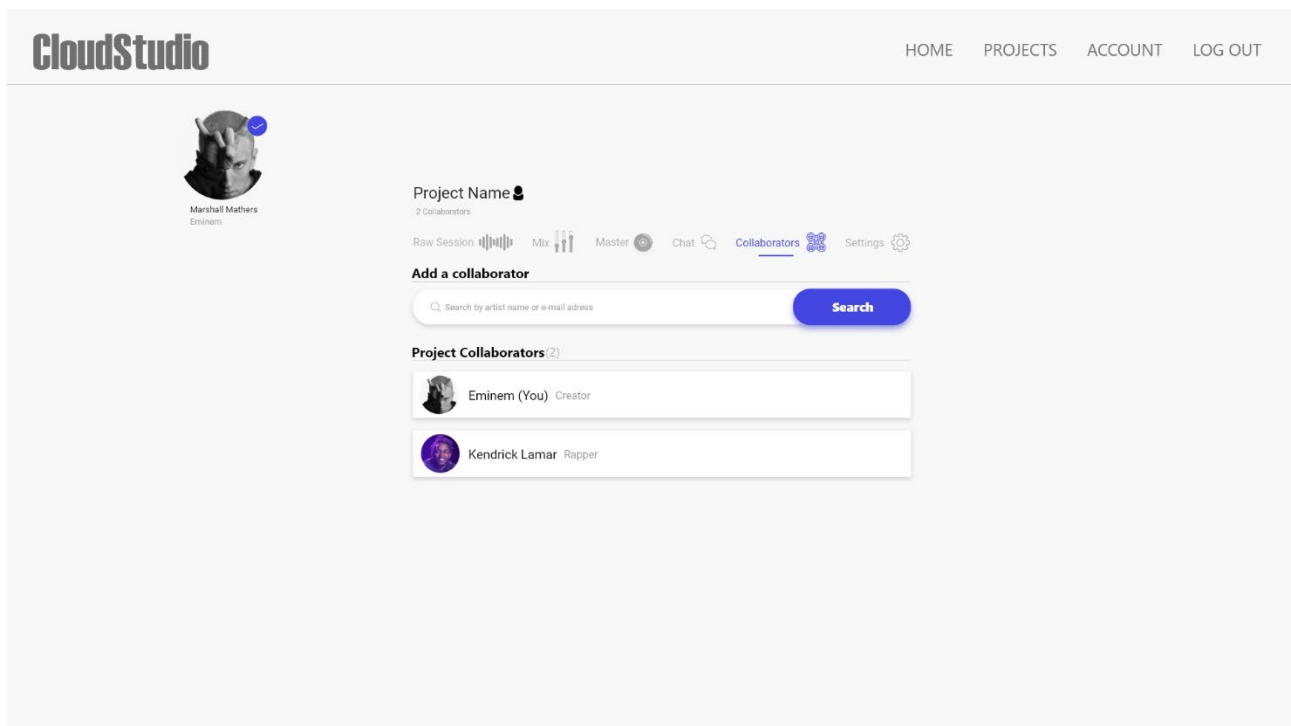
Voici les maquettes prototypes des différentes pages principales de notre plateforme (d'autres images sont disponibles en annexe) :



Page 1 Page d'accueil



Page 2 Page de gestion d'un morceau



Page 3 Page Ajout de collaborateur dans un projet

Sur notre maquette nous avons mis en avant le respect des critères ergonomiques (guidage, charge de travail, Contrôle explicite, Adaptabilité, Gestion des erreurs, Cohérence, Signifiante des codes, Compatibilité), nous avons créé une interface épurée, mais offrant de nombreuses possibilités 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 annulée en revenant en arrière (contrôle explicite). 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 créé. 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 demandant à l'utilisateur d'entrer les informations nécessaires pour se connecter ou s'inscrire, l'utilisateur doit obligatoirement accepter les termes et conditions pour se créer un compte. 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 +. Elle permet également de supprimer ses projets. Les termes utilisés sont clairs et permettent de guider l'utilisateur efficacement.

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 proposées par notre plateforme, cela regroupe la création de versions du projet, l'upload/download de fichier audio et de fichier de projet (.flp par exemple), l'ajout de collaborateur, etc. Lors de la création d'un projet une version est créée par défaut et ne peut pas être supprimée par l'utilisateur tant qu'il n'a qu'une version. Les boutons permettant d'effectuer les actions principales ont tous un thème commun ce qui permet à l'utilisateur de se repérer rapidement sur la page. L'interface de la page est épurée et construite avec des blocs, chaque bloc correspondant à un type d'action, ce qui permet également de rendre la page utilisable sur tous types de plateformes, même sur téléphone, grâce au côté responsive de la plateforme.

