MASTER EN SCIENCES INFORMATIQUES

Construction d'une plateforme web générique de gestion de rapports

Arnould, Axel

Award date:
2021

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR
Faculté informatique
Année Académique 2020 - 2021

Construction d'une plateforme web générique de gestion de rapports

Axel Arnould

Mémoire présenté en vue de l'obtention d'un master en sciences informatique

RÉSUMÉ

La gestion de rapports est un domaine très vaste. Il se retrouve dans beaucoup de sphères de la vie que ce soit dans le travail ou dans un club sportif. L'objectif principal de ce mémoire est de permettre une gestion simple et accessible à tous des rapports au travers d'un site internet. L'intérêt de cette plateforme est qu'elle soit générique pour permettre à n'importe qui de n'importe quel domaine de l'utiliser. Elle va en réalité être une meta plateforme qui contient plusieurs instances. Ce qui permet de coller le plus possible aux besoins de chaque domaine. La grande difficulté de cette plateforme réside en l'intuitivité qu'elle doit avoir pour permettre à des personnes de tous domaines de l'utiliser aisément et de coller à leurs besoins spécifiques. Après une introduction à la problématique abordée, une première analyse haut niveau va montrer les caractéristiques nécessaires à l'élaboration de cette plateforme. Les outils utilisés vont ensuite être expliqués ainsi que l'analyse en profondeur et le développement de cette plateforme. Pour finir, une enquête est élaborée afin de mettre en évidence les points d'améliorations. Les perspectives d'évolution seront également abordées.

Mots clé : gestion de rapport, Développement web, Nodejs, MySQL, site internet

ABSTRACT

Managing reports is a very broad field. It can be found in many areas of life, whether in the workplace or in a sports club. The main objective of this thesis is to allow a simple and accessible management of reports through a website. The purpose of this platform is to make it generic in order to allow anyone from any background to use it. This platform will in fact be a meta platform that contains several instances. This allows it to meet the needs of each domain. The challenging part lies in the user friendliness of the platform, to allow people from all domains to use it easily and to fit their specific needs. After an introduction to the problem, a first high-level analysis will show the characteristics necessary for its development. The tools used will be then explained as well as the in-depth analysis and its implementation. Finally, a survey will be designed in order to highlight the areas for improvement. The perspectives of evolution will also be discussed.

Keywords : report management, web development, Nodejs, MySQL, website

REMERCIEMENTS

Je remercie le Professeur Jean-Marie Jacquet pour avoir accepté d'être mon promoteur et pour son suivi et ses relectures tout au long de ce mémoire.

Je remercie toutes les personnes ayant participé à ce mémoire dans le cadre de l'enquête réalisée.

Je suis très reconnaissant envers ma famille pour leur soutien et leurs encouragements durant l'écriture de ce mémoire.

Une pensée particulière à Bénédicte et Fiona pour leurs nombreuses relectures.

TABLE DES MATIÈRES

TABLE DES FIGURES

INTRODUCTION

Ce mémoire présente la construction d'une plateforme web générique de gestion de rapport. L'idée derrière ce nom est de proposer une plateforme qui soit utilisable par des personnes venant de différents domaines d'activités.

Le développement d'une telle plateforme ne se réalise pas sans préparation préalable. Il est nécessaire d'analyser différents points afin de produire une plateforme suffisamment aboutie. Ces différents points vont être abordés dans ce mémoire.

Le premier point est la problématique abordée. Il permet de faire un premier tour d'horizon sur ce qui est demandé et ce qu'implique une telle plateforme. La situation initiale et l'orientation que va prendre la plateforme seront également décrites. Il permet aussi de comprendre l'intérêt d'une telle plateforme dans son usage.

Le deuxième point est le concept. Cela permet d'avoir une première analyse haut niveau de la plateforme. Le principe de base de celle-ci sera décrit afin de comprendre comment l'utiliser. Une plateforme d'inspiration va être présentée en décrivant quelles sont les similitudes avec la plateforme qui va être développée mais également les points qui vont différer. Une liste de caractéristiques est développée. Ce sont les caractéristiques que la plateforme doit suivre pour répondre à ses exigences. Et pour finir une analyse haut niveau de toutes les fonctionnalités sera expliquée.

Le troisième point se concentre sur les outils à utiliser pour développer une telle plateforme. Il est évident qu'il ne faut pas laisser ce choix au hasard. Une étude approfondie des différents outils permet de s'assurer de la faisabilité mais également de l'évolutivité du site. Le choix est fait sur les différentes parties du site internet. Que ça soit pour l'aspect visuel, l'aspect logique et la gestion des données.

Le quatrième point décrit l'analyse et le côté programmation du site. Il est extrêmement difficile de faire un projet structuré sans une analyse approfondie des besoins de ce site. Celle-ci complète et examine plus en détails l'analyse effectuée au deuxième point. Ce point décrit également une partie du développement du site.

Le cinquième point confronte le site internet à des utilisateurs. Cette confrontation est réalisée sur base d'une enquête qui a pour but d'étudier l'intuitivité de la plateforme. Puisque celle-ci se veut générique, les participants à cette enquête viennent de domaines différents. Cette enquête permet de mettre en lumière les premiers changements nécessaires pour rendre la plateforme plus intuitive et plus accessible à toutes les personnes.

Le sixième point annonce les perspectives de ce site. La plateforme décrite le long du mémoire peut être améliorée. Il va de soi que pour une approche de production, des améliorations du point de vue ergonomique, programmation et légale doivent être réalisés. Tous ces aspects

sont repris dans ce dernier chapitre et permettent de comprendre comment continuer cette plateforme pour qu'elle soit un jour disponible en ligne.

Tous ces points permettent de mener à bien ce sujet et de comprendre toutes les spécificités dans la création d'une telle plateforme.

Chapitre 1: PROBLÉMATIQUE ABORDÉE

Ce chapitre présente aux lecteurs le sujet de ce mémoire qui est de créer une plateforme de gestion de rapports. Pour commencer, le contexte va être posé ainsi que la situation initiale. Cela permet de délimiter les bases de notre travail. Ensuite, l'orientation du projet sera décrite afin d'en avoir une explication globale. Enfin, un mot d'explication va être donné quant à l'intérêt d'une telle plateforme et à l'usage qu'on pourrait en faire.

1.1 CONTEXTE

Nous vivons à une époque où tout est de plus en plus digitalisé. Il n'est maintenant presque plus possible de travailler sans un support informatique. A l'université, l'outil de travail principal reste l'ordinateur. Les cours sont sur une plateforme en format électronique. Les virements se font sur internet et les commandes pour beaucoup de produits aussi.

Poussée par la crise sanitaire, la digitalisation a été encore plus développée. Le télétravail obligatoire pousse la plupart des entreprises à réinventer leur manière de faire et de penser. Les cours pour les étudiants sont passés en ligne, ce qui contribue à la digitalisation.

C'est dans ce contexte qu'on se rend compte qu'il faut pouvoir s'adapter rapidement. Notamment avec les plateformes qui nous permettent de s'appeler par vidéo, telles que Teams ou Zoom, où la demande a fortement augmenté. Mais il y a aussi une demande de centralisation des données afin de pouvoir distribuer facilement aux différentes personnes les documents, les informations, … dont elles ont besoin pour leur travail.

La plateforme créée au fur et à mesure de ce mémoire est en adéquation avec le contexte actuel. Elle présente l'intérêt de centraliser les informations au même endroit. Elle a été développée afin de pouvoir s'ouvrir au plus de monde possible en rendant l'accessibilité assez intuitive et en limitant les barrières au maximum.

1.2 SITUATION INITIALE

L'objectif de ce mémoire est de créer une plateforme générique de gestion de rapports. L'idée du Professeur Jacquet était de pouvoir créer une plateforme utilisable autant pour les étudiants qui doivent rendre un devoir, un mémoire, … mais également accessible aux professionnels en entreprise. L'intérêt est assez simple, il n'existe pas vraiment une telle

plateforme suffisamment générique qui permette de répondre aux besoins de plusieurs entités différentes en même temps.

La gestion de rapports veut dire qu'un document est déposé à un groupe de personnes et qu'il est possible à chaque personne de commenter et de noter ce document. Ce système est assez utilisé dans le domaine des conférences scientifiques. En effet, pour participer à une de ces conférences, un article doit être déposé afin d'être critiqué par des « reviewers ». Ces critiques permettent de choisir si l'article sera présenté lors de la conférence. Malheureusement, les autres domaines qui seraient susceptibles d'utiliser ce même mécanisme ne possèdent pas toujours d'outil adapté. On peut prendre l'exemple d'un conseil d'administration. Un document est envoyé à tous les membres sur les différents points abordés lors de ce conseil. Les différents membres pourraient alors faire des remarques sur le document. Ce système peut aussi être utilisé dans une commune. Avant le conseil communal, le bourgmestre dépose le rapport avec les différents points abordés. Les échevins pourraient alors émettre leur avis en proposant d'ajouter ou de retirer certains points. Un autre exemple pourrait être dans la gestion d'un club sportif. Bien souvent, le conseil d'administration d'un tel club est composé de plusieurs membres. Ceux-ci pourraient déposer un document contenant les points à aborder dans la prochaine réunion. Les différents membres pourraient alors émettre un avis sur les sujets qu'il faut ajouter ou retirer. Grâce à ces différents exemples, on remarque que beaucoup de domaines ont le même principe de fonctionnement. Il possède alors le problème de ne pas avoir de plateforme dédié pour commenter les documents avant leur réunion. Des solutions alternatives ont été trouvées mais manquent souvent de clarté et de cohésion. Les exemples de solutions peuvent être les mails, les groupes de discussions sur un réseau social, … Aucun de ces exemples n'est prévu pour le problème décrit.

En effet, l'utilisation de mail n'est pas optimal. Soit la personne envoyant le documents aux autres membres va recevoir toutes les réponses sans que les membres voient ces commentaires. Soit tous les membres sont mis en copie de tous les mails mais cela devient très vite illisible quand le conseil est composé de beaucoup de membres. De plus, l'affichage des différents commentaires ne sera pas optimale puisque les mails ne sont pas exactement prévu pour ce cas d'utilisation.

L'utilisation des réseaux sociaux n'est pas forcément meilleur que le mail. Effectivement, un groupe sur Messenger, Whatsapp, Signal, Discord, … sont parfois utilisé mais manque souvent de clarté. Le premier problème est que le document déposé ne doit pas dépasser une certaine taille pour être envoyé. Le deuxième est que la personne voulant donner son avis doit d'abord trouver et lire le fichier mais ce type de groupe n'est pas uniquement dédié à la gestion de rapport. Cela veut dire qu'il faut parfois remonter énormément dans les messages avant de retrouver l'information voulue. En plus, les critiques se mélangent avec les autres messages.

La plateforme pourrait alors donner aux différents groupes un formidable outil pour les aider dans leur travail.

1.3 ORIENTATION DU PROJET

L'objectif de ce mémoire n'est pas de faire une plateforme parfaite prête à être distribuée et utilisée par le plus grand nombre. L'objectif est plutôt de montrer comment d'une méta plateforme, on peut dériver des plateformes concrètes pour différents type d'utilisation tel qu'un conseil d'administration ou de la gestion d'une conférence et faire une preuve de concept de celle-ci. L'analyse permettra de montrer les différentes étapes à réaliser avant de créer le projet. La preuve de concept sera quant à elle développée pour montrer une partie de la réalisation et surtout prouver l'analyse réalisée auparavant.

L'analyse devra être suffisamment poussée et ne pas négliger un sujet en particulier. Pour prendre un exemple, il faut mettre en lumière les différents langages de programmation qui existent afin de montrer l'ensemble des possibilités. Il est également important de ne pas oublier la base de données qu'il convient d'analyser suffisamment, que ce soit pour le paradigme utilisé que pour le système de gestion de base de données. Avant tout cela, il convient évidemment de dégager les grands points clés d'une telle plateforme de façon abstraite pour ensuite se spécialiser dans les différents points. Une ligne de conduite doit également être définie afin de citer les différentes caractéristiques que doit respecter cette plateforme. Grâce à ces caractéristiques, on peut plus facilement savoir quelles sont les valeurs utilisées et intrinsèques à la plateforme.

Il n'est pas nécessaire que la preuve de concept soit autant poussée que l'analyse. Effectivement, le but n'est pas de faire la meilleure plateforme que tout le monde va s'arracher mais plutôt une plateforme simple sans fioritures. Elle n'est pas obligée de posséder un beau design ou des fonctionnalités extrêmement poussées. Elle contient plutôt les fonctionnalités de base qui prouvent la vraisemblance de l'analyse et permet d'avoir une première idée de l'utilisation qu'on peut en faire. C'est en réalité un MVP (Minimum viable product) qui permet de produire des premiers tests sur des potentiels utilisateurs et récolter des feedbacks. Les données récoltées serviront à améliorer encore plus l'expérience utilisateur de cette plateforme.

1.4 INTÉRÊT D'UNE TELLE PLATEFORME

L'intérêt de cette plateforme est décrit dans le nom : elle sera générique. C'est-à-dire que le but final est de la rendre accessible au plus grand monde afin qu'elle soit utilisée dans des contextes qui peuvent être différents. Il faut qu'elle soit suffisamment simple afin de ne pas la réserver à une élite. Intrinsèquement, elle ne doit pas se spécialiser pour un domaine en particulier.

Les exemples d'utilisations ne manquent pas comme celui du conseil d'administration décrit précédemment. On peut aussi penser à un groupe de recherche dont une personne voudrait partager un document, les autres pourraient lui faire une critique constructive. On s'imagine aussi des étudiants qui doivent rendre un travail cela permet de vérifier que tout le monde l'a lu et a pu émettre son avis avant de le rendre. On peut également s'imaginer un groupe de travailleurs dans une entreprise qui doivent rendre un rapport. Le rapport peut être lu et commenté par d'autres personnes en permettant de centraliser les remarques dans une même plateforme.

La liste des exemples d'utilisations pourrait continuer longtemps puisqu'on touche énormément de domaines. En réalité, beaucoup peuvent y trouver un intérêt. C'est le principe même de la plateforme qui permet une grande flexibilité pour les utilisateurs.

1.5 CONCLUSION

Ce chapitre a permis de poser la situation initiale du projet et le contexte dans lequel celui-ci est développé. Il a aussi permis de voir l'orientation qu'il va prendre au fil des prochains chapitres. Cette orientation permet de poser les lignes directrices de l'analyse et du développement de la plateforme. Un mot d'explication a été fait sur l'intérêt d'une telle plateforme pour comprendre à qui s'adresse l'outil. Cela permet aussi de se rendre compte de la portée que cet outil va avoir et d'en prendre conscience lors de l'analyse de celui-ci.

Chapitre 2: CONCEPT

Ce chapitre va présenter les concepts sous-jacents à la plateforme générique de gestion de rapports. Tout d'abord, le principe de la plateforme sera expliqué. Ensuite, une analyse va être effectuée sur une plateforme utilisée pour la gestion de rapport dans le domaine des conférences scientifiques. Cette plateforme est une source d'inspiration pour celle développée dans le mémoire. Néanmoins, notre travail ambitionne d'aborder de nouveaux aspects qui résulteront en des nouveautés présentés dans la suite.

Afin d'obtenir une cohérence et une ligne directrice sur le développement du projet, deux analyses vont être effectuées. La première permet d'expliquer les différentes caractéristiques d'une plateforme idéale. La deuxième va faire une première analyse globale des différentes fonctionnalités nécessaires. Ces deux analyses permettent de ne pas se perdre dans le projet au fur et à mesure de l'avancée. Elles permettent aussi de vérifier la cohérence entre les besoins perçus et la plateforme construite.

2.1 PRINCIPE DE BASE

Le principe de base d'une telle plateforme peut être résumé en quelques phrases. Le but est de mettre en relation des personnes pour qu'elles puissent discuter autour de un ou plusieurs sujets. Un groupe est créé par un utilisateur. Dans celui-ci, un ou plusieurs rapports sont déposés et sont visibles par les participants de ce groupe. Ces participants vont lire le(s) document(s) déposé(s) et exprimer un avis. L'ensemble de ces notes est ensuite visible pour récolter les avis de chacun.

2.2 PLATEFORME SOURCE D'INSPIRATION

La plateforme déjà existante et sur laquelle ce mémoire s'est basé pour comprendre les premiers points de la gestion de rapports est Easychair (1). Cette plateforme est bien connue dans le domaine scientifique. En effet, elle est souvent utilisée pour les conférences scientifiques.

Le principe qui nous intéresse pour le sujet de ce mémoire, c'est comment EasyChair gère les reviews des différents articles déposés pour espérer participer à une conférence.

Pour commencer, il faut créer une conférence. La création se fait sur base d'une réelle conférence physique ou en ligne. Il est nécessaire de s'y prendre suffisamment à l'avance afin de permettre ce processus de review. Lors de cette création, il faut ajouter certaines deadlines telles que le début de la conférence, la date de fin pour envoyer sa review, … Dans ces conférences, deux rôles sont disponibles. Les personnes du premier rôle déposent leur article et les personnes du deuxième rôle vont lire et faire une critique des fichiers déposés. Une même personne peut jouer les deux rôles en même temps. C'est-à-dire qu'elle dépose un document et donne également un review (ou une évaluation) sur un document d'une autre personne.

Il faut d'abord que les personnes voulant être acteurs de la conférence déposent un article. Cela donne un ensemble de tous les articles qu'il va falloir noter et commenter. Une fois la date limite dépassée, il n'est plus possible de déposer.

Puisque la même personne peut déposer et critiquer un article, quelques règles doivent être mis en place avant le début des reviews. Les conflits d'intérêts entre les personnes participantes sont déclarés. Cela permet de ne pas proposer un article à un reviewer avec lequel il serait en conflit d'intérêt. Il semble évident également qu'une personne ne peut pas critiquer son propre article.

Ensuite, les articles sont visibles par les personnes dites « reviewers » tout en respectant les règles décrites juste avant. C'est à ce moment-là qu'ils choisissent un article à critiquer en fonction de leur domaine de prédilection et leur attrait pour le sujet. C'est évidemment impossible pour un reviewer de lire tous les articles disponibles, cela prendrait beaucoup trop de temps.

Une fois ce choix fait, il est temps de lire et d'émettre une critique accompagnée d'une note au(x) article(s) choisi(s). Les notes peuvent être comprises entre -2 (la note la plus basse) et 2 (la note la plus haute). Elles permettent de savoir si l'article va être présenté à la conférence ou non. L'article qui obtient -2 par l'ensemble du groupe de personnes l'ayant noté est en très mauvaise posture et obtiendra alors un rejet. Cela signifie qu'il n'a pas le droit d'aller plus loin et de présenter son article à la conférence. A contrario, un article qui reçoit 2 de tous ses reviewers est en très bonne posture pour continuer le processus. Les notes sont toujours accompagnées d'un commentaire afin de justifier la note et ce qu'il y a à améliorer, changer, supprimer, …

Bien évidemment, c'est assez rare d'obtenir la note minimale ou maximale de l'ensemble des critiques. La plupart du temps, c'est plus nuancé. Il est d'ailleurs possible dans le temps alloué à la période de reviewing de changer sa note et son commentaire. Par exemple, une personne met la note de 0 avec un commentaire mais voit que les autres ont tous mis des notes plus basses. En lisant les différents commentaires, la personne peut voir qu'elle est passée à côté de quelque chose et modifier sa note et son commentaire. La première critique de cette

personne est encore visible mais n'est plus prise en compte. C'est la deuxième qu'on considère comme efficiente.