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).

Le schéma suivant illustre le modèle MVC (Modèle Vue Contrôleur) décrit précédemment.

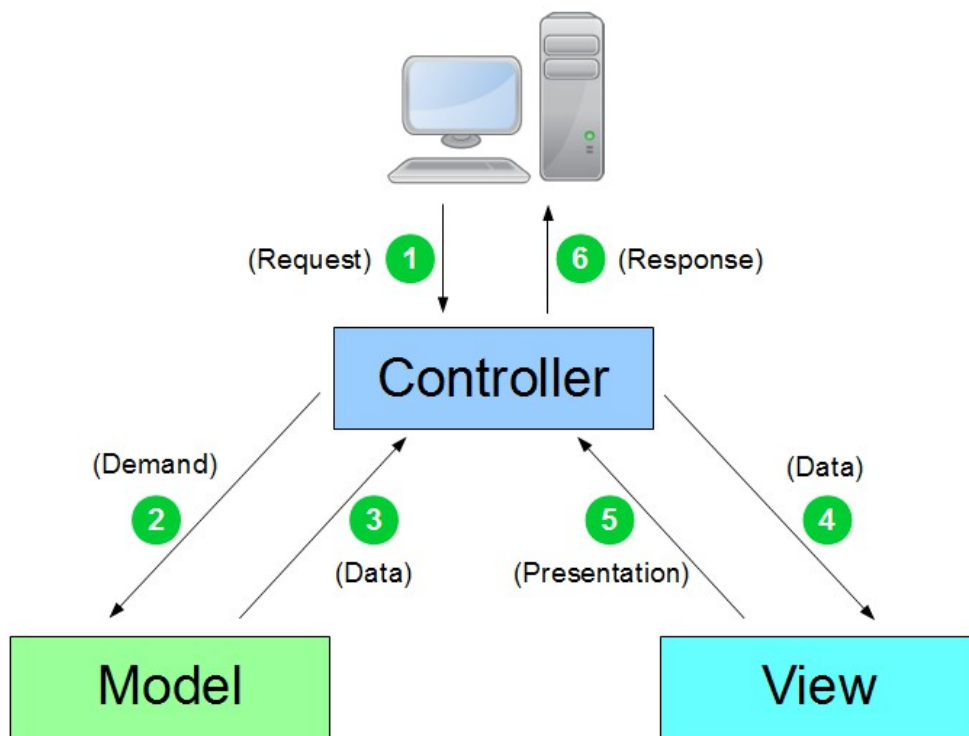
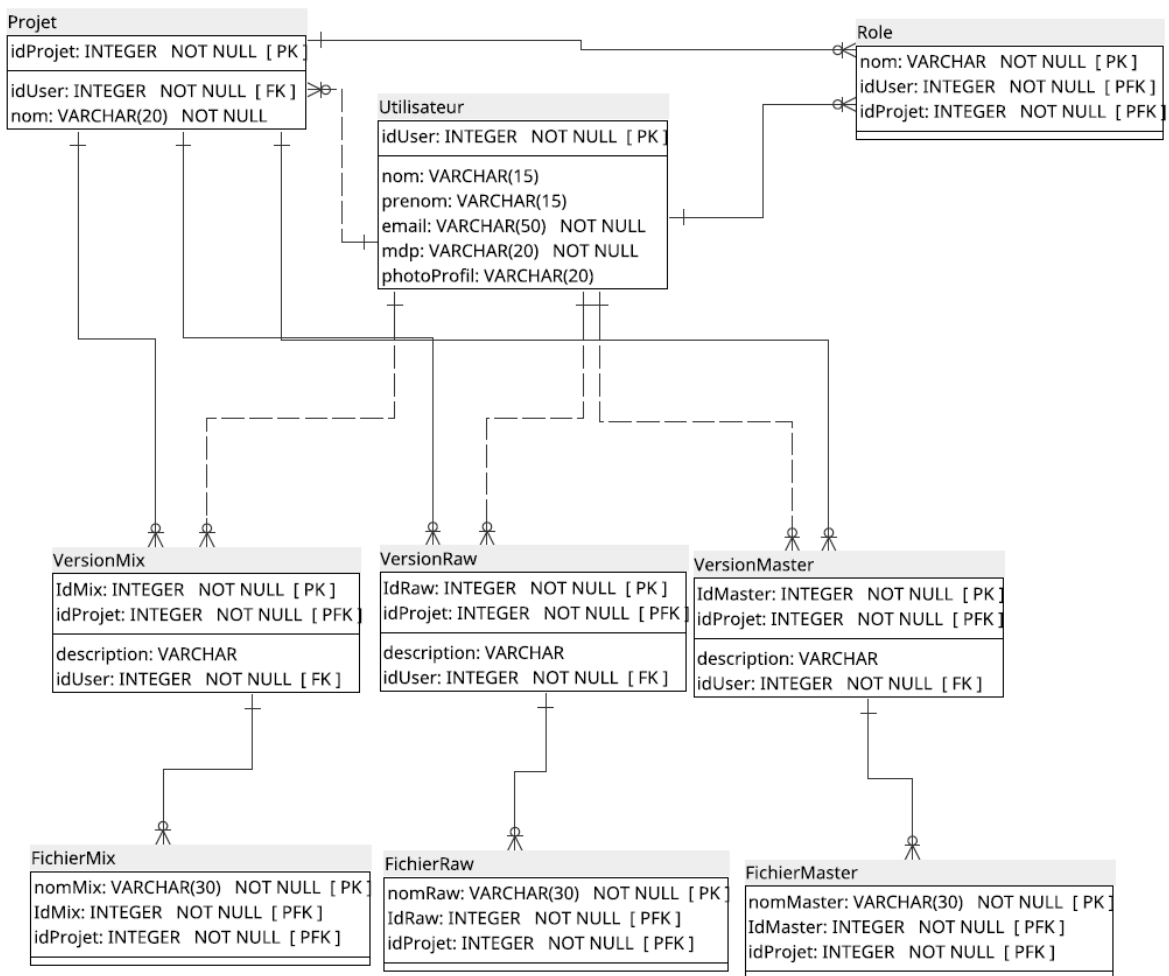