Quand toutes les critiques sont postées et que la deadline est dépassée, c'est à ce moment-là que le choix des articles retenus s'opère. Il est évident que la note finale aide énormément dans ce choix. Il y a cependant quand même un administrateur qui aide au choix. Pour ce faire, il existe une petite discussion en ligne assez simple afin que les personnes qui doivent évaluer le même rapport ainsi que l'administrateur puissent discuter. Dans certains cas, l'administrateur donne la décision finale et attend un retour bien souvent quand les notes sont assez serrées. Il va dire si l'article va être retenu ou non pour la conférence. Il est d'usage que les critiques donnent leur avis sur cette décision.

Une fois tous les articles pour participer à la conférence choisis, chaque auteur reçoit un mail. Celui-ci contient plusieurs informations. Il contient le fait que l'article est accepté ou non ainsi que les différentes remarques faites par les différents reviewers.

Le cas décrit n'est pas absolu. Il peut y avoir des subtilités différentes dans chaque exécution mais les grandes lignes directrices restent les mêmes.

Il est à noter que la plateforme EasyChair permet de réaliser d'autres tâches mais il n'est pas nécessaire de s'étendre davantage sur les autres domaines d'actions car cela n'apportera pas de renseignements supplémentaires à la plateforme que ce mémoire va développer.

Il est évident qu'une plateforme qui se veut générique ne doit pas simplement recopier la plateforme EasyChair. Celle-ci s'adresse à un public et un domaine particulier et pas à un ensemble de domaines.

2.3 CARACTÉRISTIQUES DE LA PLATEFORME IDÉALE

Après l'analyse de la plateforme EasyChair, il est possible de décrire des caractéristiques que notre plateforme devra suivre pour accomplir ses objectifs. Cette section va citer ces différentes caractéristiques et expliquer pourquoi elles sont si importantes. Les différents points sont des concepts qu'il va falloir respecter au fur et à mesure du développement.

2.3.1 Facilité d'utilisation

La plateforme se doit d'être facile d'utilisation puisqu'elle se veut être une plateforme générique de gestion de rapports. Si la plateforme est générique, il est évident que des gens de tous horizons et de tous domaines d'activités vont être susceptibles d'utiliser celle-ci. En outre, au plus elle est facile à utiliser de par des fonctionnalités intuitives et une bonne ergonomie, au plus elle va susciter l'adhésion du plus grand monde. Personne n'a envie d'une

plateforme incompréhensible où il faut chercher longtemps avant de trouver la fonctionnalité que l'on souhaite.
Pour cela l'analyse en amont des différentes fonctionnalités à implémenter et de l'intuitivité de l'interface doit être assez poussée. La plupart du temps quand un utilisateur va sur un site, il ne lui faut pas longtemps avant de juger celui-ci. Si l'utilisateur remarque une interface utilisateur déplorable il y a beaucoup de chance pour qu'il ne reste pas. Heureusement, beaucoup se sont déjà posé la question dont le géant d'internet Google (2 & 3). En effet, Google propose pas mal de tutoriels pour rendre une site, une application mobile, … les plus attractifs possible. C'est une sorte de charte de bonne conduite que Google recommande de respecter pour avoir l'interface utilisateur la meilleure possible. Même si Google n'oblige pas du tout les développeurs à suivre leur charte, ce sont des documents qui ont été rédigés grâce à plusieurs années de recherches par une des plus grandes entreprises d'internet donc cela reste extrêmement pertinent.
Pour rencontrer ce critère, la plateforme va être développée en suivant les consignes d'ergonomie d'un site de Google. Une enquête va également s'opérer sur des utilisateurs potentielles. Tout cela sera expliqué dans le chapitre 5.

2.3.2 Adaptabilité

Etant donné que cette plateforme est générique, il est primordial qu'elle s'adapte aux différents domaines d'activités. L'objectif est de ne pas pénaliser l'utilisateur par des fonctionnalités non pertinente. Cela étant il est important de conserver des fonctionnalités de base et de permettre à la plateforme développée de rester adaptive. Il y a pas mal choses qui permettent à un site internet d'être adaptatif. On peut penser au fait d'être accessible à tous mais aussi à la traduction dans la langue de l'utilisateur. Il faut aussi penser à satisfaire les objectifs que chaque utilisateur a en utilisant cet outil.

Pour répondre à ce critère, des choix vont être laissés à l'utilisateur pour personnaliser certaines fonctionnalités de la plateforme. Par exemple, lors de la création d'un projet pour déposer des documents, il sera possible de choisir si il faut noter le document et mettre un commentaire ou si un commentaire suffit. Un deuxième exemple est qu'il sera possible de décliner la plateforme en fonction des domaines d'utilisation. Si un club de sport utilise la plateforme, le thème ne sera pas le même que si c'est un conseil communal qui l'utilise.

2.3.3 Compatibilité

Nous sommes dans une époque où les objets électroniques sont de plus en plus puissants et de plus en plus diversifiés. Nos téléphones portables en sont un bel exemple puisqu'ils sont parfois aussi puissants qu'un ordinateur. De plus en plus de personnes utilisent leur téléphone comme un véritable outil de travail. Pour satisfaire ces potentiels utilisateurs, la plateforme se doit de s'adapter à toutes les tailles d'écrans et à toutes les utilisations possibles. Pour

rejoindre un peu la caractéristique citée juste avant, le site doit être facile d'utilisation sur tous les appareils possibles sur lesquels nous sommes potentiellement susceptibles de l'utiliser (ordinateurs, tablettes, téléphones, …).
L'appareil physique est un point mais il ne faut pas oublier qu'il doit pouvoir afficher la plateforme et cela se fait sur un navigateur web. La compatibilité se fait également à ce niveau-là. Il est primordial que le site web soit accessible par les navigateurs préférés des utilisateurs. Il ne faut pas non plus qu'il y ait une grande différence d'affichage entre les différents navigateurs afin de ne pas perdre un utilisateur qui passerait de Firefox à Chrome par exemple. Pour parer à ce problème, il va falloir être très attentif aux différents langages, outils utilisés lors de la création du site. Certains ont déjà des fonctionnalités intégrés qui permettent de s'adapter facilement à différentes tailles d'écrans

2.3.4 Compétitivité

Il est évident qu'une plateforme idéale est une plateforme qui est au moins aussi bien que ses concurrentes. Sinon, il n'y a aucun intérêt pour des utilisateurs d'utiliser celle-ci. Il y a beaucoup de moyens de rester compétitif. Nous ne sommes pas obligés d'avoir forcément la meilleure plateforme avec les meilleures fonctionnalités par rapport aux concurrents. Cela peut simplement être le fait que la plateforme a les fonctionnalités essentielles que les utilisateurs attendent au minimum. En contrepartie, la plateforme est nettement moins chère voire gratuite par rapport à ses concurrentes directes. Il peut aussi y avoir une fonctionnalité qui n'existe que sur cette plateforme et pas sur les autres. Ou encore, la facilité d'utilisation et l'intuitivité avec laquelle on utilise la plateforme est meilleure. Il faut toutefois faire attention car les sites concurrents ne sont pas laissés à l'abandon et peuvent très bien faire des modifications qui les rendraient beaucoup plus utiles et attrayants.

2.3.5 Evolutivité

L'évolutivité est peut-être une des caractéristiques la plus complexe car elle regroupe beaucoup de points. En effet, cela passe par le choix de base des langages de programmation mais aussi du fait que la plateforme est continuellement améliorée et donc implique un suivi par des développeurs. Pour une évolutivité continue, il faut absolument choisir un ou plusieurs langages de programmation qui sont encore très utilisés et qui bénéficient d'une popularité déjà énorme. Evidemment, on ne peut pas prendre un langage qui est de moins en moins utilisé car nous n'avons aucune certitude qu'il va être maintenu encore des années ou qu'il sera encore compatible avec les prochaines mises à jour de nos navigateurs. Il est aussi difficile de choisir un tout nouveau langage qui commence seulement à se faire connaitre puisque pour lui non plus, nous n'avons aucune certitude quant à son avenir et au fait qu'il va être maintenu dans le temps. Ce principe est exactement le même pour les SGBDR (système de gestion de base de données relationnelles). Le bon compromis peut être de choisir un outil

qui est très utilisé à l'heure actuelle et qui est soutenu et utilisé par les grands noms des entreprises web. Il est très peu probable qu'un langage soutenu et utilisé par Google, Facebook, Amazon, … disparaisse du jour au lendemain.

L'architecture du projet a son importance aussi dans l'évolutivité. Si celle-ci est bien analysée et réalisée dès le départ, il est beaucoup plus facile de maintenir le projet au fur et à mesure des années. A contrario, si l'architecture est mal faite au départ, il va être difficile pour un développeur de comprendre le code et de pouvoir modifier certaines parties en étant sûr de ne pas faire d'erreur quelque part. De plus, s'il faut réaliser toute l'architecture une nouvelle fois, la perte de temps sera considérable.

Un autre point important dans l'évolutivité est la lisibilité du code et les différents commentaires présents dans celui-ci. Il est important de maintenir un code clair avec des noms de variables explicites en n'oubliant pas de placer des commentaires compréhensibles à des endroits stratégiques du code afin de clarifier au maximum. Cela pourrait aider le développeur principal ou un développeur qui reprend le projet en cours de route pour lui faciliter sa compréhension et éviter les erreurs.

Ces points vont être respectés au mieux lors des différentes étapes de développement. Ils seront repris et expliqués lors des deux prochains chapitres.

2.3.6 Utilité

L'utilité est sans aucun doute la caractéristique la plus importante. En effet, une plateforme qui n'a pas d'utilité n'ajoute pas de valeur et ne sera pas utilisée. Vu le nombre de personnes qui utilisent EasyChair, l'utilité d'une telle plateforme n'est pas à démontrer. Sachant que la plateforme se voudra ouverte à tous domaines, on peut imaginer que plus de monde sera susceptible d'en avoir besoin. Le but aussi est de faciliter la vie d'utilisateurs qui ont dû trouver des systèmes alternatifs (bien souvent l'envoi par mail ou les réseaux sociaux). La plateforme ne doit pas déborder d'outils en tout genre pour satisfaire le moindre désir, mais elle doit faire son travail principal sans soucis et le plus facilement possible pour l'utilisateur.

L'objectif est la qualité de la plateforme plutôt que la quantité. Il vaut mieux des fonctionnalités complètes et pratiques qu'énormément de fonctionnalités incomplètes et peu pratiques.

Après le listing et l'explication de ces différentes caractéristiques, on se rend vite compte qu'il sera important de respecter celles-ci. Elles sont primordiales pour que l'utilisateur ne soit pas perdu devant son écran. Cela permettra aussi aux utilisateurs de faire plus facilement la transition entre la plateforme qu'ils utilisent habituellement et celle-ci.

Les caractéristiques citées vont jouer un rôle sur les outils choisis pour le développement de la plateforme. En prenant l'évolutivité comme caractéristique, il faut des outils qui vont être

maintenu dans le temps. La compatibilité aussi nécessite une analyse des différents outils afin qu'ils soient adaptés aux différents navigateurs, ordinateurs, téléphones, …

2.4 ANALYSE HAUT NIVEAU DE LA PLATEFORME

Afin d'analyser au mieux le système développé dans ce mémoire, un cas d'usage, plus connu sous le nom de « use case » en anglais, va être fait. Le cas d'usage (4) permet de mettre en évidence les exigences d'un système. Une fois créé, il sert de fil conducteur à l'ensemble des étapes de développement du projet.

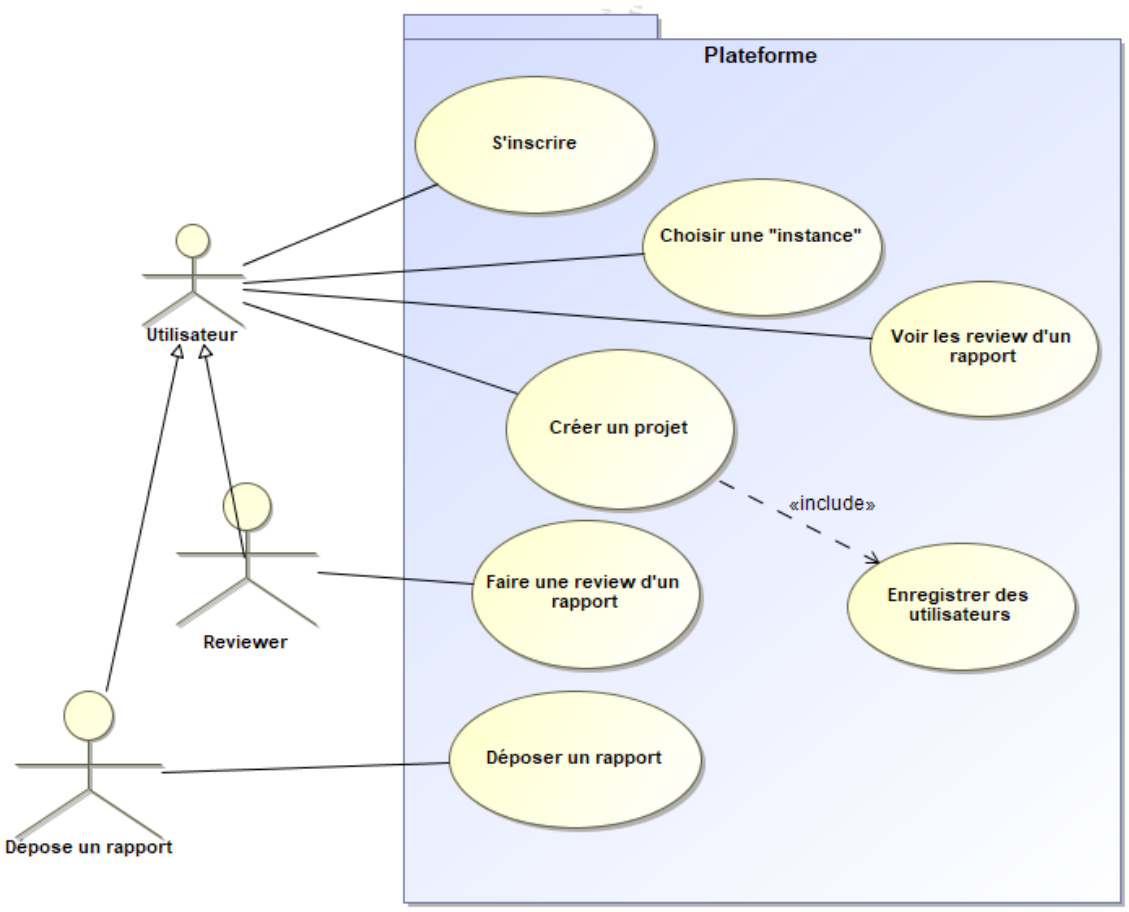

Figure 1 Use case

En voyant ce cas d'utilisation qui correspond à celui de la plateforme, toutes les actions possibles à faire avec celle-ci sont notées. On a alors une vue d'ensemble de l'utilisation de la plateforme. Dans la suite, il va falloir aller plus en profondeur dans ces différents points.

Avec ce use-case, il est possible de dégager, sous une forme de liste, les différents éléments qui doivent apparaitre dans la plateforme.

- Système de login
- Un système de choix d'instance

- Un système de création de projets
- Un système d'ajout ou d'inscription au projet
- Un système de rôle dans un projet(ex : reviewer ou déposer ou les deux)
- Un système d'ajout de papier/rapport à un projet
- Un système de commentaires et de votes des reviewer
- Un système qui permet de voir toutes les remarques d'un rapport en particulier.

On remarque en lisant cette liste qu'elle structure bien le projet. Elle permet une analyse plus aisée dans la suite que ce soit pour la base de données, le choix des différents outils, le choix des fonctionnalités,…

Le système de login est important car il permet d'identifier de manière unique l'utilisateur. Cette caractéristique offre la possibilité de créer des sous-groupes sur la plateforme. Sans cela, toute personne se connectant au site pourrait voir tous les projets créés.

La possibilité de choisir une instance est l'atout de la plateforme. Cela permet de la différencier des autres plateformes. En effet, le but de ce site internet est qu'il soit utilisable dans différents domaines. Néanmoins, il semble évident que tous les domaines n'ont pas besoin d'exactement les mêmes fonctionnalités. Les instances vont en fait permettre d'avoir une version de la plateforme dédié à leur domaine. Il existe par exemple une version pour les clubs sportifs, une autre pour les conférences scientifiques, pour les conseils communaux, …

A la création d'un projet, l'élaborateur doit sélectionner d'autres utilisateurs pour participer à celui-ci. Il est alors possible d'ajouter des documents que les participants peuvent commenter. Pour chaque document déposé dans ce projet, les utilisateurs peuvent voir les critiques des autres participants.

Cette analyse de haut niveau permet d'appréhender les principales fonctionnalités de la plateforme. Cela aide dans le choix des outils à utiliser mais également dans l'analyse complète.

2.5 CONCLUSION

Dans ce chapitre, une première analyse a été faite. Au départ, une brève explication du principe de base de la plateforme a permis de mieux comprendre vers quoi le projet se dirige. Ensuite, un tour d'horizon a été fait afin de se rendre compte de ce qui existait déjà. Ce tour d'horizon a permis d'expliquer en détails la partie du site EasyChair qui est la plateforme qui nous a inspiré. Cette partie a aussi mis en lumière les différences qu'il va y avoir entre ces deux plateformes.

Une liste des caractéristiques de la plateforme idéale a été énumérée. Chaque caractéristique a été détaillée afin de comprendre la logique de chacune mais également de savoir comment

cela va se répercuter sur la plateforme. Cette liste va servir le long du développement du projet.

Une analyse globale a aussi été réalisée, ce qui permet d'avoir les grandes fonctionnalités qui doivent être implémentées. Cela permet de comprendre les exigences du système mais aussi d'avoir un fil conducteur dans les étapes de développement du système.

Grâce aux différents points abordés dans ce chapitre, il est désormais possible de choisir les outils nécessaires à la confection de la plateforme. Ce choix va être fait dans le chapitre suivant.

Chapitre 3: OUTILS

Dans ce chapitre, un tour d'horizon sur tous les outils et les langages est fait afin de sélectionner les meilleurs d'entre eux pour réaliser la plateforme. La première partie va être consacrée aux langages web. Les langages les plus pertinents vont être passés en revue que ce soit pour le frontend ou le backend. Une brève explication va aussi être réalisée sur les différents outils qui peuvent aider le programmeur et qui gravite autour des langages présentés. A la fin, un choix sera réalisé entre tout ce qui a été présenté.

Dans la seconde partie, une description est réalisée sur la façon de gérer les données et les outils disponibles pour le faire. Une explication est également donnée sur les différences qui existent entre les bases de données. De la même façon, un mot d'explication va être fait sur les différents systèmes de gestion de base de données qui existent. Comme dans la première partie, un choix est fait entre tous les outils présentés à la fin.

3.1 LANGAGE WEB

3.1.1 Mise en contexte

Pour faire une plateforme de gestion de rapports, il faut déterminer un langage de programmation web. Trouver des langages de programmation web actuellement est assez facile. Il y en a de plus en plus qui se développent avec chacun leurs spécificités et leur popularité. Mais choisir un langage web est moins aisé. Il est bien évident qu'ils ne se choisissent pas au hasard et qu'il vaut mieux bien analyser ses besoins pour faire le choix le plus judicieux. Cette section propose un bref aperçu des langages qui existent et qui ont une certaine popularité et présente ensuite une analyse du langage que je vais prendre en fonction de mes besoins. Elle va se concentrer sur des langages populaires car ils ont souvent une communauté très active. Cela permettra d'avoir beaucoup de réponses aux questions qui pourraient apparaitre en cours de programmation. Mais avant ça, un petit point théorique s'impose sur la structure d'un site web.

3.1.2 Analyse de la structure d'un site web

Pour démarrer dans la programmation web, il faut être au courant d'une chose primordiale. C'est qu'un site web se sépare en deux parties distinctes. Ces parties sont le front end et le back end (5).

Le front end est la partie visible du site web. C'est tout ce qui est sur la page que l'on consulte que ce soit les titres, les images, le texte, les boutons, … bref tout ce que l'on voit et avec quoi on peut interagir. Quand on développe le front end, on s'occupe en fait de l'interface graphique mais aussi de l'ergonomie du site pour qu'il soit intuitif. Cette partie est vraiment importante car elle est en contact direct avec l'utilisateur.

Le back end est plutôt la partie invisible mais sans cette partie un site web reste très limité. Il est en réalité composé de trois parties. D'abord, le serveur qui permet d'héberger le site web. C'est comme ça qu'il est consultable par tous. Ensuite, l'application en elle-même (administration du site web, …). Et enfin, la base de données qui est un composant essentiel pour enregistrer des informations et les retrouver ensuite. Cette partie est très importante et pour ne pas la négliger, un point lui est consacré.

Dans l'explication de ces deux parties, on se rend compte qu'elles sont presque indissociables. On pourrait se dire que pour un site dit « vitrine », nous n'avons pas besoin de backend. Mais même pour ce type de site, nous en avons besoin puisque pour qu'il soit consultable par tous, il est nécessaire de l'héberger sur un serveur. Ce qui est une des trois fonctions du backend.

Il est, par contre, évident que ces deux parties n'utilisent pas le même langage de programmation puisqu'ils ont une fonction très différente. Pour le back end, on dit qu'on utilise un langage serveur alors que pour le front end, on utilise plutôt les langages comme html, css et javascript.

3.1.3 Frontend

Dans le cas du front end, ce sont majoritairement trois langages qui ressortent et qui sont complémentaires dans l'élaboration de cette partie. Une brève explication de chacun est réalisée dans la suite.

3.1.3.1 HTML

L'acronyme HTML (6) qui veut dire Hyper Text Markup Langage réfère à un langage de programmation qui a la particularité d'être composé de tags. Sa syntaxe est donc un peu différente d'autres langages qu'on a l'habitude d'utiliser. Il est conçu pour représenter les squelettes des pages web. Sa dernière version stable actuelle est la version 5. Elle est aussi nommée HTML5. Il est presque tout le temps associé au css et au javascript. Il s'est imposé avec le temps comme le langage de référence du web pour créer le frontend des sites internet. L'associer avec du css et du javascript permet une plus grande liberté sur l'affichage et sur le dynamisme de nos sites internet.

3.1.3.2 CSS

Le css qui veut dire Cascading Styles Sheets permet de « décorer » les pages web (7 & 8). Grâce à des sélecteurs, il permet de modifier des éléments HTML en ajoutant un style à ceux-ci (couleur, taille, arrière-plan, …) Sans lui nos pages web se ressembleraient toutes et ne seraient vraiment pas attrayantes. Il permet aussi de mettre plus facilement les éléments là où on veut qu'ils soient en utilisant certains attributs de placement. Il est à noter que pour l'instant avec l'html et le css nos pages web restent statiques.

3.1.3.3 Javascript

Bien que Javascript ne soit pas obligatoire pour créer un site internet, il permet de mettre un peu de vie au sein des pages web (9). Il permet de l'interactivité et de la logique sur celle-ci. Par exemple, il va s'occuper de certaines séquences après avoir appuyé sur un bouton, un menu déroulant, … Une des grandes forces du javascript est qu'il existe beaucoup de frameworks qui facilitent la vie du développeur. Il y a par exemple la bibliothèque libre Reactjs créé en 2013 qui permet de créer des applications web ou mobile monopage plus facilement. Il existe aussi Angular ou VueJS pour ne citer que les plus populaires. Ces frameworks contribuent également à la popularité de ce langage. Ceux-ci vont être expliqués ci-après.

Il faut savoir que ces trois langages peuvent être utilisés sur un même fichier qui serait un .html. Mais il est plus fréquent et plus propre surtout lorsque l'on programme un site assez imposant en termes de lignes de codes de les séparer. La séparation se fait alors en minimum trois fichiers distincts (.html, .css, .js). Les fichiers .css et .js sont appelés dans le fichier html grâce à un tag prévu à cet effet.

3.1.3.4 Frameworks et bibliothèques javascript

Avant de passer aux différents langages backend, il est important de faire le point sur les frameworks et bibliothèques créés autour du langage javascript. Tout d'abord, il faut saisir la nuance entre une bibliothèque et un framework (10).

Une bibliothèque est composée de nombreuses classes et fonctions que l'on peut utiliser directement dans le code. C'est un ensemble d'outils qui a pour but de faire gagner du temps dans le travail d'intégration et de développement. Si une ressource dont on a besoin est disponible dans cette bibliothèque autant l'utiliser plutôt que de réinventer quelque chose qui existe déjà.

Un « framework » n'est pas une bibliothèque. Bien souvent une confusion est faite parce que dans la plupart des cas, un framework inclut une ou plusieurs bibliothèques. Un framework est en fait une infrastructure logicielle qui constitue un schéma de fonctionnement. Cette infrastructure est elle-même constituée de composants logiciels qui sont autonomes.

La différence entre un framework et une bibliothèque est assez subtile à comprendre. Pour un framework, c'est lui qui va appeler le code. En tant que développeur, on gère juste les différentes données que l'on peut lui injecter, leur traitement et leur affichage. Le framework

se charge du reste. Une bibliothèque n'étant qu'un inventaire d'outils, c'est notre code qui l'utilise.

3.1.3.4.1 ReactJS

ReactJS (11) est une bibliothèque javascript. Elle est libre et est développée par Facebook depuis 2013. React est utilisé par les plus grands tels que Netflix, Sony, Airbnb et évidemment Facebook.

C'est Jordan Walke , ingénieur chez Facebook, qui a créé le premier prototype appelé FaxJS. Il a été déployé pour la première fois en 2011 sur le fil d'actualité de Facebook. Mais c'est le 29 mai 2013 que la version 0.3.0 fait la première diffusion publique.

3.1.3.4.2 AngularJS

AngularJS(12) est un framework web front-end open-source qui est maintenu et développé par Google. Il a pour but de faciliter le développement d'application monopage (SPA : Single page application). La première version connue d'AngularJS est apparue fin 2010. AngularJS repose sur le principe que la programmation déclarative doit être utilisée pour créer l'interface utilisateur alors que la programmation impérative doit être utilisée pour la logique métier. Ses objectifs sont de découpler un maximum le coté client du coté serveur d'une application.

3.1.3.4.3 Bootstrap

Bootstrap (13 & 14) est un framework CSS open source pour le développement front end réactif et mobile. Il a été développé au départ par Mark Otto et Jacob Thornton sur twitter pour garder une cohérence entre tous les outils internes. Pendant quelques mois seulement un petit groupe de développeurs travaillaient sur ce projet. Mais dans le cadre de la Hack Week (une sorte de hackathon chez Twitter) de nombreux développeurs chez twitter se sont penchés sur son développement. Il a fini par être publié le 19 août 2011. Plusieurs versions de Bootstrap se sont succédées jusqu'à arriver à la version actuelle qui est la version Bootstrap 5 Beta 3 qui est apparue le 23 mars 2021. Certains développeurs préfèrent rester sur Bootstrap 4 qui elle est arrivée le 29 octobre 2014 car la version 5 est encore une béta alors que la version 4 a presque 7 ans d'ancienneté.

3.1.3.4.4 VueJS

VueJS (16) est un framework javascript open-source permettant la construction d'application web monopage. Il a été créé par Evan You (15) qui a sorti la première version le 11 février 2014. VueJS est notamment utilisé dans des grandes enseignes connues tel que Adobe, Alibaba et Gitlab. Il se veut progressivement adaptable.

Evan You a travaillé chez Google en utilisant principalement AngularJS. Le but de VueJS à la base était pour Evan You d'extraire la partie qu'il aimait vraiment d'AngularJS pour développer quelque chose de vraiment léger.

3.1.3.5 Moteur de template

Un moteur de template permet de combiner des données avec des pages web. Il permet également d'ajouter certaines conditions en fonction des données récupérées. Etant donnée que le plateforme va devoir afficher beaucoup de données, il est très utile d'en utiliser un. Beaucoup de moteurs de template existent sur le marché tel que Twig, Smaty ou encore Handlebars.

Ces différents moteurs de template ont évidemment chacun leurs spécificités. Néanmoins, l'utilisation faite dans cette plateforme ne nécessite pas de fonctionnalités très complexe. Celle-ci consiste en réalité uniquement en l'affichage de variable, les conditions et les boucles. Pour ce faire, Twig sera utilisé lors du développement de ce projet.

3.1.4 Backend

Vu les fonctions que doit remplir le back end, il faut un langage polyvalent. Il doit être capable d'être le métronome. Il doit communiquer avec la base de données, répondre aux demandes du site pour les changements de pages par exemple. Il remplit beaucoup de fonctions. Heureusement, il existe une multitude de langages qui peuvent être utilisés dans le backend. Dans cette section, je ne vais approfondir que les plus connus. Il est évident qu'il existe bien d'autres langages qui pourraient remplir cette fonction.

3.1.4.1 Java

Java (17) est sans doute un des langages les plus utilisés ces dernières années si pas le plus utilisé. Il est assez polyvalent c'est-à-dire qu'on peut utiliser java dans plusieurs contextes que ce soit pour des logiciels, pour des applications android ou encore des applets dans les navigateurs web. C'est une des raisons pour lesquelles il est aussi populaire. Une des caractéristiques de ce langage est qu'il est orienté objet. Un objet est la représentation d'une entité du monde physique ou d'un concept. L'orienté objet lui consiste en la définition et l'interaction entre ces objets dans le programme. Ce qu'on sait moins sur ce langage c'est qu'il peut être utilisé pour la programmation backend d'un site internet. De plus, certains frameworks java pour le backend tels que Spring (18) pour ne citer que lui permettent de faciliter la programmation pour le côté serveur.

3.1.4.2 Python

Python (19) est, comme java, un langage très populaire. Il est, par contre, un peu différent. En effet, on dit qu'il est multi paradigme car il peut être orienté objet, impératif et même fonctionnel. Ce langage laisse donc une grande liberté aux personnes qui l'utilisent. Il est considéré comme assez simple à apprendre parce qu'il a été conçu pour améliorer la productivité du programmeur grâce notamment à une syntaxe simple. Il a aussi comme différence avec java le fait que c'est un langage interprété. Par la suite, les différentes manipulations d'analyse et de traduction sont faites à chaque exécution du programme. Python est largement utilisé pour faire des scripts ou faire des prototypes rapidement grâce à sa facilité d'utilisation. La partie web de python est principalement son framework django qui

se veut simple et rapide. Le slogan est d'ailleurs « Le framework pour les perfectionnistes avec des deadlines » (20). En effet , ce framework permet de gérer le back end sans forcément beaucoup de lignes de code.

3.1.4.3 Ruby

Ruby (21) a été créé par Yukihiro « Matz » Matsumoto en 1993. Il décide de le créer suite à plusieurs frustrations quand il développait en Smalltalk et Lisp.

C'est un langage qui, tout comme Python, est multiparadigme puisqu'il peut être objet, impératif, concurrent et fonctionnel. Il a aussi un autre point commun avec Python car il est interprété. Sa syntaxe est, quant à elle, inspirée de Eiffel et Ada. Il a connu un engouement depuis l'arrivée du framework web ruby on rails et de sa documentation traduite en anglais.

Ruby a la particularité que tout est objet. Même pour les types de données primitives. Il est réputé pour sa lisibilité mais aussi pour son côté « facile à apprendre ».

Son framework Ruby on Rails (22) est celui qui nous intéresse ici puisque c'est un framework web qui est de plus en plus utilisé. Il est open source. On peut contribuer à son développement et ce sont près de 5000 personnes qui y ont déjà contribué. Il permet de ne pas écrire énormément de lignes de code mais nécessite un grand niveau d'abstraction. Il a pour avantage de fournir des outils pour faciliter le déploiement de l'architecture MVC (modèle – vue – contrôleur) dans nos projets.

3.1.4.4 Scala

Scala (23) est un langage multiparadigme. Il est concurrent, fonctionnel, impératif et orienté objet. Scala fonctionne sur la machine virtuelle Java. Il existe un compilateur Scala qui compile en javascript qui se nomme Scala.js. Il permet d'écrire des programmes en Scala qui fonctionnent sur les navigateurs web ou Nodejs. Il est donc à la fois dans le front end et dans le back end. Le framework play permet aussi d'utiliser Scala dans le backend. Le plus gros avantage de scala est son côté multiparadigme qui permet une grande flexibilité dans le code. Attention, la liberté qu'autorise Scala dans son côté multiparadigme nécessite une certaine expérience pour être bien manipulé.

3.1.4.5 PHP

PHP est un langage qui a été créé à la base par Rasmus Lerdof en 1994 qui n'était en fait à l'origine qu'une bibliothèque logicielle en C (24). Au fur et à mesure qu'il avançait sur son site, il ajoutait de nouvelles fonctionnalités. C'est alors qu'il a transformé sa simple bibliothèque en un outil capable de communiquer avec des bases de données et de créer des applications dynamiques. Aujourd'hui, PHP en est à sa version 8.0. Il a parcouru pas mal de chemin depuis ses débuts et a considérablement été amélioré. Parmi les événements importants dans la vie de PHP, on peut noter qu'en 1998 il est passé d'une personne à le développer à une équipe de développeurs (PHP 3.0). En 2000, il y a eu un ajout d'un système d'analyse syntaxique (Zend Engine) pour la sortie de PHP 4.0. Et en 2015, la performance a été grandement améliorée grâce aux nouveaux moteurs « phpng » ce qui lui a permis d'être presque 2 fois plus

rapide que sa version 5.6. On peut noter aussi qu'il n'y a pas eu de version PHP 6.0, il est directement passé de la 5.0 à la 7.0. En réalité, la sortie de la version 6.0 était bien prévue dans le courant 2008. Néanmoins, la phase de développement de celle-ci a été chaotique avec beaucoup de problèmes. Ils ont préféré recommencer de zéro et sortir directement la version 7 fin 2015. (25)

Hypertext Preprocessor (php) est un langage dit « accessible pour les débutants ». Il est inspiré des langages C et Perl et il est open source. Enormément de sites internet utilisent encore le PHP bien qu'il soit de plus en plus considéré comme dépassé. C'est un langage de script utilisé côté serveur. Le principe est que le serveur interprète le code PHP et génère du code pouvant à son tour être interprété par un navigateur web. Il est directement imbriqué dans le code HTML. Cette façon de faire propose une manière simple de créer des contenus dynamiques.

Etant sans doute le langage le plus vieux encore largement utilisé dans le développement web, il contient quelques défauts. Le plus notoire et qui ressort le plus est le défaut de la sécurité. Il est assez connu que des sites en php qui n'ont pas mis en place des stratagèmes pour améliorer la sécurité possède des failles. Celles-ci sont par exemple l'injection de code SQL, des failles de session, des failles d'upload, … Toute personne utilisant php doit donc faire attention à ce type de problème.

3.1.4.6 Nodejs (javascript)

Nodejs (27) n'est pas un langage à proprement parler mais un environnement de bas niveau qui permet l'exécution de javascript coté serveur. Pour être précis le titre de cette section devrait se limiter à « javascript » si on ne voulait noter que le nom du langage en titre.

C'est Ryan Dahl qui a créé Nodejs en 2009 (26 & 28). L'idée lui est venue quand il a vu que sous Flickr, on ne voyait pas le pourcentage de téléchargement d'un fichier. Il était obligatoire de faire une requête au serveur pour connaitre ce pourcentage ce qui n'est pas très optimisé. Il a, alors, essayé de faire quelque chose de plus simple. Quelques temps plus tard Nodejs était créé. Depuis, le succès est grandissant jusqu'à toucher les plus grandes entreprises puisque Paypal, LinkedIn, Walmart et bien d'autres sont passés à Nodejs pour leur backend.

Nodejs est construit sur le moteur JavaScript V8. Il est single thread non bloquant. Non bloquant car lorsqu'un client fait une requête, il peut la lancer sans attendre le résultat. Si un deuxième client fait une autre requête, il est tout de suite disponible.

Un autre avantage de Nodejs réside dans toutes les bibliothèques créées par une communauté active pour faciliter la vie au développeur. En effet, quand on veut faire quelque chose en Nodejs, il est assez fréquent que d'autres ont également voulu faire la même chose et ont peut-être même créé une bibliothèque dédiée qu'ils ont partagée. Mais l'avantage principal de Nodejs réside dans le fait que c'est le langage javascript qui est utilisé. Comme le javascript est aussi utilisé du coté front end ce langage est déjà connu des développeurs.

3.1.5 Choix des différents langages

Après toutes les explications sur les différents langages, je peux faire un choix pour la plateforme. Comme nous avons vu, le choix du front end est déjà fait puisque les trois langages présentés fonctionnent ensemble. De plus, la plupart des front end des sites sur internet sont faits de cette manière-là. Les langages pour le front end seront donc html, css et javascript.

Pour le backend, le choix va être un peu plus compliqué puisque plusieurs ont été présentés. Il est évident, par contre, que tous les langages possibles n'ont pas été présentés et qu'une limite à dû être posée. J'ai essayé de garder les plus populaires bien qu'il est parfois difficile de mesurer la popularité d'un langage. De plus, certains développeurs pourraient avoir d'autres langages qu'ils considèrent comme meilleurs que ceux présentés. Cela dépend aussi beaucoup de l'affinité avec ce langage en question.

Afin de choisir au mieux, j'ai défini des critères importants.

Le premier critère est qu'il doit y avoir une communauté active et une documentation assez fournie. Ce critère permet de faire gagner du temps précieux durant la programmation. En effet, plus la communauté est active et plus la documentation est complète, plus vite on trouve des solutions aux différents problèmes et bugs qui peuvent survenir.

Le deuxième critère est l'extensibilité du langage. La plateforme qui va être développée devra surement être améliorée au fil du temps. En effet, il est rare qu'une plateforme de ce type n'évolue pas pendant dix ans et soit toujours utilisée / utilisable. Le but est d'avoir un langage qui nous permet une grande liberté quant à l'ajout de nouvelles fonctionnalités.

Grâce à ces critères, le choix est plus facile bien qu'il reste arbitraire. Mon choix s'est alors porté sur Nodejs. Effectivement, il remplit tous les critères. Il possède une communauté très active car il est de plus en plus utilisé ces dernières années. En plus de cela, des grandes entreprises tel que Netflix, Nasa, .. (40) lui ont fait confiance donc il y a peu de risque que le langage soit abandonné. Et enfin, il possède une grande extensibilité grâce notamment à la possibilité de l'intégrer avec une large variété d'outils.

Concernant les autres programmes, mon choix ne s'est pas posé sur eux pour différentes raisons. Le php est un langage vraiment très connu dans le web. Il possède cependant des défauts comme expliqué dans sa présentation. Ces défauts mais également l'engouement des grandes entreprises pour le Nodejs l'a écarté des choix. Ruby aurait pu être un candidat mais il est un peu moins reconnu que le Nodejs. Quant à Java et à Python, ils auraient aussi été de très bons candidats mais ils sont un peu moins utilisés que le Nodejs.

3.1.6 Application monopage

Dans les explications précédentes, on part du principe que l'outil à développer est un site internet « classique ». Les applications web traditionnelles sont composées de plusieurs pages. Lorsqu'un utilisateur veut afficher du contenu supplémentaire, une nouvelle page est

affichée. C'est en réalité à l'appui d'un bouton ou d'un lien que le serveur reçoit une requête et y répond en envoyant une nouvelle page. Cette méthode dite « classique » existe depuis le début de l'existence des sites internet. Seulement depuis quelques années, une autre méthode a émergé. C'est ce qu'on appelle la « Single page application ». Mais qu'est-ce que c'est ?

Une application monopage (SPA : Single page application) est un site ou une application web qui réécrit la page web dynamiquement. Cela permet des transitions plus rapides et plus harmonieuses qui donnent l'impression à l'utilisateur d'utiliser une application native. Il y également moins de problème de sécurité dû au fait qu'il n'est pas possible de sauter de page en changeant l'url.