Schéma 1 Modèle Mvc (Source : OpenClassroom)

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

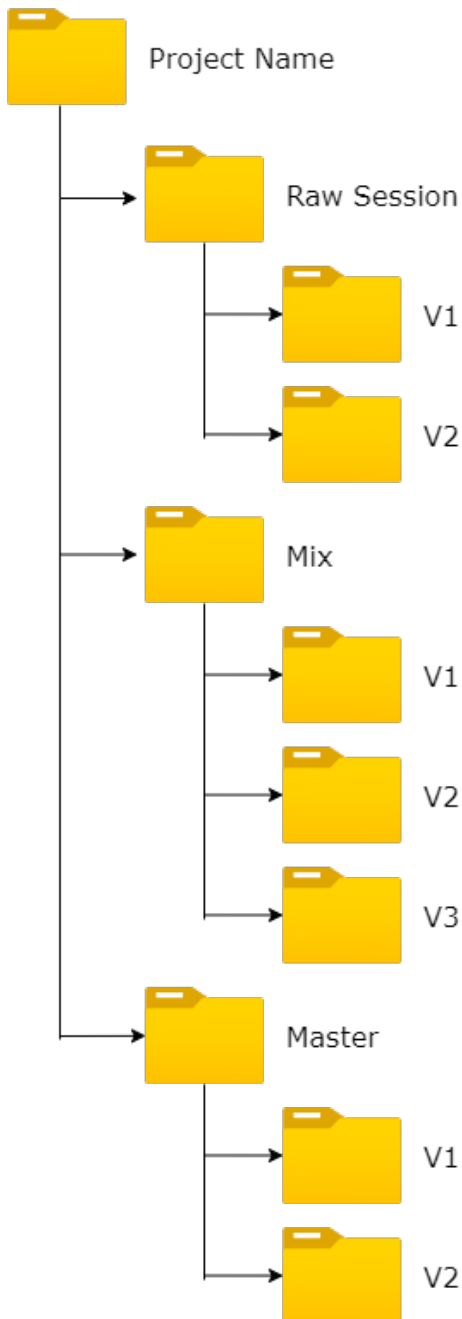
PFK : clé primaire et étrangère

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 correspond à 3 onglets différents sur le site web. Chacune de ces parties contient une liste de versions, et chacune de ces versions contient une liste de fichiers. Toutes ces informations sont 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.

Le stockage des versions sur le serveur se fera comme sur le graphique ci-après :



Structure de fichier / Structuration des fichiers sur le serveur

*Le nombre de versions dans chaque sous-catégorie (Raw Session, Mix, Master) dépendra du nombre de versions créées par l'utilisateur. Le graphique représente donc un cas parmi tant d'autres, les dossiers communs à tous les utilisateurs sont donc : Project Name (qui changera de nom selon le choix de l'utilisateur), Raw Session, Mix et Master.