Dans l'application monopage, soit tout le code est récupéré dans le navigateur en utilisant uniquement un seul chargement de page (lorsqu'on clique sur l'url du site et que l'onglet s'ouvre). Soit les ressources sont chargées dynamiquement en fonction de la navigation et des actions de l'utilisateur. Dans les deux cas, il n'y a pas de rafraichissement (hormis le premier chargement) de la page ni de changement de page. A contrario, dans la méthode traditionnelle, il y a des rafraichissements et des changements de pages.

Ce concept a gagné en popularité ces dernières années de par les différents frameworks qui aident à son développement et le fait que de plus en plus de sites internet l'utilisent. En réalité, bien qu'il est difficile de connaitre la date précise des origines de ce concept, on sait qu'il a été discuté avant 2003 (29). On sait aussi qu'un petit groupe de développeurs avaient expliqué la réalisation d'une application monopage dans un brevet en 2002.

Il est assez étonnant de voir qu'il est possible de communiquer avec un serveur sans pour autant recharger la page. Pourtant, c'est l'élément qui a permis au SPA d'être utilisé. Il existe, en réalité, plusieurs façons de faire.

- L'utilisation d'un framework ou d'une bibliothèque permet cela. Tous ceux cités précédemment tel que React, Angular JS, Vue.js mais aussi bien d'autres permettent de faire des SPA.
- Ajax qui permet de faire des requêtes asynchrones vers un serveur.
- Websockets qui permet une communication Client-serveur bidirectionnelle.

Etant donné que le modèle SPA est assez différent du modèle traditionnel, les navigateurs mais aussi les différents frameworks ont dû s'adapter au fur et à mesure de l'avancée. HTML5 (30) s'est adapté pour améliorer l'usage et la stabilité lors de l'utilisation du modèle. Différents frameworks web coté serveur spécialisés dans le modèle SPA sont sortis tels que derbyjs ou sails par exemple.

Le cycle de vie d'une SPA peut être résumé assez facilement. Dans un premier temps, le chargement initial charge entièrement la page que l'on voit à l'écran. A ce moment-là, soit tout est chargé coté client soit des petits fragments sont remplacés au fur et à mesure des

besoins. La deuxième solution évite un téléchargement excessif de fragments ou de fonctionnalités non utilisés au final. Dans les deux cas, la page que l'utilisateur voit à l'écran ne va jamais être changée par une autre mais ce sont des éléments à l'intérieur de celle-ci qui vont être modifiés, ajoutés, supprimés, …

Les SPA offrent différents avantages. Elles sont plus rapides et réactives. Elles permettent une expérience utilisateur linéaire. Mais elles comportent certains inconvénients tels que l'optimisation de référencement ou encore certains problèmes de rapidité lorsqu'il y a beaucoup d'éléments à modifier.

Bien que faire une application monopage pour la plateforme aurait été très pertinent, ce n'est pas vers ce choix que je me tourne. En effet, les avantages sont nombreux mais ceux d'utiliser un site multipage aussi. Le multipages permet une évolutivité presque infinie puisqu'on peut ajouter des pages et des informations sans se soucier de la performance. Ce qui n'est pas le cas d'une SPA où il faut limiter les fonctionnalités sans quoi la rapidité en patira.

3.2 BASE DE DONNÉES

3.2.1 Mise en contexte

Au fur et à mesure de l'avancée de l'informatique, les programmes sont devenus de plus en plus complexes et complets. Ils sont devenus capables de travailler avec un plus grand nombre de données. Le problème qui s'est assez vite posé est qu'au départ toutes les données étaient stockées en mémoire durant l'exécution du programme. C'était en fait des variables du programme. Une fois le programme terminé toutes ces informations étaient perdues. Il a fallu inventer une technique pour éviter de tout perdre lors de l'arrêt du programme.

Deux grandes solutions ont été trouvées, la première étant de sauvegarder les données dans un fichier qui se trouve localement sur l'ordinateur qui fait tourner le programme. Cette technique fonctionne très bien mais ne permet pas de partager les données avec plusieurs utilisateurs potentiels du programme en question. De plus, toutes les données sont perdues si on change d'ordinateur. Cette solution est plus souvent utilisée pour créer un prototype d'une application.

La deuxième technique consiste à utiliser des bases de données sur un serveur. Grâce à cela, on peut se connecter de n'importe quel ordinateur et récupérer les données enregistrées puisque la base de données ne se trouve pas localement sur l'ordinateur mais plutôt sur un serveur distant accessible à tous les utilisateurs du programme. Etant donné que la plateforme qui est développée doit rendre accessible des informations à plusieurs utilisateurs qui sont susceptibles de se connecter de partout dans le monde, je vais plus amplement me pencher sur la deuxième technique des bases de données sur un serveur distant.

Comme le nom l'indique, les bases de données sont des endroits où sont stockées les données. En fonction de l'application, le site web, le logiciel utilisé, il est possible que nous soyons amenés à stocker des données très différentes telles que des nombres, des images, des chaines de caractères,… Parfois, la base de données est l'outil le plus important du logiciel comme dans les sites d'e-commerce par exemple. La façon dont on stocke les informations ne doit pas être laissée au hasard et doit être réfléchie à l'avance.

Les bases de données sont un domaine, comme tous les autres domaines dans l'informatique, qui est en constante évolution. L'évolution ne s'est pas faite linéairement mais différents types sont apparus au fur et à mesure des années. Dans ces types, on peut citer par exemple les bases de données hiérarchiques, réseaux, relationnelles, …

Le type le plus connu et le plus utilisé est à l'heure actuelle le relationnel mais un nouveau type tend à se démarquer ces dernières années : Le NoSQL (Not only SQL). Dans cette partie, je vais entrer plus en profondeur dans ces deux types différents, montrer leurs avantages et inconvénients puis faire un choix sur ce qui va être utilisé dans la plateforme.

3.2.2 Relationnelle

Une base de données relationnelle est comme son nom l'indique, une base de données qui a pour particularité d'organiser les informations en tableau contenant des lignes et des colonnes. Il est possible alors de créer une ou plusieurs tables qui peuvent être en relation entre elles. Chaque table contient des colonnes qui sont les attributs et des lignes qui sont un ensemble de ces attributs et représentent un enregistrement de données. Le langage de programmation utilisé en majorité pour interagir avec ces tables est le langage SQL. Celui-ci nous permet d'interagir avec les données en ajoutant, retirant des données ou encore en faisant des jointures, des intersections, etc…

3.2.2.1 Histoire

Dans les années 70 avec l'expansion de l'informatique, il a vite été urgent de trouver une façon d'organiser les données sous une certaine structure. C'est alors que Edgar F. Codd, informaticien chez IBM, pose les bases du modèle de données relationnelles. Depuis lors, ce modèle est utilisé en nombre et est toujours le plus populaire aujourd'hui.

3.2.2.2 Le modèle relationnel

Comme mentionné dans l'introduction de cette partie, le modèle relationnel est composé de tables contenant chacune des lignes et des colonnes. Les colonnes sont des attributs et les lignes sont un ensemble de valeurs. Dans la plupart des cas, une base de données ne se limite pas à une seule table mais à un certain nombre de tables. L'ensemble de ces tables vont être reliées entre elles par des valeurs qu'elles contiennent. C'est le principe des clés primaires et des clés étrangères. La clé primaire sert à identifier de manière unique une ligne dans une table. La clé étrangère, de son côté, a pour objectif d'identifier une ou plusieurs colonnes d'une table comme référençant une ou plusieurs colonnes d'une autre table. Cette contrainte a pour but de garantir que les valeurs de chaque ligne de la table référençant existent dans la

table référencée. Elle permet donc d'établir des liens entre plusieurs tables et est un des grands principes du modèle relationnel. Il est aussi possible de faire des opérations sur les données enregistrées. Ces opérations sont des opérations d'algèbres relationnels. Il y a plusieurs catégories d'opérateurs provenant de l'algèbre relationnel qu'on peut utiliser sur les données stockées. Pour ne citer que quelques exemples, on peut faire des intersections, des jointures, du renommage, des projections, des sélections, …

3.2.2.3 Propriété ACID

Un des fondements des bases de données relationnelles est le respect des propriétés ACID dans chacune de leur transaction. Dans la langue des bases de données, une transaction est une opération sur les données. Les propriétés ACID ont pour but de garantir des opérations fiables. Ici, je ne parle que pour les bases de données relationnelles, d'autres bases de données ne respectent pas toujours ces propriétés comme par exemple le NoSQL.

Ces propriétés se découpent en 4 parties (31) :

- Atomicité : Permet d'assurer qu'une transaction est faite dans sa totalité ou qu'elle n'est pas faite du tout. Si une partie de la transaction ne peut pas être faite, les parties déjà opérées sont effacées pour respecter cette propriété.
- Cohérence : Permet de garantir que pour toutes les transactions, le système passe d'un état valide à un autre état valide.
- Isolation : Oblige chaque transaction à s'exécuter comme si elle était la seule sur le système et ne doit pas dépendre d'une autre transaction. Cela permet de garantir que deux transactions qui s'opèrent simultanément produisent le même résultat que si elles s'opéraient l'une à la suite de l'autre.
- Durabilité : Permet de garantir que si une transaction a été confirmée, elle reste enregistrée même dans le cas d'une panne quelconque immédiatement après.

On peut remarquer que grâce au respect de ses propriétés, on évite bon nombre de problèmes qui seraient vraiment dérangeants.

3.2.2.4 Avantages / Désavantages

On sent au travers de ce modèle de base de données une certaine robustesse surtout grâce au respect des propriétés ACID. Il y a un classement des données qui est très organisé, peu faillible et assez intuitif puisque ce sont des tableaux. Ce modèle possède aussi une grande capacité d'interrogation et garantit qu'aucun attribut n'est répété.

Par contre, il possède quand même quelques inconvénients. De gros problèmes de rapidité peuvent apparaitre lorsque la base de données comporte trop d'informations. Elle est aussi limitée dans son extension : il est assez difficile d'étendre horizontalement la base de données. On est contraint de l'étendre verticalement.

3.2.3 Non relationnelle (NoSQL)

Le NoSQL est un type de base de données qui a pour caractéristique de s'écarter du type bien connu dans les bases de données qui est le relationnel. Les frontières de la famille des bases de données NoSQL sont encore floues. Le terme se rattache évidemment aux caractéristiques techniques qui les différencient des bases de données « traditionnelles » qui sont les relationnelles. Mais il est aussi attaché aux différentes SGBD qui ont commencé à être connues aux alentours de 2010. Le fait que le NoSQL ait émergé en 2010 n'est pas une coïncidence puisque son principal avantage est de pouvoir traiter de grosses quantités de données. En effet, depuis quelques temps, les grosses sociétés ont besoin d'énormément de données sur différents points. Que ce soit pour enregistrer les différents faits et gestes de leurs utilisateurs ou pour stocker différentes données pour entrainer leurs intelligences artificielles , … Il y a de plus en plus de domaines où on a besoin d'énormément de données. Le problème est que le paradigme du relationnel n'est pas assez efficace pour traiter les nombreuses données. Par contre, le NoSQL fait assez bien le travail.

3.2.3.1 Les débuts du NoSQL

Ce sont les plus grosses sociétés amenées à travailler avec des volumes de données astronomiques qui ont été les premières à voir les limitations des SGBD relationnelles. En effet, celle-ci ont été conçues pour respecter les propriétés ACID et ont des problèmes d'extensibilité puisqu'elles sont conçues pour tourner sur un seul serveur. Cela pose problème quand le volume de données grossit énormément.

Les premières entreprises concernées ont été Google, Amazon, Facebook, … Elles ont très vite vu les limites de leurs bases de données et ont dû trouver des solutions pour les améliorer. Elles ont donc développé leur propre système de gestion de base de données qui peut fonctionner sur plusieurs serveurs et grâce à ça, étendre considérablement l'extensibilité. Ces systèmes propriétaires ont été le départ du NoSQL. Il avait pour but d'être scalable par architectures matérielles distribuées. Parmi ceux-ci, il y a eu BigTable pour Google, Dynamo pour Amazon, HBase pour Facebook, …

3.2.3.2 L'explosion du NoSQL

Grâce à ces entreprises qui ont développé ces bases de données et aux différents articles qui présentaient celles-ci, certaines personnes se sont penchées plus amplement sur le sujet et ont développé des débuts de projets open source. Le but au départ était « juste » de faire un système extensible mais qui ne respectait pas strictement les propriétés ACID.

En voyant le nombre de projets open source développés, un meeting regroupant les personnes qui travaillaient dessus a été fait en 2009. Ce meeting s'appelait NoSQL et c'est le nom qui est encore le plus répandu jusqu'à présent malgré les contestations. En effet, beaucoup de spécialistes du domaine se sont plaints en mentionnant que le nom était trompeur et devrait plutôt être « NoREL », pour Not only relationnel. C'est vraiment lors de ce meeting qui a regroupé plus de cent développeurs que le NoSQL a pris une ampleur importante.

3.2.3.3 Quels sont les buts de NOSQL

L'explication ci-dessus de la naissance du NoSQL résume assez bien son but principal mais est-ce le seul but ?

En réalité pas vraiment, bien qu'au départ et encore maintenant d'ailleurs, il permet de traiter des quantités de données gigantesques, d'autres avantages lui ont été trouvés par rapport au relationnel. En effet, cela a eu un succès assez énorme du côté des start-ups pour deux raisons principales.

La première est que ces entreprises du web naissantes n'ont pas forcément de gros moyens. Ils ne peuvent pas se permettre de faire des dépenses inutiles. Ils jouent clairement leur survie dans leurs premières années de vie. C'est ainsi que bon nombre d'entre elles ont préféré s'orienter vers cette nouvelle technologie de gestion de données au lieu d'acquérir des licences Oracle trop coûteuses pour ce genre d'entreprises.

La deuxième raison est l'extensibilité. Au début, une start up dans le web n'a pas forcément beaucoup de clients/utilisateurs. Mais il n'est pas rare de voir certaines de ces entreprises obtenir énormément de nouveaux clients en très peu de temps. Pour cela, ils ont besoin d'être très flexibles sur l'extensibilité de leur base de données. Le NoSQL leur donne une grande liberté de ce point de vue.

3.2.3.4 Plusieurs types de NoSQL

3.2.3.4.1 Clé – Valeur

La méthode de stockage clé – valeur est assez simple. Les données sont en fait stockées sous forme de paires avec d'un côté la clé et de l'autre la valeur. La clé est l'identifiant unique pour la valeur. Les clés et les valeurs peuvent être des objets simples comme des objets composés complexes. Il n'y a pas vraiment de restrictions importantes de ce point de vue. Le principal avantage est que la base de données est fortement divisible avec cette méthode ce qui permet une mise à l'échelle horizontale (avoir plusieurs serveurs pour une même base de données). C'est aussi assez simple à mettre en place.

3.2.3.4.2 Orienté documents

Ce type de base de données orienté document est conçu pour stocker et interroger des données sous forme d'un document JSON. Cela permet de manipuler les données en utilisant le même modèle que celui utilisé dans le code de l'application directement. Ce type fonctionne assez bien lorsqu'il faut stocker des catalogues ou encore des profils utilisateurs. Comme c'est un type assez souple cela permet d'évoluer avec l'application au fil du temps.

3.2.3.4.3 Orienté colonnes

La plupart des bases de données trie l'information en ligne. Ici, c'est l'opposé qui se passe. L'information est triée en colonnes. Ce type de base de données est très utile pour analyser de grandes quantités de données car elle a l'avantage d'être plus rapide que celle triée par

ligne dans ce domaine. C'est donc très utilisé dans le domaine de la recherche par exemple qui a besoin d'analyser des données en permanence.

3.2.3.4.4 Orienté graphes

Ce type permet de stocker des données en se basant sur la théorie des graphes. Il utilise plusieurs notions : nœuds, relations, propriétés. Ce stockage permet une représentation du monde réel dans une base de données. Il est utilisé pour les données des réseaux sociaux. C'est donc assez bien adapté pour des domaines complexes organisés en réseaux mais ça ne doit pas être utilisé si les données ne sont pas en forte relation.

3.2.3.5 Avantages et inconvénients

Le principal avantage du NoSQL se situe sur l'analyse d'énormément de données. Il est vraiment rapide et est d'une efficacité redoutable dans ce domaine car il a été inventé pour cela. L'extensibilité horizontale le rend encore plus pertinent pour les grandes quantités de données.

Par contre le fait qu'il s'écarte des propriétés ACID est plutôt un inconvénient. De plus, l'écriture des requêtes complexes peut parfois s'avérer difficile d'autant plus qu'il n'existe pas encore vraiment de langages d'interrogations standardisés pour celui-ci.

3.2.4 Choix entre Relationnel ou NoSQL

Un choix entre ces deux types de bases de données n'est pas facile et est crucial pour la suite du projet. En effet, une fois le choix fait et la plateforme lancée, il n'est pas facile de revenir en arrière puisque tout a été prévu pour un type de base de données. Il ne faut donc pas prendre ce choix trop à la légère. En regardant les avantages et inconvénients de chacun, il est néanmoins possible de s'orienter. La plateforme n'est pas censée utiliser énormément de données et n'est pas susceptible d'avoir besoin de plusieurs serveurs pour la faire tourner. L'avantage qu'offre le NoSQL sur sa facilité à gérer les quantités astronomiques de données ne devrait pas être utile pour ce projet. De plus, les bases de données relationnelles respectent les propriétés ACID qui assurent les bons fonctionnements d'une base de données contrairement au NoSQL qui s'écarte parfois de ces propriétés pour améliorer les performances. Le modèle relationnel est plus adapté à ce projet et c'est lui qui va être utilisé dans la suite. Il y a cependant une exception dans le stockage des données de ce site. Les documents déposés par les différents utilisateurs ne vont pas être stocker directement dans la base de données. En effet, ils seront téléchargés sur le serveur de la plateforme et la base de données stockera uniquement le chemin d'accès au fichier.

Bien que le type est choisi, un autre choix doit encore être fait : celui de la SGBDR. Ce choix aussi est assez important pour la suite car bien qu'ils ont tous la même fonction, ils ont chacun leurs spécificités à ne pas négliger.

3.2.5 SGBDR

3.2.5.1 Explication des différentes SGBDR

Pour maintenir et créer des bases de données relationnelles, nous avons besoin de logiciels qui s'en chargent. Nous les appelons des systèmes de gestion de base de données relationnelles (SGBDR). Il en existe énormément et ils ont parfois des caractéristiques différentes. Il faut bien réfléchir à la taille de notre base de données et aux évolutions qu'elle pourrait avoir à l'avenir pour bien choisir notre SGBDR. Il ne faut pas oublier que plus cette SGBDR est populaire plus il est susceptible d'y avoir de la documentation et des informations sur internet ce qui peut économiser beaucoup de temps dans certains cas. En plus de cela, si le projet doit être repris et maintenu à l'avenir il est bon d'utiliser la plus populaire. Il sera plus facile après de trouver des gens compétents.

3.2.5.1.1 MySQL

MySQL fait partie des plus connus. Il possède une édition communautaire gratuite qui est parfaite pour des petits projets (32). Il bénéficie d'une grande communauté, ce qui permet d'avoir beaucoup d'exemples et de documentations de son utilisation. Il est assez facile à prendre en main de par sa syntaxe simple et son apprentissage facile. Un développeur expérimenté n'est donc pas nécessaire pour développer une base de données sous MySQL. Par contre, il est à noter que MySQL ne permet pas une évolution facile et infinie. On le préfèrera donc dans des applications de petites tailles.

3.2.5.1.2 MariaDB

MariaDB est la petite sœur de MySQL au propre et au figuré puisque les deux sont issues du même créateur, Michael Widenius, qui a deux filles My et Maria (34). Venant du même créateur, les deux SGBDR se ressemblent un peu mais il y a quand même quelques différences. MariaDB (33) permet une grande sécurité car des fonctionnalités de cryptages ont été implémentées. De plus, elle est très performante grâce à certaines optimisations mises en place comme par exemple le fait que dès qu'une ligne est supprimée le système accède directement à l'espace libre. Néanmoins, le problème le plus important est que la communauté est encore grandissante. Il est donc parfois difficile d'obtenir des réponses aux questions qui posent problème.

3.2.5.1.3 Oracle

Logiciel propriétaire, Oracle a l'avantage de bénéficier d'un support technique et de beaucoup de documentations (35). C'est un moteur puissant qui permet de traiter une grosse quantité de données donc de grosses bases de données. Le problème est que son coût est élevé. Il possède une version gratuite mais elle est très limitée. Pour bénéficier des avantages, il est obligatoire de payer. De plus, bien souvent, il faut passer par des ingénieurs certifiés pour utiliser cet outil car il est assez difficile à prendre en main. Ce logiciel s'utilise quand on a beaucoup de données à conserver et quand on dispose du budget nécessaire.

3.2.5.1.4 PostgreSQL

PostgreSQL est open source et possède l'avantage d'avoir une communauté conséquente qui permet d'avoir une aide complète et surtout gratuite (36). Il permet également une évolution très facile, plus facile que MySQL par exemple. Le plus gros problème avec ce système de gestion de base de données est que l'on doit vérifier assez fréquemment si tout se passe bien. En effet, Postgre ne possède pas d'outil de révision. Postgre est assez populaire dans les institutions financières et les systèmes de télécommunication grâce ses puissantes capacités d'analyse et d'entreposage.

3.2.5.1.5 MSSQL

MSSQL offre beaucoup d'avantages. Il possède une offre très diversifiée ce qui permet de choisir correctement en fonction des fonctionnalités dont on a besoin (37). Il dispose aussi d'une assistance et d'une documentation très complète. Il est à noter toutefois qu'il cible plutôt les entreprises et ce parce que son service est payant et même assez cher. De plus il reste assez complexe à utiliser. Il vaut mieux l'éviter si le projet reste modeste. Il reste toutefois très intéressant quand on possède d'autres abonnements Microsoft en parallèle.

3.2.5.1.6 SQLite

SQLite est le moteur de base de données relationnel le plus utilisé. Il utilise le langage SQL et respecte les propriétés ACID. A la différence des autre SGBDR traditionnels tel que MySQL ou PostgreSQL, il n'utilise pas le schémas habituel Client-serveur. Il est directement intégré aux programmes qui l'utilisent. L'ensemble de la base de donnée est stockée dans un fichier indépendant de la plateforme.

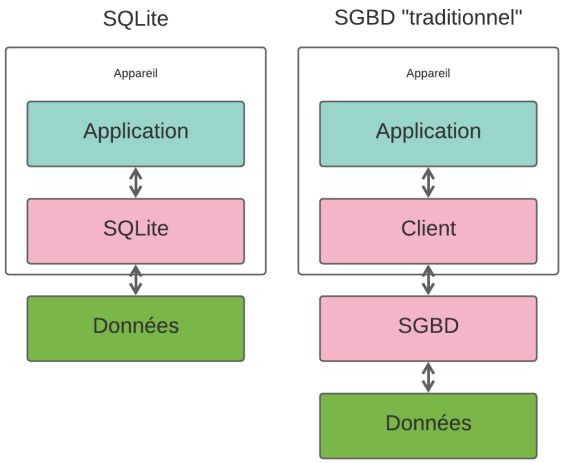

Figure 2 SQLite/SGBD "traditionnel"

La différence est assez flagrante sur le schéma de la Figure 2. A gauche se trouve le modèle qu'utilise SQLite. On voit qu'il est directement dans l'espace de l'application et qu'il communique avec des données qui se trouvent dans un fichier. A droite, on voit le modèle client-serveur qui est utilisé notamment par MySQL, Postgree, … On remarque que le SGBD est en dehors de l'espace de l'application.

C'est Dwayne Richard Hipp qui a créé SQLite. Il a fait le choix de mettre cette bibliothèque mais aussi son code source dans le domaine public. Cela permet l'utilisation de SQLite sans aucune restriction que ça soit pour des projets propriétaires ou open source.

SQLite est utilisé un peu dans tous appareils et existe en plusieurs milliards d'exemplaires. Pour ne citer qu'une liste exhaustive d'appareils dans lesquels SQLite est utilisé, on peut dire :

- Tous les appareils Android
- Tous les iPhone
- Chaque instance de Skype
- La plupart des systèmes multimédias automobiles

La liste pourrait continuer longtemps. Les chiffres sont difficiles à obtenir précisément mais il est assez probable que SQLite soit présent à plus de 1 billion d'exemplaires. (38)

Est-ce que SQLite est interchangeable avec un outil comme MySQL par exemple ?

La réponse est non. SQLite n'a pas pour but de résoudre les mêmes problèmes. Les différents moteur SQL qui existent priorisent l'évolutivité, la centralisation et le contrôle. Contrairement à SQLite qui lui permet de fournir un outil de stockage local et qui met plutôt l'accent sur l'économie, l'efficacité, l'indépendance et la simplicité.

Il est plutôt conseillé d'utiliser SQLite quand :

- On veut procurer une base de données à des appareils embarqués. Il est très efficace lorsque les appareils doivent fonctionner sans assistance humaine du fait qu'il n'y ait aucunement besoin d'administration.

- On désire un format de fichier d'application. Cela permet de stocker les données d'une application.

- On souhaite avoir une base de données pour un site internet à faible trafic. Si le site internet envoie des requêtes à faible fréquence à sa base de données SQLite est très efficace.

Il est plutôt déconseillé d'utiliser SQLite quand :

- On crée une application fortement orientée client-serveur. Si plusieurs clients sont susceptibles d'envoyer des requêtes SQL à la même base de données, SQLite peut avoir de la latence et des bugs. La règle d'or que donne SQLite lui-même c'est d'éviter de l'utiliser quand la même base de données sera accessible directement et simultanément à partir de nombreux ordinateurs.

- Une application possède un ensemble de données très volumineux. Une base de données SQLite est limitée en taille car toute la base de données est contenue dans un seul fichier et les systèmes de fichiers limitent très souvent leur taille.

- Il existe une concurrence élevée. SQLite autorise un nombre illimité de lectures mais il n'autorise qu'une seule écriture à la fois. La plupart du temps cela ne pose pas trop de problèmes car une queue se crée entre les écrivains. Mais pour les applications qui ne peuvent pas se permettre d'attendre ou d'avoir une queue entre les écrivains, il est vraiment déconseillé de prendre SQLite.

SQLite est très conscient de ses défauts et de ses qualités. Son site web propose une page entière pour aider les développeurs. Cette page précise quand il vaut mieux éviter SQLite et dans quelles circonstances il est moins efficace que les autres SGBDR connus tels que MySQL, PostgreSQL, et bien d'autres.

Dans notre cas précis, SQLite n'est pas une option qui va être retenue. La plateforme nécessite des informations centralisées et la possibilité à plusieurs utilisateurs d'écrire en même temps dans la base de donnée. Or, dans ces conditions, SQLite est déconseillé car il peut y avoir certains bugs (39).

3.2.5.2 Comparatif de certaines SGBDR

Maintenant que les principaux SGBDR ont été résumés, un tableau comparatif de tous va être présenté pour avoir une idée plus claire et pouvoir poser le choix sur lequel va être utilisé pour cette plateforme.

Nom	MySQL	MariaDB	Oracle	PostgreSQL	MSSQL
Etat du projet	Développement permanent	Développement permanent	Développement permanent	Développement permanent	Développement permanent
Type	Relationnel	Relationnel	Relationnel	Relationnel	Relationnel
Installation et facilité de maintenance	Simple	Simple	Nécessite des compétences particulières	Difficultés moyennes	Complète
Limites matérielles	Aucune	Aucune	1GO de RAM/1 coeur	Aucune	4 Go de RAM / 4 coeurs
Support, Documentation	Enormément	Peu	Beaucoup	Moyen	Beaucoup

Capacité IN MEMORY	Oui	Oui	Oui	Non	Oui
Licence	GPL	GPL	Propriétaire	BSD	Propriétaire

Figure 3 Comparaison des différentes SGBDR

3.2.5.3 Choix du SGBDR

Ce dont je ne parle pas forcément dans tout ce comparatif et qui a aussi une grande importance dans un choix comme celui-là c'est l'expérience qu'a déjà le développeur dans une de ces technologies. En effet, même une seule expérience peut déjà faire gagner un temps assez conséquent au développeur de l'application. Un temps qui n'est pas négligeable puisqu'il pourra le réinvestir dans une autre facette de la plateforme et potentiellement aller plus loin dans le développement.

Le choix n'est pas facile pour une raison simple c'est qu'il n'y a pas vraiment de mauvais choix. En effet, on peut voir un avantage dans un système qu'il n'y a pas dans un autre mais ce n'est que minime. Le choix est alors arbitraire. J'ai néanmoins décidé d'utiliser des critères assez précis pour m'aider. Les quatre critères principaux sont la facilité d'utilisation, la documentation mise à disposition, le fait qu'il soit gratuit au moins pour l'utilisation que je vais en faire et mon affinité avec le SGBDR en question.

Je me suis tourné vers le MySQL. Il a l'avantage d'être très populaire et permet de faire pas mal de choses gratuitement. J'ai aussi pensé à l'évolution que pourrait avoir la plateforme. En choisissant MySQL il sera assez facile de trouver un développeur qui s'y connait assez bien. Il est facile dans sa mise en place et dans son utilisation. De plus j'ai déjà une petite expérience avec MySQL. Ce SGBDR est très puissant et stable, il est très utilisé et maintenu avec des mises à jour régulières, beaucoup de sites internet l'utilisent donc normalement il n'y aura pas de mauvaises surprises.

3.3 CONCLUSION

Suite aux explications fournies dans les différents points précédents, deux choix sont ressortis. Nodejs donc javascript pour le langage web utilisé coté serveur. Du coté frontend, le choix est html, css et javascript. Pour aider à l'adaptabilité sur les différentes tailles d'écran, le framework Bootstrap va être utilisé. Il facilite beaucoup le travail et permet d'avoir un design uni. Afin de permettre facilement l'insertion de variables dans les différentes pages, le moteur de template Twig va être utilisé.

Coté base de données, le SGBDR MySQL va être pris. Il s'agit d'une valeur sûre avec une grande communauté derrière. Il n'y aura normalement pas de mauvaises surprises à l'utiliser.

Tous ces choix ont été faits le plus soigneusement possible et expliqués dans les sections précédentes. Ils ont pour but d'obtenir une plateforme robuste et efficace. Ils ont aussi été pensés avec l'idée d'une extensibilité de la plateforme dans le futur.

Il est évident que ces choix ont un coté arbitraire. Bien qu'ils remplissent les critères mentionnés, d'autres auraient pu faire une autre sélection. Tous les outils présentés ont une grande pertinence et sont de qualité.

Chapitre 4: DÉVELOPPEMENT

Lors de la conception d'une telle plateforme, il est important de commencer par une analyse complète avant de directement s'occuper du code. Ce chapitre va montrer pour commencer l'analyse du site internet. Que ce soit pour l'architecture du code ou pour la base de données. Cette analyse permet de s'attaquer à la partie programmation qui est le deuxième point de ce chapitre. Cette section présente certains fragments de codes de cette plateforme.

4.1 ANALYSE

Cette partie présente l'analyse de la programmation des pages du site en général, du serveur et de la base de données. C'est une étape obligatoire au développement de toute plateforme.

4.1.1 Quel type de plateforme

Pour commencer l'analyse, il faut d'abord déterminer pourquoi notre plateforme va être un site internet plutôt qu'une application mobile ou une application de bureau. En effet, la question peut paraître basique mais c'est la première à se poser.

Une application de bureau permet de réaliser une tâche directement depuis son ordinateur. Il faut l'installer sur son ordinateur pour pouvoir l'utiliser. Par exemple, un éditeur de texte, un navigateur web ou même un jeu vidéo sont tous des applications de bureau.

Un site internet est un ensemble de pages et de données accessibles par une adresse web. Habituellement, on utilise un navigateur web afin de consulter ce site.

Une application mobile est une application développée pour un appareil électronique mobile comme un téléphone portable, une tablette, … Cela ressemble à une application de bureau pour appareil électronique portable. La plupart sont distribué grâce à des plateformes telles que l'app Store, Google play, …

Grâce aux différentes définitions, il est possible de comprendre pourquoi notre choix s'est porté sur un site internet. Dans les caractéristiques, il est mentionné qu'il faut que les utilisateurs puissent utilisés la plateforme depuis n'importe quel appareil. L'application mobile ainsi que l'application de bureau sont donc tout de suite écartées. En effet, la première ne peut pas être utilisée sur un ordinateur et la deuxième ne peut pas être utilisée sur un appareil électronique mobile.

Le site internet, quant à lui, est accessible grâce à un navigateur web. La plupart des ordinateurs et des appareils électroniques mobiles possèdent ces navigateurs. Un site peut

alors être accessible à partir de toutes sortes d'appareils. Il va, par contre, falloir porter une attention particulière sur l'adaptabilité du site afin qu'il s'adapte aux différentes tailles d'écrans. En effet, la taille d'un écran d'ordinateur est différente de la taille d'écran d'une tablette ou d'un téléphone.

4.1.2 Analyse/Fonctions du site internet

Pour débuter dans l'analyse des différents points de la construction de ce site internet, il faut avant tout déterminer ces différents objectifs. Dans le chapitre 2, le début de ce travail a été réalisé grâce au use case. Il a permis de déterminer plusieurs étapes que l'utilisateur doit réaliser pour utiliser la plateforme.

- S'inscrire
  Afin de pouvoir identifier l'utilisateur, il doit pouvoir s'inscrire. Cela permet de savoir qui crée un projet ou qui écrit une critique, … Pour implémenter la fonctionnalité d'inscription, il y a besoin de plusieurs choses.
  Tout d'abord, une page qui permet à l'utilisateur de rentrer ses informations personnelles (au minimum un nom d'utilisateur et un mot de passe). Ensuite, une table dédiée aux utilisateurs dans la base de données. Elle a pour objectif d'enregistrer en un seul endroit tous les utilisateurs afin de se souvenir d'eux. Enfin, il faut une page de connexion. Evidemment, il semble logique qu'une fois inscrit les utilisateurs puissent utiliser leur compte pour se connecter ultérieurement. La page de connexion servira à cela. Dès que l'utilisateur veut utiliser la plateforme, il entre ses identifiants. Le serveur, quant à lui, vérifiera à ce moment-là si les identifiants rentrés correspondent à des identifiants déjà enregistrés dans le passé.

- Choisir une instance
  Cela permet de spécialiser le site dans un domaine précis. Tous les domaines n'ont pas besoin exactement des mêmes fonctionnalités, filtres, … En choisissant une instance de site, les éléments se spécifient dans un domaine précis. Par exemple, une gestion de rapports pour des conférences scientifiques ne demande pas exactement les mêmes fonctionnalités qu'une gestion de rapports pour un conseil communal. Lors d'une conférence, il suffira d'aller sur l'instance du site qui s'occupe plus précisément de celle-ci.

- Créer un projet
  L'intérêt de la plateforme étant de faire la gestion de rapports, il faut d'abord qu'un projet soit créé pour ensuite déposer les rapports à l'intérieur. Pour cela, il faut que les utilisateurs puissent avoir une page qui permette de créer un projet. Il faut aussi que les utilisateurs aient la possibilité de voir tous les projets dans lesquels ils participent. Ces deux fonctionnalités nécessitent une page dédiée sur la plateforme. Pour pouvoir enregistrer et consulter l'ensemble de ces projets, la base de données doit comporter une table supplémentaire. Elle contiendra les informations des différents projets.

- Enregistrer différents utilisateurs dans le projet
  Le but des projets est de regrouper plusieurs utilisateurs afin de discuter des différents rapports postés. Pour ce faire, il faut que le créateur d'un projet ait la possibilité d'ajouter des utilisateurs à son projet. Pour cela, lors de la création, des champs supplémentaires sont disponibles sur la page d'enregistrement d'un projet. Ces informations sont aussi enregistrées dans la base de données. Cela permet de savoir qui participe à quel projet.

- Déposer un rapport
  Une fois le projet créé, les utilisateurs doivent être capables d'ajouter un rapport à celui-ci. Ils doivent avoir une page d'enregistrement de rapports qui leur permet de les ajouter avec des informations supplémentaires. Il doit également y avoir une table dans la base de données où sont stockés les rapports et les différentes informations les concernant. La petite subtilité réside dans le fait que le rapport ne va pas être enregistré en brut dans la base de données. Il va être téléchargé dans un emplacement spécifique sur le serveur et la base de données contiendra le chemin d'accès de celui-ci. Une page devra également être créée pour voir tous les rapports déjà déposés.

- Faire une review d'un rapport
  Un des objectifs de cette plateforme est d'avoir la possibilité de critiquer et de noter les rapports déposés. Cette partie permet de le faire mais cela implique plusieurs choses. Premièrement, il est important de voir les différents rapports déposés. Comme expliqué ci-dessus, une page récapitulative de tous les rapports déposés doit être créée. Il faut également donner la possibilité aux personnes de consulter le document sur leur appareil. Un bouton pour télécharger le document devra donc être présent. Ensuite, une page pour ajouter une critique à ce document doit être créée. Les notes sont enregistrées dans une nouvelle table de la base de données. Une page supplémentaire existe aussi pour consulter les différentes critiques qu'a reçues un rapport en particulier.

- Voir les review d'un rapport
  Grâce à la base de données qui contient toutes les notes et commentaires pour chaque document, il est possible de remplir la page récapitulative des critiques reçues par un rapport.

Grâce aux différentes fonctionnalités de cette plateforme, il est possible de dégager les différentes pages nécessaires à sa construction ainsi que les tables de la base de données. Ces pages doivent bien sûr être agencées de façon logique en fonction de l'utilisation. Pour prendre un exemple, il est inutile pour l'utilisateur d'avoir la page pour noter et commenter un rapport si aucun n'a encore été déposé. Il est nécessaire d'avoir une certaine logique derrière.

Il ne faut pas oublier non plus qu'il y a une certaine liberté quant au contenu des différentes pages et que plusieurs fonctionnalités décrites peuvent en réalité n'appartenir qu'à une seule page dans la plateforme. On peut imaginer par exemple que dans la page qui liste les différents rapports du projet, il soit possible d'ajouter également un rapport. Ce sont deux opérations qui sont susceptibles d'être sur une même page en fonction de l'ergonomie et de l'intuitivité que cela offre.

L'énumération des différentes opérations disponibles avec cette plateforme offre quoi qu'il en soit une excellente base de travail. Cela permet de savoir ce qu'il doit apparaitre dans la plateforme mais également dans quel ordre cela doit apparaitre. Voici alors un premier schéma sur lequel s'appuyer lors de la création du site.