Cette structuration permet d'accéder facilement aux fichiers pour les opérations de téléchargement sur la plateforme. Ainsi que de placer les fichiers téléversés par l'utilisateur facilement dans le serveur. De plus cette structuration se rapproche fortement de la structuration de fichier utilisée dans les projets professionnels de musique, notamment par les ingénieurs son.

Les diagrammes de séquences ci-dessous précisent le fonctionnement des fonctionnalités principales du site (les plus importantes).

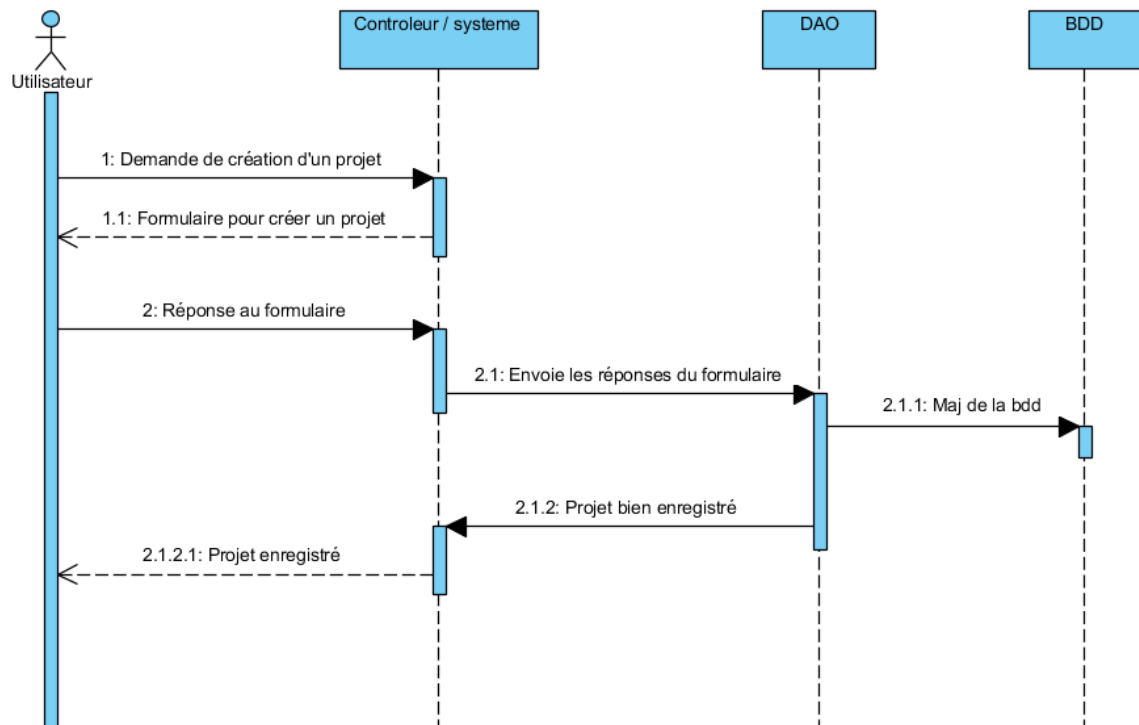


Diagramme de séquence 1 Création d'un projet

L'utilisateur est connecté, et se trouve sur la page où sont répertoriées les projets.

Il demande alors à l'aide d'un bouton, la création d'un nouveau projet. Un formulaire est envoyé à l'utilisateur et une fois le formulaire rempli, le DAO se charge de mettre à jour la base de données, et confirme la création du projet. L'utilisateur est alors redirigé vers la liste de ses projets.