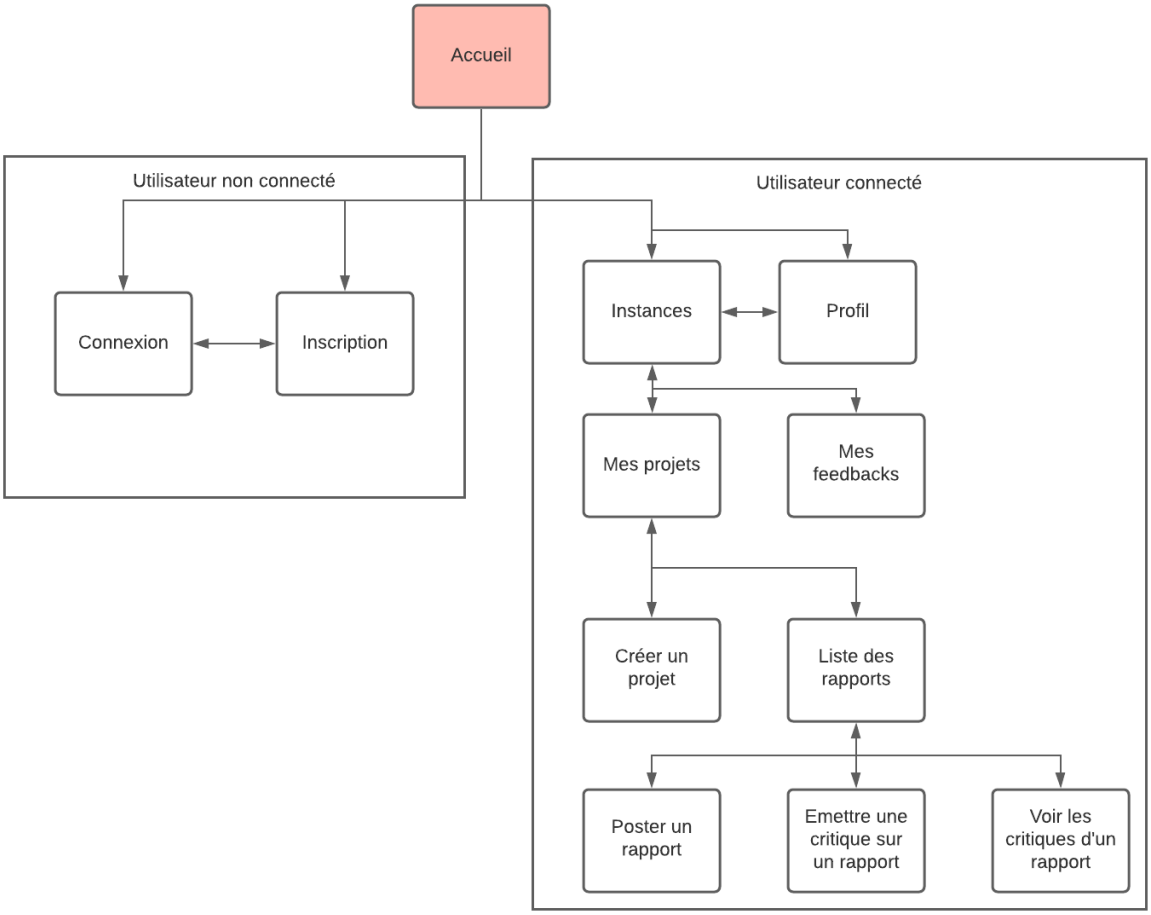

Figure 4 schéma des différentes pages du site internet

Ce schéma permet de voir toutes les pages qui vont devoir être créées ainsi que les liaisons entre celles-ci. Il est à noter qu'il est toujours possible et à tout moment de revenir sur la page d'accueil. Celle-ci permet de se rendre sur la page d'inscription ou de connexion si l'utilisateur n'est pas connecté. Par contre, si l'utilisateur est connecté, la page d'accueil permet de choisir l'instance du site désiré ou aller sur son profil.

4.1.3 Architecture MVC

Un des premiers problèmes rencontréss lors de la création d'un site internet est comment gérer les différents fichiers créés. En effet, il faut savoir gérer les fichiers pour le frontend et le backend. Dans l'absolu, tous ces fichiers peuvent être dans le même dossier les uns à côté des autres. Le problème avec cette méthode est qu'il n'y a pas d'organisation particulière. La personne qui programme ou qui reprend le projet en cours de route va mettre beaucoup de temps pour comprendre le rôle des différents fichiers et les liens entre eux.

Le projet a besoin d'être un minimum organisé. Il est important de créer différents sous dossiers pour séparer certains groupes de fichiers. La question qui se pose est comment faire pour structurer les fichiers du site correctement ? On peut bien sûr s'imaginer qu'au vu du nombre de sites internet disponibles, d'autres se sont également posé la question.

En informatique, lorsqu'un problème est rencontré par beaucoup de développeurs, il est fréquent qu'une série de bonnes pratiques soit mise en place pour créer au finale un design pattern.

Un design pattern très connu en informatique est le MVC. Il veut dire Modèle – Vue – Contrôleur et est destiné aux interfaces graphiques (il est très populaire dans le web). Le but de cette architecture est d'aider à organiser le code source. Il permet de savoir quels sont les fichiers à créer et de définir leur rôle (41). Le MVC sépare donc les fichiers en trois parties.

- **Modèle** : Les fichiers dans cette partie permettent de gérer les données du site. Ils vont récupérer les données dans la base de données et vont les organiser de façon à ce qu'elles puissent être traitées par la partie contrôleur. On retrouve donc les requêtes SQL.

- **Vue** : Cette partie est là pour l'affichage. Elle est composée principalement des fichiers HTML qui peuvent eux-mêmes être composés de moteurs de templates qui seront expliqués dans la suite. Cette partie ne fait donc que peu de calculs mais récupère des variables pour savoir ce qu'il faut afficher à l'utilisateur.

- **Contrôleur** : Cette partie est plutôt la logique du code. On peut considérer que le contrôleur est l'intermédiaire entre le modèle et la vue. Il demande au modèle les données, les traite, les analyse et prend des décisions. Il renvoie ensuite à la vue ce qu'il faut afficher. Il sert aussi à la gestion des droits d'utilisateurs, etc …

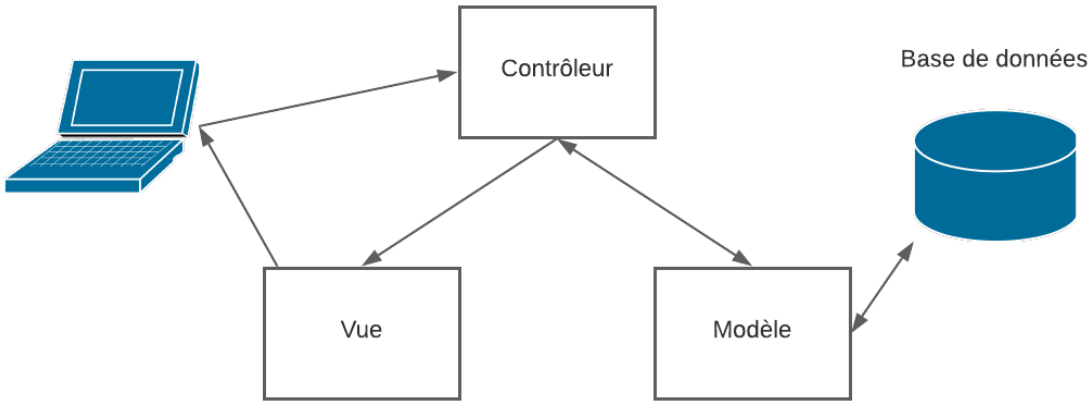

Figure 5 Schéma architecture MVC

Cette architecture permet d'organiser son code d'une façon très efficace. Une variante de ce modèle est également assez connue. Celle-ci ajoute un module routes afin de gérer le trafic des requêtes utilisateur.

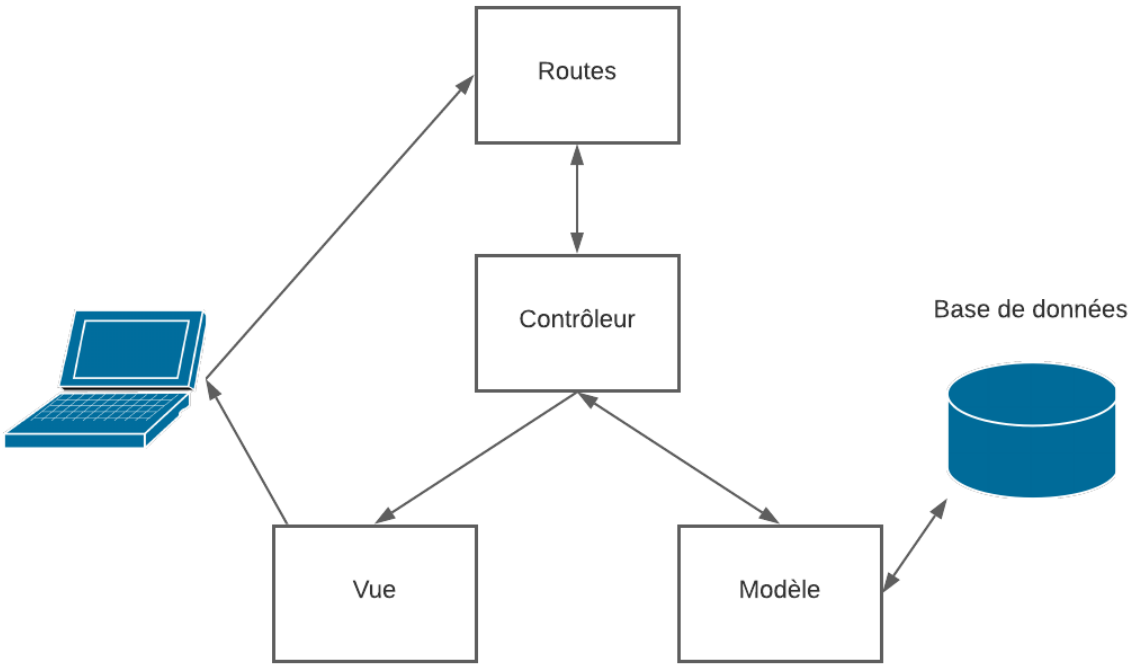

Figure 6 Schéma architecture MVC + Routes

Sur cette image, une partie « Routes » est ajoutée. Le chemin de la requête est divisé en cinq étapes distinctes.

- Etape 1 : Une requête est envoyée par un utilisateur. Exemple : L'utilisateur appuie sur un bouton qui doit afficher une nouvelle page.
- Etape 2 : La partie « routes » va diriger vers la bonne partie du contrôleur.
- Etape 3 : Le contrôleur, si besoin, va interagir avec le(s) modèle(s).
- Etape 4 : Le contrôleur avec les données dont il a besoin va invoquer la bonne vue pour l'utilisateur.
- Etape 5 : La vue est fournie à l'utilisateur.

Le fichier routes clarifie bien le travail lorsqu'il commence à y avoir beaucoup de pages différentes sur un site. Cela permet de s'y retrouver et de ne pas avoir un fichier de contrôleur trop volumineux

L'architecture MVC décrite ici comporte deux grands avantages. Le premier, comme expliqué, est qu'elle permet de structurer le projet en dossiers et en fichiers distincts ce qui facilite la compréhension. Le deuxième est que cette architecture est très connue et très utilisée. Beaucoup de développeurs la connaissent ce qui permet à un développeur qui reprendrait le projet de comprendre plus rapidement le code.

4.1.4 Dossier d'application

L'explication de l'architecture MVC décrite juste au-dessus permet d'y voir plus clair sur les dossiers présents dans le projet. Néanmoins, cela ne suffit pas. Lors de l'introduction des outils du frontend dans le chapitre 3, il est mentionné qu'il est tout à fait possible de mettre les trois langages de programmation dans le même fichier. En effet, dans un fichier avec extension .html, il est tout à fait possible d'insérer du CSS et du javascript en plus du code HTML.

Le problème qui se pose assez vite lorsque cette méthode est utilisée, c'est que le code dans les fichiers devient très rapidement illisible et désordonné. Ces deux caractéristiques ne sont vraiment pas souhaitables quand il s'agit de code. Pour faciliter la compréhension, les trois langages sont écrits dans des fichiers séparés. Les fichiers .html contiennent le code HTML avec des balises qui permettent de faire le lien avec les deux autres types de fichiers utilisés qui sont les fichiers .css (qui contiennent le code css) et le .js (qui contient le code javascript).

Dans le cas du projet de cette plateforme, il y a un dossier « public » qui contient tous les fichiers statiques publics. Dans ce dossier se trouve plusieurs sous dossiers tel que « css », « js » et « images » qui eux contiennent respectivement les fichiers css, javascript et images.

Dans l'analyse de la plateforme faite durant ce chapitre, une autre problématique est mentionnée, le fait que la base de données ne va stocker que le chemin vers où est enregistré le fichier. Il faut donc stocker le fichier quelque part pour obtenir son chemin. Pour ce faire, les rapports sont stockés dans un dossier nommé « uploads ». Celui-ci contiendra tous les rapports téléversés sur la plateforme ce qui permet de faciliter la compréhension et d'enregistrer le chemin dans la base de données.

4.1.5 Analyse de la base de données

Lors de l'analyse de la plateforme, certaines tables devant apparaitre dans la base de données ont été mises en évidence. En effet, ces tables permettent de représenter et d'enregistrer les données utiles au bon fonctionnement de la plateforme.

Afin de faire une analyse complète de la base de données, il faut commencer par schématiser celle-ci. Plusieurs schémas vont être nécessaires afin de procéder méthodiquement. Le premier à produire est un schéma MCD (Modèle conceptuel des données) qui est une analyse conceptuelle de la base de données.

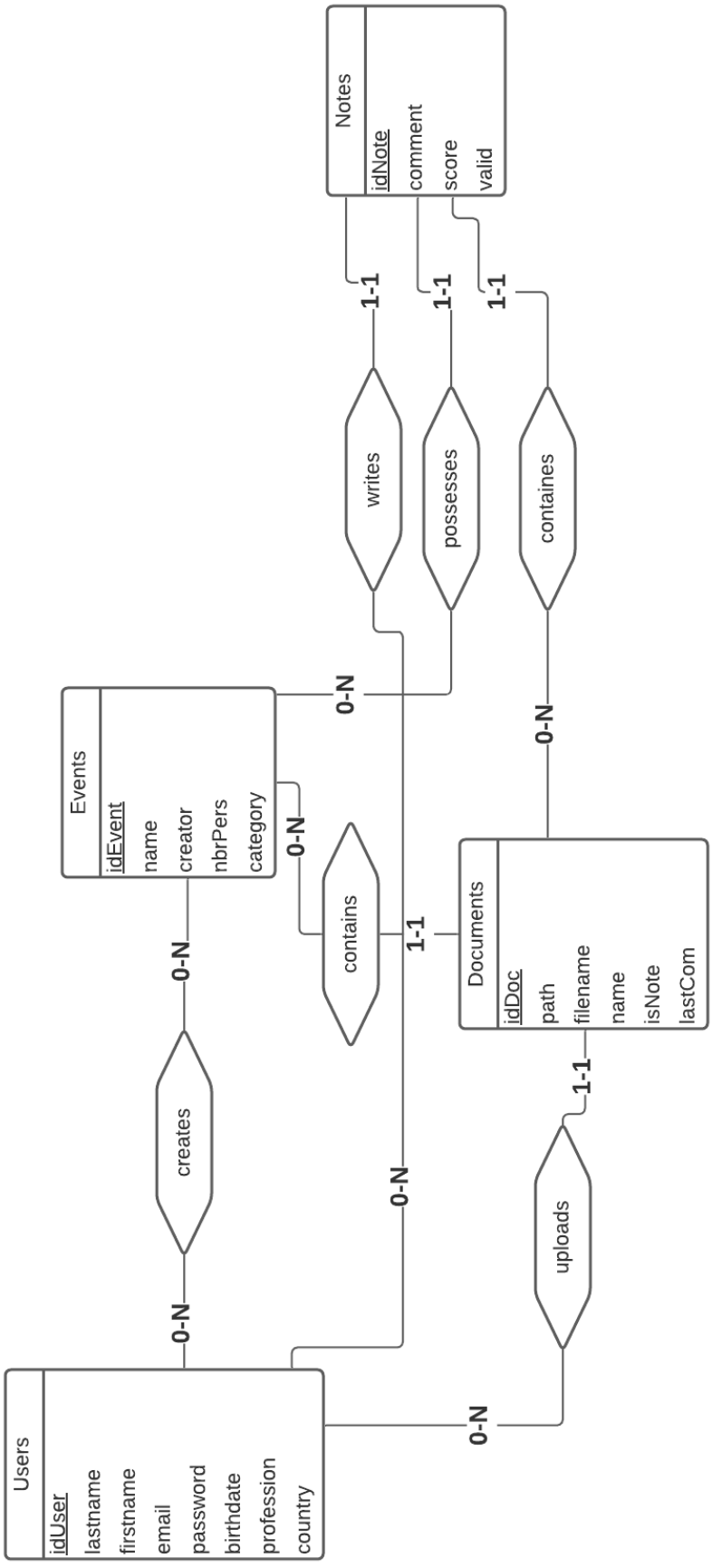

Figure 7 Schéma MCD

Ce schéma MCD est composé de plusieurs éléments. Tout d'abord, on remarque différentes tables telles que celles-ci :

Les entités :

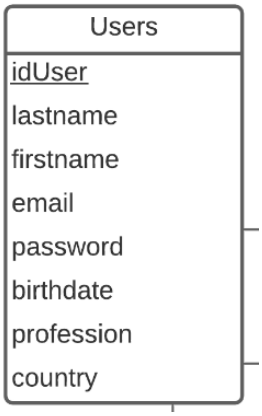

Figure 8 Exemple de table : Table Users

Ces tables représentent un ensemble de données. Ici, la table a pour nom « Users ». Les attributs présents sous le nom de la table (lastname, firstname, email, …) correspondent aux colonnes. Dans chaque table, il doit y avoir un identifiant. Celui-ci doit être unique pour chaque ligne de la table. Dans ce cas précis, l'identifiant est l'attribut souligné c'est-à-dire idUser. Il faut faire attention que parfois l'identifiant correspond à un ensemble de plusieurs attributs.

Les associations :

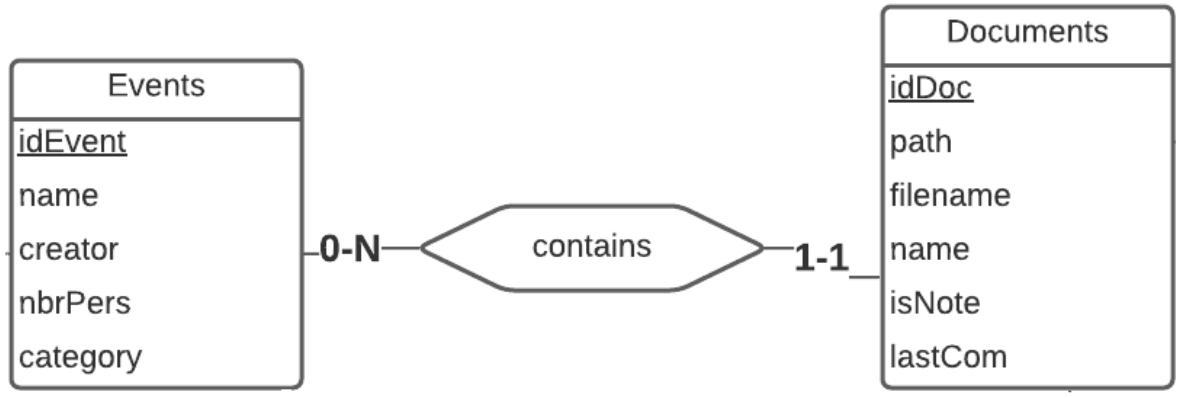

Figure 9 Exemple d'association

Ici, il y a une association entre la table « Events » et la table « Documents ». Celle-ci signifie qu'un projet peut contenir zéro ou plusieurs documents.

Les différentes tables :

- Users : Permet de représenter les utilisateurs inscrits sur la plateforme avec leurs différentes informations telles que le mail, le mot de passe, le nom, …
- Events : Permet de représenter les différents projets créés par les utilisateurs.
- Documents : Représente les rapports que les utilisateurs ont déposés dans les différents projets ouverts.
- Notes : Représente les différentes critiques reçues pour un rapport et qui ont été faites par des utilisateurs.

Une remarque sur ce schéma est que par simplicité et pour éviter de référencer une entité par un groupement d'attributs (dans le cas d'identifiant composés), des identifiants techniques ont été intégrés.

Maintenant, il faut transformer le schéma MCD, selon les règles de transformation(42), en un schéma MCD équivalent.

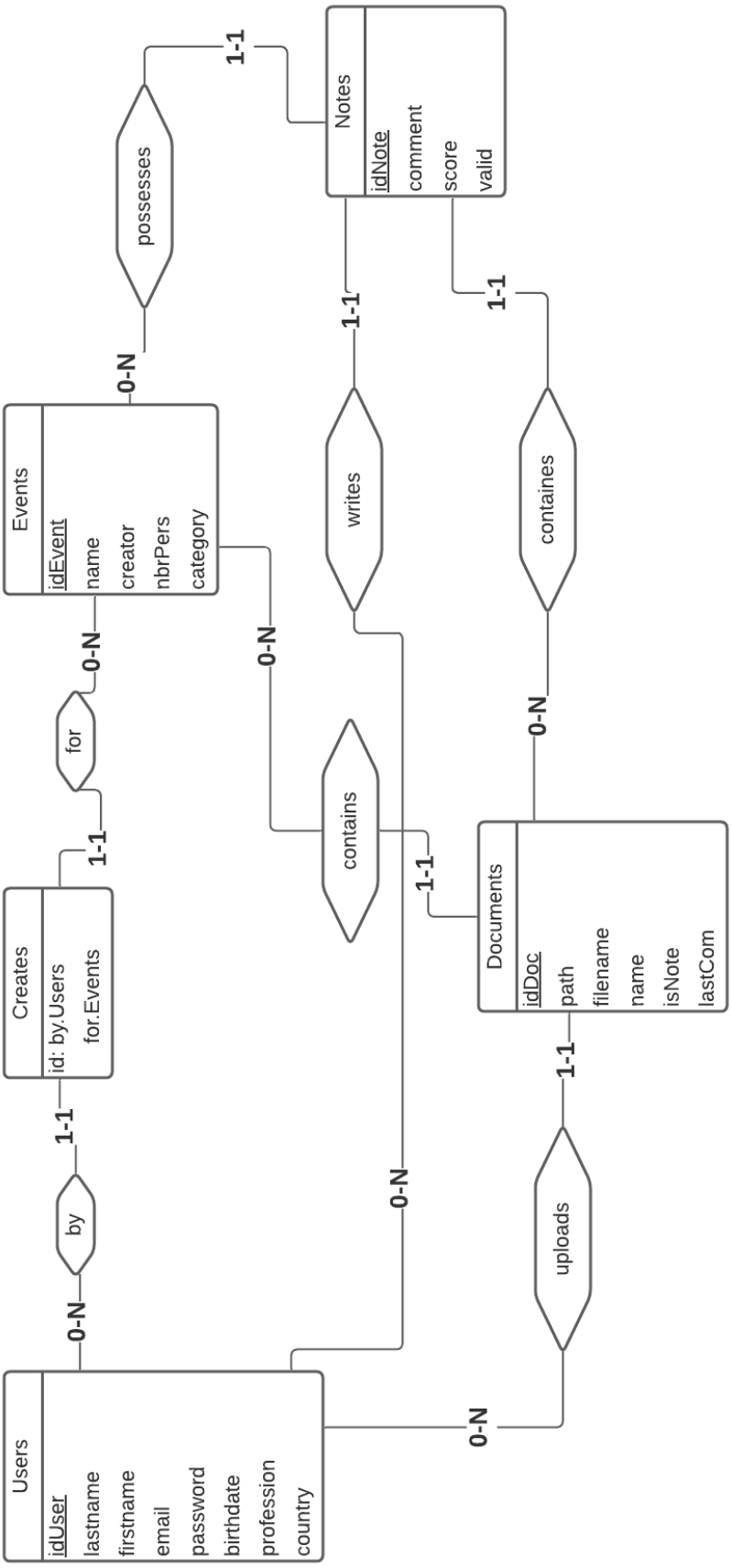

Figure 10 Schéma MCD équivalent

La principale transformation faite dans ce nouveau schéma est entre la table Users et Events. En effet, une nouvelle table s'est ajoutée entre les deux pour éviter une relation 0-N de part et d'autre.

Après cette analyse, il est maintenant possible de créer un schéma MLD (Modèle logique de données)

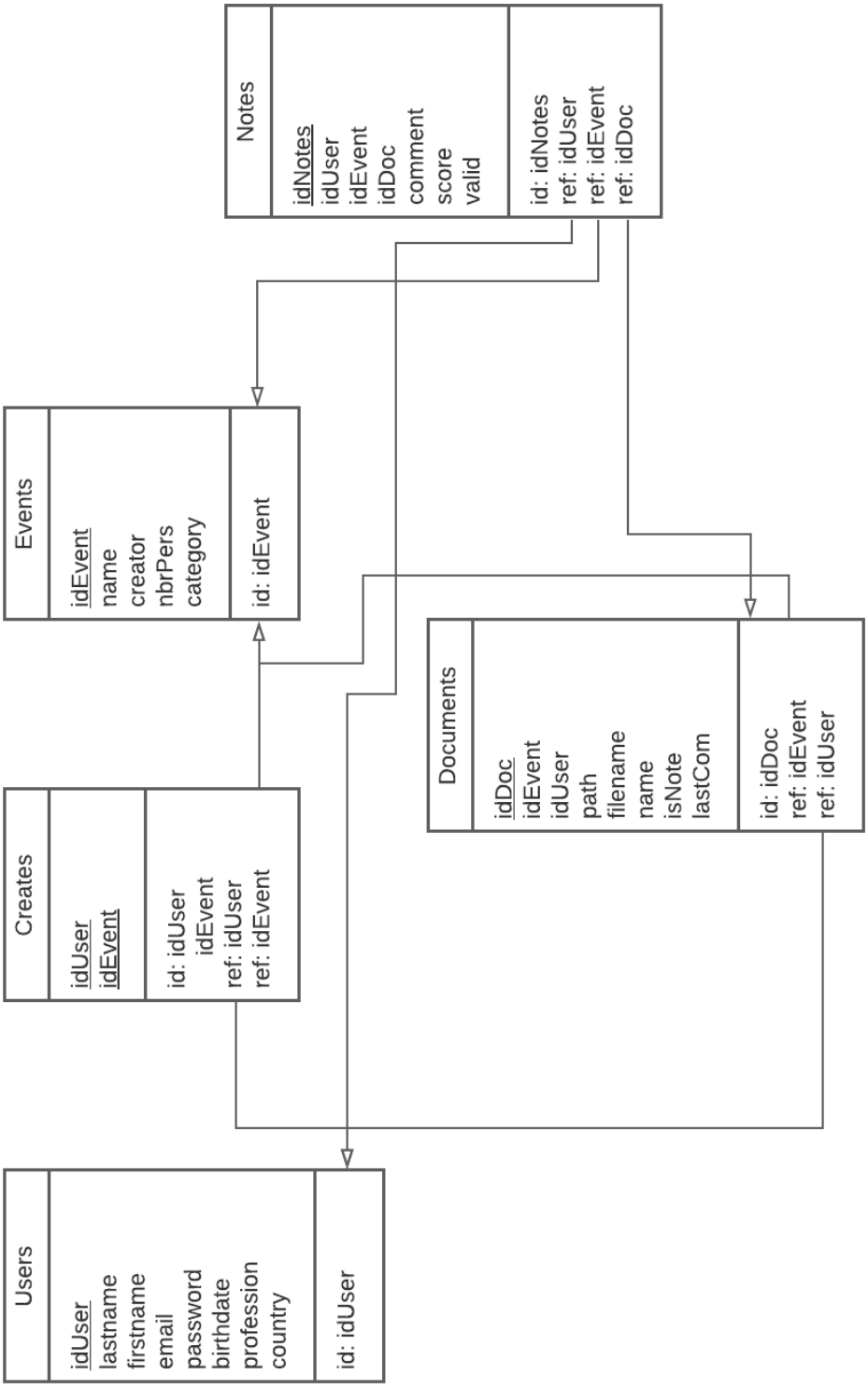

Figure 11 Schéma MLD

Ce dernier permet maintenant de créer la base de données. Avant cela, un petit mot d'explication est nécessaire pour certaines parties.

On peut remarquer qu'à la fin de certaines tables, il y a un mot clé ref. Ce mot est utilisé pour désigner une clé étrangère. Un autre mot est également employé, c'est « id » et il permet de

nommer les identifiants (clé primaire) de chaque table. Celui-ci apparait dans chaque table et doit être unique pour chaque ligne de celle-ci.

4.1.6 Résumé de l'analyse

L'analyse de la programmation du site internet et de la base de données étant réalisée, il est possible d'en faire un petit résumé pour récapituler les différents points.

Pour le langage web, l'architecture MVC est choisie afin d'organiser notre code et nos fichiers. Cela permet d'avoir une base pendant toute la durée du développement. Cela permet également d'avoir une organisation identique à beaucoup de sites web ce qui facilite la maintenance.

Pour la base de données, les différentes tables utiles pour gérer les données de cette plateforme ont été énoncées. Les associations entre ces tables sont aussi mises en évidence grâce aux différents schémas présentés ci avant.

4.2 PROGRAMMATION

Il est maintenant temps de passer à la partie programmation de cette plateforme.

Cette partie est précisément consacrée à l'explication de la programmation du site internet. Avant de débuter, certaines remarques doivent être énoncées.

Premièrement, la plateforme créée se veut être une preuve de concept. Elle ne se veut pas du tout être une plateforme complète, déjà prête à l'emploi. Le site présenté est plutôt basique et regroupe les fonctionnalités nécessaires au fonctionnement minimal. Certains éléments ont donc été omis afin de se concentrer sur l'essentiel.

Deuxièmement, l'explication réalisée dans les pages ci-dessous n'est pas exhaustive. En effet, certains passages, certains éléments répétitifs ou certaines fonctionnalités ne sont volontairement pas expliquées. Tout ceci dans le but de clarifier au maximum l'explication et de ne pas rendre cette partie trop longue inutilement.

4.2.1 Mise en place

Grâce à l'étude accomplie sur les différents outils et langages de programmation dans le chapitre précédent, il a été possible de dégager ceux qui vont être utilisés pour créer ce site. Le choix s'est porté sur le html, css et javascript pour le frontend et pour du nodejs (javascript) pour le backend.

Avec ces différents langages, il est possible de créer des sites internet des plus basiques aux plus compliqués. Avant d'utiliser ces langages, il faut mettre en place un environnement de travail.

Tout d'abord, il est important d'avoir un IDE (integreted development environment) ou EDI (Environnement de développement intégré) en français. C'est un logiciel qui permet de créer, de modifier, d'enregistrer, … des fichiers textes. Il en existe une multitude avec plus ou moins de fonctionnalités dans chacun d'entre eux.

Le choix d'un IDE est bien sûr fait en fonction des fonctionnalités utilisées mais aussi de l'éventuel prix s'il n'est pas gratuit. Le choix est également réalisé en fonction des préférences et des habitudes du programmeur. Pour le développement de cette plateforme, c'est Visual Studio Code qui est choisis comme IDE. Il est gratuit et a toutes les fonctionnalités nécessaires pour produire la plateforme décrite.

Ensuite, il faut un outil capable de créer un serveur web localement pour avoir la possibilité de tester le site au fur et à mesure de son développement. Pour ce faire, le choix s'est tourné vers XAMPP (44) qui est un outil libre et open source. Il est composé de plusieurs sous – logiciels qui permettent entre autres d'avoir un serveur Apache http et MySQL. Il est donc très utile pour essayer localement un site web et sa base de données en cours de développement. Il n'est cependant pas recommandé pour un serveur de production à cause de certains problèmes de sécurité (43).

Maintenant que ces deux outils sont en place, Il est possible de commencer la programmation des différentes parties.

4.2.2 Programmation de la base de données

La base de données va être assez simple à mettre en place. Elle ne comporte pas énormément de tables et son analyse est déjà détaillée dans les schémas juste avant.

Pour créer cette base de données, la plateforme phpMyAdmin est utilisée. Elle permet de créer les différentes tables en remplissant les noms des colonnes. Il est également possible d'ajouter des contraintes. L'utilisation de phpMyAdmin est plus amplement décrite dans l'annexe 1.

4.2.3 Programmation du site coté frontend

La programmation de la partie frontend consiste à développer les différentes pages qui vont être affichées sur l'écran de l'utilisateur. C'est la partie « vue » de l'architecture MVC. Pour réaliser ces différentes pages, il y a bien évidemment besoin des trois langages de programmation décrits dans le chapitre 3 (html, css, javascript). Mais deux autres outils vont être utilisés pour faciliter la confection des différentes pages.

Le premier est Bootstrap (également expliqué dans le chapitre 3). Il permet d'obtenir des designs préconçus par la communauté. Cela permet d'avoir un rendu propre. Il existe aussi des thèmes pour site internet déjà préconçu qui permettent d'obtenir des pages et des éléments cohérents point de vue design entre eux. Etant donné que la plateforme est une preuve de concept, il n'est pas nécessaire de produire un design propre à ce site. Il est acceptable de reprendre un thème existant. Pour construire ce prototype, c'est le thème Cosmo(45) qui est utilisé.

Le deuxième outil est Twig(46). Ce moteur de template permet de facilement afficher des données récupérées de la base de données et de créer des conditions. Il permet également d'organiser le code HTML en différentes parties afin de pouvoir les réutiliser sur d'autres pages.

Chaque page du site internet va posséder le même modèle c'est-à-dire une barre de navigation au-dessus et un titre sur la page. Afin d'éviter de réécrire le code pour chaque page, il est préférable de créer un fichier html qui contient le morceau de code réutilisé.

```
<!DOCTYPE html>
<html lang="en">
<head>···
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <a class="navbar-brand" href="/">Mon Site</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarColor02" aria-controls="nav
            <span class="navbar-toggler-icon"></span>
        </button>

        <div class="collapse navbar-collapse" id="navbarColor02">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/">Accueil</a>
                </li>
                {% if user %}
                    <li class="nav-item"><a class="nav-link" href="/conferences">Mes Projets</a></li>
                    <li class="nav-item"><a class="nav-link" href="/rapports">Mes Feedbacks</a></li>
                    <li class="nav-item"><a class="nav-link" href="/profile">Profile</a></li>
                    <li class="nav-item"><a class="nav-link logout" href="/logout">Logout</a></li>
                {% else %}
                    <li class="nav-item"><a class="nav-link" href="/login">Se connecter</a></li>
                    <li class="nav-item"><a class="nav-link" href="/register">S'inscrire</a></li>
                {% endif %}
            </ul>
        </div>
    </nav>
 <div class="container">
    <h1 class="rounded border border-dark m-2 p-2 text-center bg-primary text-white">{% block titre %} {% endblock %}</h1>

    {% block contenu %}{% endblock %}
    </div>


 <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38M
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.6.0/dist/js/bootstrap.bundle.min.js" integrity="sha384-Piv4xVNRyMGpqk



</body>
</html>
```

Figure 12 Fichier HTML utilisé comme base

Ce code servant de base contient plusieurs éléments intéressants. D'abord, dans la partie head, il y a plusieurs balises « link ». Ces balises servent à faire le lien avec des ressources externes à la page.

```
<link rel="stylesheet" href="https://bootswatch.com/4/cosmo/bootstrap.min.css">
<link rel="stylesheet" href="/css/main.css">
<link rel="shortcut icon" href="/images/icon.PNG" type="image/x-icon">
```

Figure 13 balise link

La première balise fait le lien avec un fichier CSS de Bootswatch afin d'importer le thème Cosmo.

Les deux liens suivants sont des références vers des fichiers contenus en local dans le dossier public du projet.

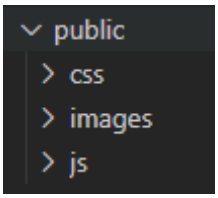

Figure 14 arborescence du dossier "public"

Ensuite, la barre de navigation représentée par le mot clé nav contient différents éléments cliquables pour accéder aux différentes pages. Il y a cependant une condition « if else » dans ce code. En effet, Twig permet d'ajouter des conditions et des boucles directement dans le code HTML. Dans ce cas précis, il permet d'afficher uniquement les boutons « Accueil », « Se connecter » et « S'inscrire » si aucun utilisateur n'est connecté. Si un utilisateur est connecté, les boutons « Se connecter » et « S'inscrire » disparaissent pour laisser place aux boutons « Mes projets », « Profile », …

Une autre utilisation de Twig est le mot clé « block ». Les deux blocs inscrits sur ce fichier ont pour but d'être complétés dans un autre fichier. Puisque le code présenté a pour objectif d'être réutilisé dans chacune des pages, il est bien obligatoire de laisser la possibilité d'ajouter du code html supplémentaire pour personnaliser chaque page. Le bloc titre permet alors de modifier le titre de chaque page. Le bloc contenu, quant à lui, permet de remplir le contenu de la page.

Pour bien comprendre le principe, voilà un exemple de page html qui utilise le code créé pour être réutiliser dans chacune des pages.

```
1   {% extends "base.html.twig" %}
2
3   {% block titre %} Ma page d'accueil {% endblock %}
4
5   {% block contenu %}
6
7   {% endblock %}
```

Figure 15 Exemple 1 de fichier HTML

La première ligne permet d'indiquer qu'il faut récupérer le code écrit dans base.html.twig (le fichier contenant le code réutilisé dans les différentes pages). Ensuite, le bloc titre est rempli avec le titre désiré pour la page et le bloc contenu peut être rempli avec de code html (ici, le bloc contenu est vide). Cet exemple mène à la création de la page ci-dessous lorsqu'un utilisateur est connecté.

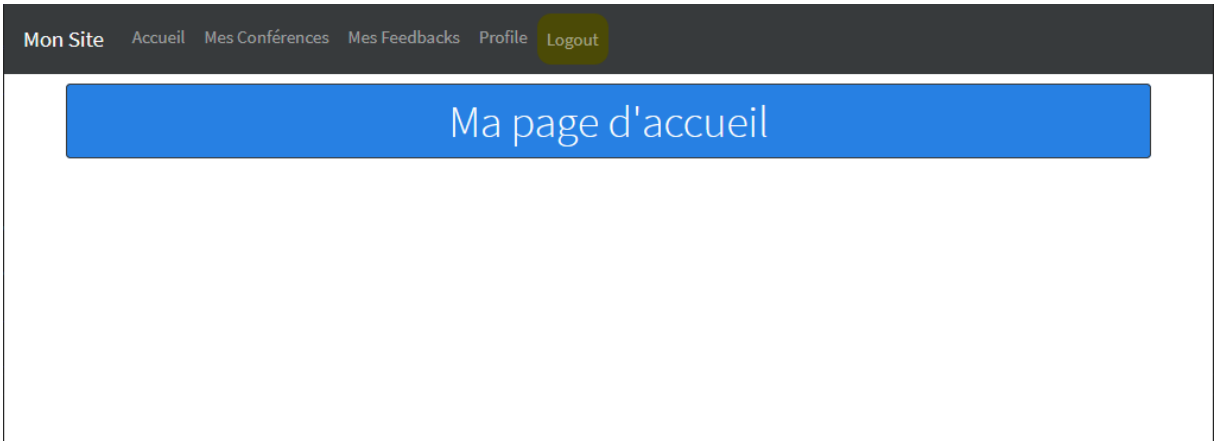

Figure 16 Page d'accueil

Le titre est bien « Ma page d'accueil » et le contenu en dessous est effectivement vide.