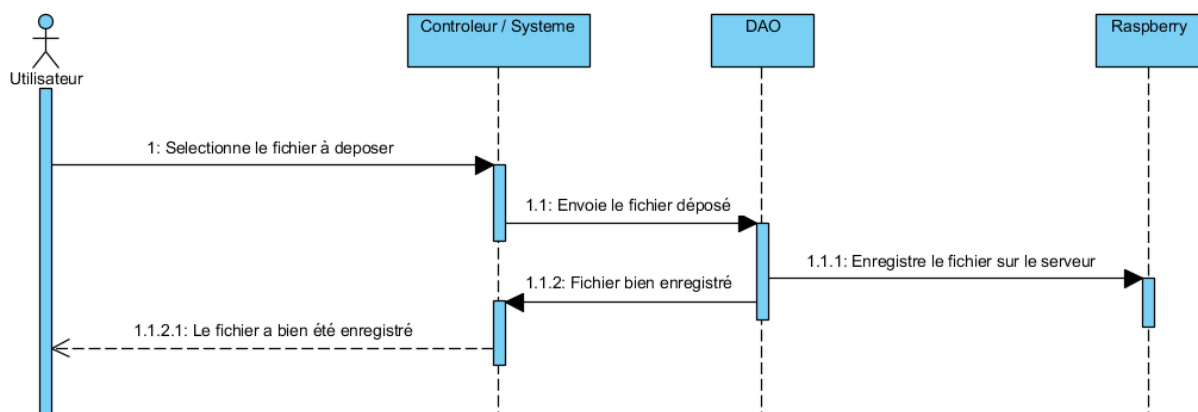


Diagramme de séquence 2 Dépôt d'un fichier

L'utilisateur se trouve sur la page de son projet, il souhaite déposer un fichier sur ce projet. Le fichier à déposer est sélectionné par l'utilisateur, puis le DAO enregistre le fichier sur le serveur (Raspberry)

et confirme que le fichier a été enregistré. L'utilisateur voit sa page de projet mit à jour, avec son fichier qu'il vient de déposer, dans une liste prévue à cet effet.

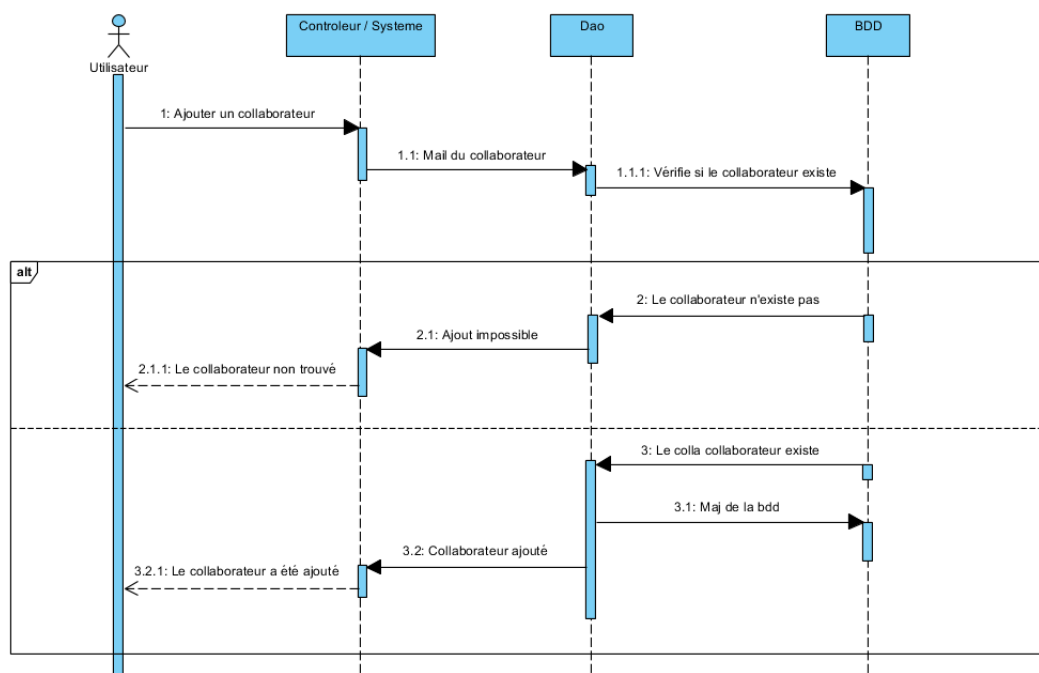


Diagramme de séquence 3 Ajout d'un collaborateur à un projet

L'utilisateur se trouve sur l'onglet collaborateur de son projet. Il écrit le mail de la personne qu'il veut ajouter, puis le DAO vérifie si le mail entré est enregistré sur la base de données (si le mail correspond à un compte déjà inscrit). Si aucun compte n'est trouvé, alors le collaborateur n'est pas ajouté, et l'utilisateur est prévenu de cette erreur. Dans le cas où le mail correspond, le DAO met à jour la base de données, et confirme que le collaborateur a été ajouté. L'utilisateur remarque que sa liste de collaborateur a été mise à jour.

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 pourrions 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 Frameworks comme Bootstrap afin de faciliter le développement 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 indispensables 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'héberger 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.

| Solutions | Avantages | Inconvénients |
|------------------|--|---|
| HTML/CSS | Précision/Contrôle total/flexibilité/Connaissance des langages/ Large ressource sur internet | Temps de mise en place |
| Bootstrap | Temps de mis en place/ Large ressource sur internet | Moins de contrôle |
| PHP | Connaissance du langage/ Large ressource sur internet | Syntaxe/Évolutivité |
| MySQL | Large ressource sur internet | |
| Postgres | Connaissance préalable | Moins de ressources que SQL en ligne |
| Visual Paradigm | Nombreuses possibilités/Connaissances du programme | |
| Raspberry Pi | Connaissance de l'outil/Facilité de mise en place | Espace limité/Débit de transfert limité/nécessite d'être branché 24h/24 |
| AWS ou Azure | Espace de stockage/ vitesse de transfert/backup/sécurité | Aucune connaissance de l'outil |

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. Nous utilisons le logiciel Bootstrap Studio qui nous permet d'accélérer grandement le temps de développement des éléments HTML de notre plateforme ainsi que leurs mises en forme CSS/Bootstrap.

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 avons eu des cours de PHP au cours de notre formation, 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.

Gestion et conception

Pour la partie gestion de versions, 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 Données.

De plus nous avons opté pour l'utilisation 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. Le Raspberry utilise un serveur apache.

| Front end | Back end | Gestion/Conception | Hébergement |
|---------------------------------|----------|--------------------|--------------|
| HTML | PHP | GitLab | Raspberry Pi |
| CSS | MySQL | Visual Paradigm | Apache |
| Bootstrap (Bootstrap Studio) | | Excel | |
| Javascript | | | |

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 soit fait 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 ont réalisé les pages Web en HTML et CSS (Alain, William, Luka et Antoine), 3 étudiants ont réalisé les fonctions PHP (Eden, Antoine, Emerik). 2 élèves (Eden et Emerik) ont réalisé la base de données et Eden s'est occupé de l'hébergement. L'analyse et la conception en amont se sont fait avec tous les élèves réunis, afin que tout le monde ait une vision et une compréhension globale du projet. La réalisation de ce dossier a été faite par l'ensemble des membres du groupe.

3.4 Suivi de l'avancement

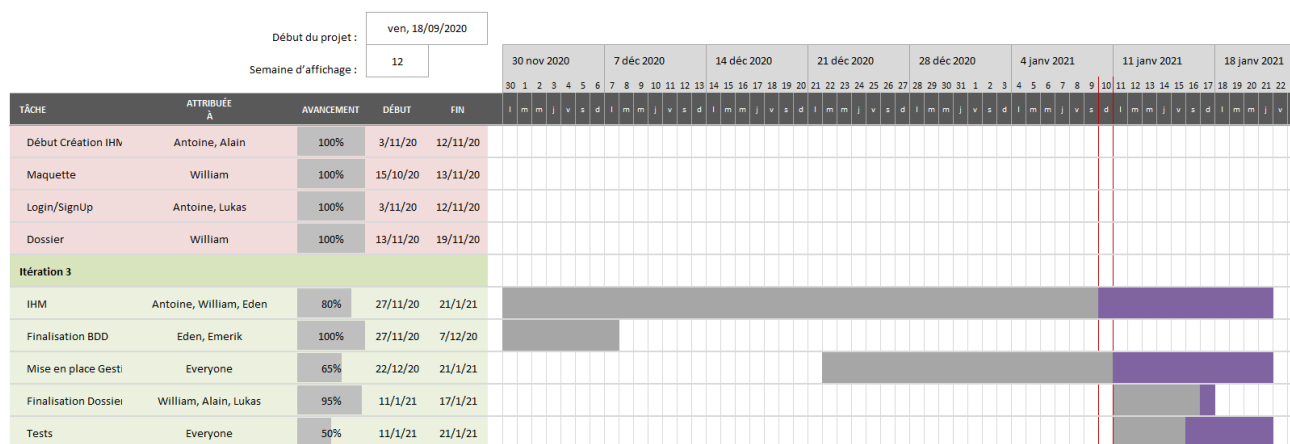


Diagramme 1 Diagramme de Gant Itération 3

*Les diagrammes des autres itérations sont disponibles en annexe.

À ce stade du projet nous avons fini une bonne partie des fonctionnalités que nous souhaitions avoir pour notre prototype néanmoins il y a quelques fonctionnalités (besoins fonctionnels) que nous n'avons pas pu implanter faute de temps. Nous n'avons pas pu implémenter le chat ainsi que les commentaires de versions, le chat s'est révélé être plus compliqué à mettre en place que ce que nous avions prévu et nous n'avons pas eu le temps de nous pencher dessus suffisamment à cause des contraintes de temps que nous avons. Nous n'avons pas non plus eu le temps d'implémenter les commentaires de versions ainsi que les paramètres de projets (qui représentaient peu de fonctionnalités au final). La dernière fonctionnalité que nous n'avons pas eu le temps d'implémenter

est l'attribution de rôle au sein d'un projet, actuellement il existe seulement 2 rôles, créateur du projet et collaborateur du projet. Nous avons peut-être été trop ambitieux, et nous n'avons donc pas pu implémenter ces fonctionnalités à temps. Néanmoins ces fonctionnalités sont toutes des fonctionnalités secondaires de la plateforme.

3.5 Tests

Nous avons effectué un certain nombre de tests à la fin du développement, visant à repérer les bugs et les soucis de conception ainsi que les problèmes de compatibilités. Ces tests nous ont permis de résoudre un grand nombre de ces problèmes. Ces tests sont listés dans le cahier de tests.

3.6 Et après ?

Si nous décidons de continuer à travailler sur ce projet après la fin de cette itération plusieurs possibilités s'offrent à nous.

Tout d'abord nous pourrions passer du Raspberry Pi à une solution plus performante tels que AWS ou Microsoft Azure afin d'augmenter le nombre d'utilisateurs potentiels, mais également d'améliorer les temps de travail. Nous pourrions alors passer nos bases de données dans le cloud afin d'augmenter la sécurité et de réaliser de plus nombreux backups.

Nous pourrions aussi choisir d'utiliser un autre langage que PHP, pour utiliser un langage plus performant, plus récent ou plus sécuritaire.

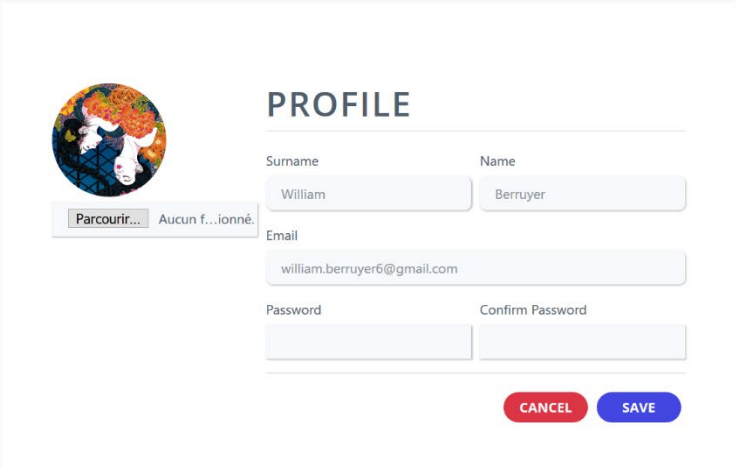
Nous pourrions également améliorer la compatibilité de l'IHM entre les différentes plateformes ainsi que créer de nouvelles pages que nous n'avons pas eu le temps d'implémenter.

Nous pourrions terminer toutes les fonctionnalités que nous n'avons pas eu le temps d'implémenter.

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



The screenshot shows a web form for editing a user profile. On the left, there is a circular profile picture placeholder with a globe and a person's face, and a text input field for a file path with the placeholder text 'Parcourir...' and a button 'Aucun fichier sélectionné'. To the right of the profile picture is the title 'PROFILE'. Below the title are several input fields: 'Surname' with the value 'William', 'Name' with the value 'Berruyer', 'Email' with the value 'william.berruyer6@gmail.com', 'Password', and 'Confirm Password'. At the bottom right of the form are two buttons: 'CANCEL' (red) and 'SAVE' (blue). A small purple circle with the number '0' is located in the bottom right corner of the page.


Page 1 Compte Utilisateur Paramètre 1

Home → Projects → Project Creation



William Berruyer
william.berruyer6@gmail.com

Projects


Create your project 

Project Name :


Software Used

CREATE


Page 3 Création d'un projet









HOME PROJECTS ACCOUNT LOG OUT



Marshall Mathers
Eminem

Project Name 

2 Collaborators

Raw Session  Mix  Master  Chat  Collaborators  Settings 

Project Title

Ownership

You are the owner of this project [Add a co-owner or a new owner](#)

Page 4 Paramètre d'un projet

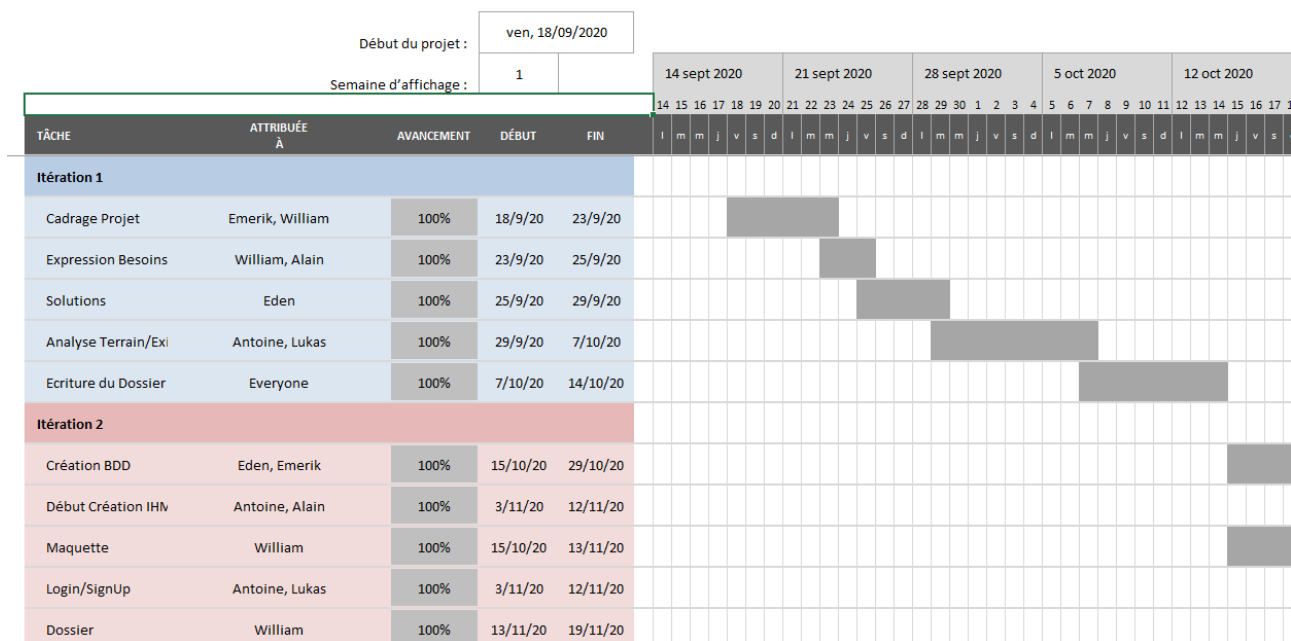


Diagramme 2 Diagramme de Gant Itération 1

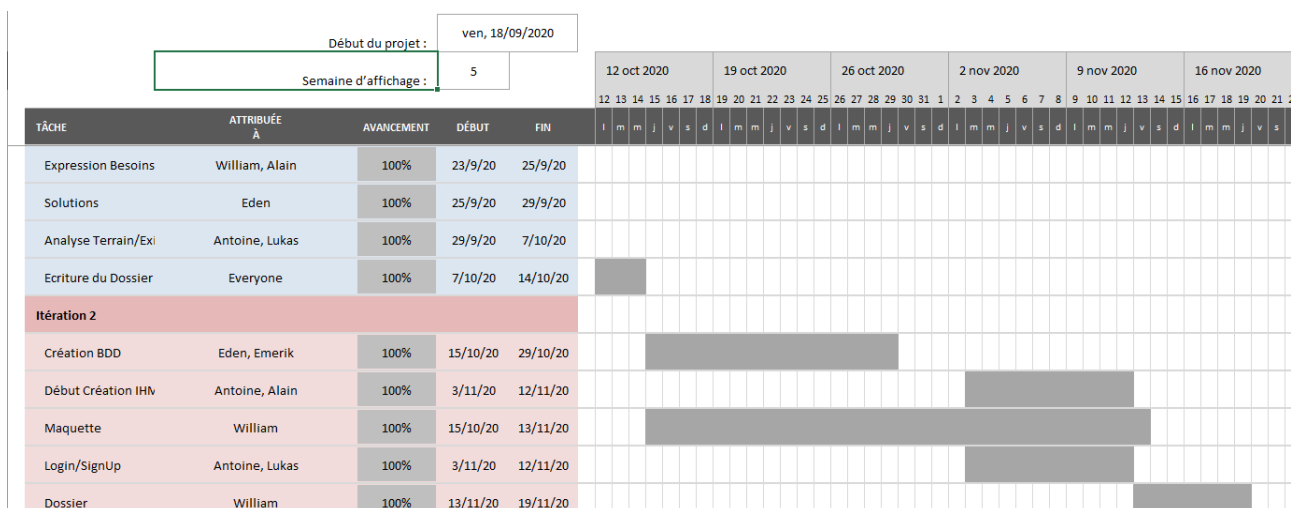


Diagramme 3 Diagramme de Gant Itération 2

Langages



Répartition des langages selon la plateforme GitHub