Un autre exemple intéressant est la page de connexion.

```twig
{% extends "base.html.twig" %}

{% block titre %} Connexion {% endblock %}

{% block contenu %}
    <div class="container mt-4">
        <div class="card">
            <div class="card-header">
                Formulaire de connexion
            </div>
            <div class="card-body">
            <form action="/auth/login" method="POST">

                <div class="form-group">
                <label for="email">Email</label>
                <input type="email" class="form-control" id="email" name="email">
                </div>

                <div class="form-group">
                <label for="password">Mot de passe</label>
                <input type="password" class="form-control" id="password" name="password">
                </div>

                <button type="submit" class="btn btn-primary">Connexion</button>

            </form>
            </div>
        </div>

        {% if message %}
            <h4 class='alert alert-danger mt-4'>{{message}}</h4>
        {% endif %}
    </div>
{% endblock %}
```

Figure 17 Exemple 2 fichier HTML

La structure de cette page est exactement la même que celle décrite juste avant. La différence est que le bloc de contenu n'est plus vide. Il contient du code HTML qui permet d'afficher des éléments en l'occurrence un formulaire de connexion.

Une subtilité est contenue dans la balise « form ».

```twig
<form action="/auth/login" method="POST">
```

Figure 18 balise form

Celle-ci possède deux attributs qui est « action » et « method ». Dans « method », il est écrit « POST ». Cela signifie que lors de l'appui de l'utilisateur sur le bouton connexion, les données contenues dans la balise « form » (l'email et le mot de passe dans ce cas) vont être envoyées au serveur. Le but étant de traiter ces données et dans ce cas vérifier si les identifiants existent et connecter l'utilisateur à la plateforme. Des informations complémentaires seront expliquées dans la partie suivante.

Il est à noter que le bouton connexion est contenu dans la balise « form ». C'est obligatoire car sans bouton, les données ne s'enverront jamais.

La page de connexion ressemble alors à ceci :

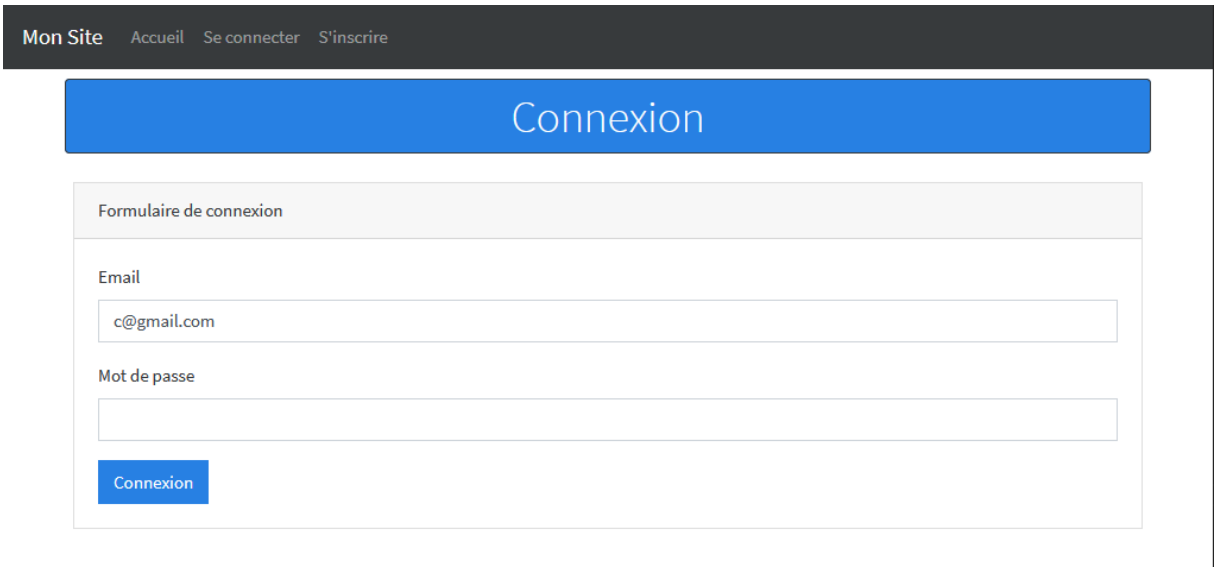

Figure 19 Page de connexion

Les autres pages développées pour ce site ne vont pas être présentées car elles ressemblent toutes plus ou moins aux pages décrites ici.

4.2.4 Programmation du site coté backend

Comme expliqué précédemment, le projet suit une architecture MVC + Routes. Cela signifie que le serveur en tant que tel est décomposé en plusieurs parties. Voici l'arborescence de dossiers et de fichiers de la plateforme.

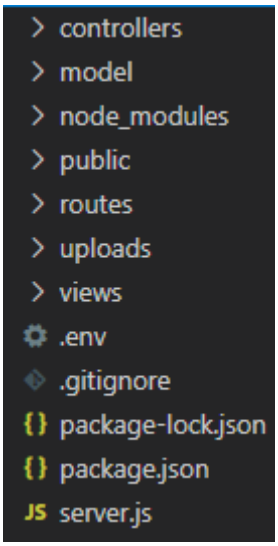

Figure 20 Arborescence du dossier du projet

Les différents dossiers ont déjà été présentés brièvement lors de l'explication de l'architecture MVC. Un fichier « serveur.js » est cependant en dehors de tout dossier. Il s'agit du fichier qui est utilisé lors du démarrage du serveur. Il permet de lancer le serveur mais aussi de se connecter à la base de données et d'importer différentes bibliothèques ou frameworks. Un des framework utilisé est « Express » qui permet de fournir un ensemble de fonctionnalités pour les serveurs Node (47).

Différents modules sont utilisés dans cette application tel que « jsonwebtoken », « bcryptjs », … (48)

Pour commencer, lorsqu'une requête est faite par l'utilisateur, elle est gérée par le fichier « Routes ». En réalité, pour ce serveur, j'ai trouvé plus organisé de séparer le fichier routes en deux fichiers distincts. Le premier s'appelant « pages.js » et le deuxième « auth.js ». Cela permet de séparer simplement les requêtes GET des requêtes POST. Les deux méthodes permettent de communiquer avec le serveur. La différence majeure entre elles est que les données à envoyer au serveur pour une méthode GET sont directement notées dans l'url et donc visibles à l'utilisateur alors que pour POST l'utilisateur ne voit pas les données envoyées.

« Pages.js » s'occupe de toutes les requêtes de type « GET ». Ces requêtes sont envoyées lorsque l'utilisateur demande une nouvelle page.

```
<li class="nav-item"><a class="nav-link" href="/register">S'inscrire</a></li>
```

Figure 21 Bouton impliquant une requête GET

Pour prendre un exemple, lors de l'appui sur le bouton « S'inscrire » de la barre de navigation, « /register » est envoyé vers le fichier pages.js.

```
router.get("/register", displayPages.register)
```

Figure 22 Requête GET dans le fichier pages.js

L'image juste au-dessus montre que dans le fichier pages.js, un « get » correspond à ce « /register ». Il est ensuite suivi d'une fonction « register » qui se trouve dans le fichier « displayPages ». En effet, pour respecter l'architecture MVC, « displayPages » se trouve dans le dossier contrôleur et permet d'appeler la vue qui sera affichée. La fonction « register » ressemble donc à ça.

```
exports.register = (req, res) => {
    res.render("register.html.twig")
}
```

Figure 23 Fonction "register" dans le fichier "displayPages.js"

A noter qu'il est tout à fait possible de ne pas passer par un contrôleur. Il suffit de mettre directement la fonction dans le get comme dans l'image ci-dessous.

```
router.get("/register", (req, res) => {
    res.render("register.html.twig")
})
```

Figure 24 Afficher la page "register" sans passer par le contrôleur

Le problème étant qu'en faisant comme ceci, l'architecture MVC n'est pas respectée car c'est « routes » qui envoie la vue et pas le contrôleur.

Il existe parfois des affichages de pages qui nécessitent des données supplémentaires. Par exemple, la page de profil utilisateur. Elle nécessite évidemment de récupérer les différentes informations de l'utilisateur pour ensuite les afficher. Pour ne pas tout mélanger, un autre fichier dans le dossier « controllers » est utilisé. Il s'agit de « control.js » et il permet de récupérer les données qui doivent être affichées dans la vue. Au final, pour la requête get, il n'y a pas de grands changements. Il suffit de mettre les fonctions nécessaires à la récupération de données dans l'ordre d'exécution dans une liste.

```
router.get('/profile', [control.isLoggedIn, displayPages.profil]);
```

Figure 25 méthode GET avec fonction

La fonction « isLoggedIn » permet de récupérer les informations de l'utilisateur et est exécutée en première. Ensuite, c'est la fonction « profil » qui est exécutée en envoyant la vue à afficher avec les informations récupérées de la première fonction.

Il est à noter que lors de l'appui sur le bouton retour du navigateur, les requêtes « Get » sont à nouveau exécutées.

Les fonctions du fichier Auth.js s'occupent pour leur part des requêtes « POST ». Elles permettent d'envoyer des informations au serveur. Comme cité précédemment, la méthode post est utilisée le plus souvent après que l'utilisateur ait dû remplir un formulaire. Par exemple, quand l'utilisateur s'inscrit sur la plateforme, le fichier « Auth.js » reçoit la requête.

```
router.post('/register', authController.registerControl );
```

Figure 26 Requête POST

Dans cette méthode, la fonction « registerControl » est appelée. Celle-ci permet de vérifier si l'utilisateur n'existe pas déjà et l'enregistrer en sauvegardant ses informations dans la base de données.

Pour accéder à l'entièreté du code, il faut consulter dépôt GitHub disponible à l'adresse :

https://github.com/AxelArn/MemoireAxelArnould

4.3 META-PLATEFORME

Cette plateforme est différente des plateformes classiques habituellement rencontrées sur le net. La particularité réside dans le fait que c'est une meta-plateforme composée de plusieurs sous plateformes. Concrètement, cela signifie que l'utilisateur lors de sa connexion doit choisir entre plusieurs instances possibles. Celles-ci correspondent à une plateforme en tant que telle.

Chaque instance possède des caractéristiques plus ou moins différentes les unes des autres. Cela permet une grande flexibilité et de coller le plus aux besoins des utilisateurs.

Pour prendre un exemple basique, l'instance utilisée pour les conférences scientifiques doit permettre de noter les différents articles déposés alors que l'instance utilisée par les clubs sportifs pas forcément.

La plateforme développée pour ce mémoire possède plusieurs instances avec de petites subtilités dans chacune d'entre elles. Néanmoins, une étude approfondie des besoins de chaque type d'utilisateur serait nécessaire pour déceler des caractéristiques devant être différentes entre certaines instances. Deux instances pourraient alors être très différentes malgré le fait qu'elles partent de la même base.

4.4 CONCLUSION

Ce chapitre a permis de décrire la plateforme de l'analyse à la programmation. La partie analyse permet d'expliquer chaque section du site que ce soit la base de données, le frontend

ou encore le backend. L'explication de la base de données permet de sortir un schéma complet de la base de données désirée. L'analyse du frontend et du backend permet quant à elle de savoir ce que l'utilisateur va voir et les interactions qu'il aura avec la plateforme. Cela a pour but d'avoir une base solide et de savoir en avance les fonctionnalités désirées. Si l'analyse est bien faite, la partie programmation est plus facile et plus rapide.

L'explication de la programmation permet de mettre en pratique les différents points expliqués précédemment. Elle ne se veut pas exhaustive car cela deviendrait beaucoup trop long mais permet de décrire les points importants.

Maintenant que le site est prêt et fonctionnel, il faut le confronter aux premiers utilisateurs.

Chapitre 5: CONFRONTATION AVEC LES UTILISATEURS

Dans ce chapitre, le prototype créé va être confronté aux utilisateurs potentiels. L'objectif est de récolter différents avis de différents utilisateurs. Pour commencer, une explication est réalisée afin de comprendre comment et pourquoi confronter le prototype aux utilisateurs. Ensuite, les différents avis récoltés vont être cités et expliqués. Cela permet de mettre en évidence les problèmes observés par les participants. Pour finir, l'analyse de ces avis est réalisée avec pour objectif de comprendre comment améliorer le prototype un maximum.

5.1 OBJECTIFS

L'objectif principal de cette confrontation est de mesurer l'intuitivité de la plateforme. En effet, une des caractéristiques la plus importante est la facilité d'utilisation. Pour satisfaire celle-ci, le site doit convenir à des personnes de tous domaines. Il faut alors mesurer la capacité de différentes personnes à utiliser la plateforme.

Il est assez difficile pour un développeur de se rendre compte de toutes les subtilités qui peuvent altérer à l'intuitivité de sa plateforme. Effectivement, il l'a construite à sa façon et en connait chaque détail.

Le conseil majeur, mis par exemple en évidence sur internet, pour améliorer un site internet est de penser comme un utilisateur ou de demander aux utilisateurs (49). L'enquête expliquée dans les différentes sections ci-dessous va justement servir à récolter ces avis.

5.2 PRÉPARATION DE L'ENQUÊTE

Une confrontation avec les utilisateurs se doit d'être préparée à l'avance. Il est nécessaire de déterminer quels sont les objectifs de celle-ci. Il faut également mettre en place un mécanisme pour récolter des résultats. Cela peut être au travers de questionnaires, d'interviews, d'observations du comportement,… Il existe de nombreuses façons de récolter des résultats mais toutes ne sont pas forcément adéquates pour chaque situation.

Tout d'abord, il convient de savoir ce qui doit être mesuré. Dans ce cas, c'est l'intuitivité d'utilisation de la plateforme. Cela permet de savoir si les utilisateurs seront capables de

l'utiliser facilement ou si un ou plusieurs éléments les bloquent. Des éléments bloquants pourraient simplement être le fait de ne pas trouver un bouton ou que le bouton soit mal nommé, …

Ensuite, il faut savoir ce que doit réaliser l'utilisateur. En effet, il y a beaucoup de façons de procéder en fonction de ce qui veut être mesuré, du temps disponible de l'utilisateur, … Ici, le but est d'essayer les cinq fonctionnalités principales de la plateforme. Les fonctionnalités sont

- S'inscrire
- Créer un projet
- Ajouter un rapport
- Noter un rapport
- Voir les différentes critiques d'un rapport

Pour finir, il faut pouvoir mesurer si les utilisateurs ont réussi les différents points demandés mais surtout avec quelle facilité ils s'en sont acquittés. En effet, il ne faut pas oublier que les utilisateurs participant à l'enquête vont très certainement essayer plus longtemps que des utilisateurs lambdas qui découvrent la plateforme pour la première fois. Pour récolter leur avis, les participants vont devoir remplir un questionnaire après chaque essai de fonctionnalité.

Lors de ces études, l'utilisateur doit donner son avis personnel mais ne doit pas être influencé. Il faut donc faire en sorte de mettre les mêmes informations à disposition de chaque personne. (52)

5.3 DÉROULEMENT DE L'ENQUÊTE

Pour réaliser cette enquête, des utilisateurs ont été sélectionnés afin qu'ils couvrent des domaines différents. Il leur a été ensuite demandé de remplir une enquête sous forme de Google Form.

Tout d'abord, ils entraient des informations personnelles telles que la tranche d'âge, le métier et les compétences en informatique.

Ensuite, leurs avis sur les cinq fonctionnalités à essayer étaient récoltés. La démarche était la suivante. Ils devaient essayer la fonctionnalité (par exemple s'inscrire à la plateforme) et ensuite ils remplissaient les questions en rapport avec la fonctionnalité dans le Google Form. Chaque fois dans l'intention de récupérer la facilité avec laquelle ils ont réalisé la fonctionnalité mais également avec leurs pistes d'améliorations.

A la fin, il leur était demandé de donner leur avis sur le site en général. Cela permet de mettre en évidence des éléments à améliorer qui n'était pas en rapport direct avec les fonctionnalités essayées.

5.4 TYPE DE PROFILS

Pour l'intérêt de cette enquête, les types de profils ont été sélectionnés pour être variés. En effet, le but est de demander l'avis à des utilisateurs bien différents pour faire un tour d'horizon complet. Il semble normal qu'une personne travaillant sur un ordinateur tous les jours ne rendra pas forcément les mêmes résultats qu'une personne n'utilisant un ordinateur qu'une fois par semaine. Or, la plateforme est susceptible de toucher un type d'utilisateur qui n'est pas forcément à l'aise avec l'informatique. Il faut donc en tenir compte lors de cette enquête.

Pour préserver la confidentialité des participants et cibler plus distinctement des groupes de personnes, ce sont des personas qui vont être utilisés. Des personas sont des personnes fictives possédant des caractéristiques tel que l'âge, le métier, ville où elles habitent, … Ceux-ci représentent un groupe d'individus cibles.

Pour cette enquête, huit personas ont participé.

Persona 1 :

Le persona 1 est une femme âgée entre 25 et 34 ans. Elle exerce la fonction d'Account Manager dans le secteur du service aux entreprises. Elle utilise tous les jours un ordinateur et se considère plutôt à l'aise avec ce type d'outils. La plateforme présentée pourrait, par exemple, lui être utile dans son travail notamment pour partager des données, des enquêtes, … avec ses collègues au sujet de ses clients. Les personnes avec qui elle travaille pourront y ajouter des informations supplémentaires qu'elles ont en leur possession. De plus, personna 1 est active dans un club de volley-ball et fait partie du comité de ce club sportif. Elle y occupe le poste de secrétaire et doit partager avec les autres membres du comité mais également avec tous les membres du club certains documents qui doivent pouvoir être consultés et commentés.

Persona 2 :

La deuxième personne interrogée est un étudiant qui se situe dans la tranche d'âge 18-24 ans. Il étudie l'audiologie et est donc amené à travailler dans le secteur de la santé. Dans le cadre

de ses études, il utilise son ordinateur tous les jours. Concernant cette personne la plateforme pourra lui être utile, comme pour tous les autres étudiants, afin de déposer des travaux à destination des professeurs et ceux-ci pourraient noter et commenter les travaux. Le persona 2 fait également partie d'un orchestre de musique. Dans ce type d'organisation, il pourrait en plus de pouvoir déposer des documents, déposer des pistes audios qui pourrait être évaluées et commentées.

Persona 3 :

Le persona 3 est étudiant en Marketing. Son âge se situe entre 18 et 24 ans. Son utilisation de l'ordinateur est quotidienne et il est assez à l'aise avec celui-ci. De son côté, la plateforme pourrait être exploitée de la même manière que l'étudiant précédent c'est-à-dire déposer des travaux que les professeurs pourront noter et commenter.

Persona 4 :

La quatrième personne est directeur général dans le secteur de la construction. C'est un homme âgé entre 45 et 54 ans. Il utilise un ordinateur à raison de cinq fois par semaine. Il considère ses compétences informatiques comme assez moyennes. Dans le cadre de son travail, cette plateforme pourrait, par exemple, être utilisée pour partager les plans d'une construction avec les travailleurs qui doivent pouvoir les consulter mais également avec les clients. Ces plans pourront donc être commentés en vue d'une amélioration.

Persona 5 :

Le persona 5 est une étudiante en gestion des ressources humaines âgée entre 18 et 24 ans. Son utilisation d'un ordinateur est en moyenne entre quatre et cinq fois semaine et elle se définit comme étant à l'aise avec les programmes de base. Comme les autres étudiants, la plateforme serait utile pour déposer les différents travaux et recevoir des commentaires de la part des professeurs.

Persona 6 :

La sixième personne est une enseignante dans le secondaire âgée entre 45 et 54 ans. Elle utilise un ordinateur tous les jours et considère ses compétences informatiques comme assez bonnes. Dans sa situation, l'utilisation de la plateforme sera similaire à celle des étudiants si ce n'est qu'elle se trouvera du côté de l'enseignant et pas de l'étudiant. De plus, le directeur

de son école pourrait envoyer divers documents à destination de ses professeurs ou/et de ses étudiants afin d'avoir un retour sur ceux-ci.

Persona 7 :

Le persona 7 qui a été interrogé est un homme âgé entre 18 et 24 ans. Il est inspecteur de police. Dans le cadre de son travail, il utilise l'ordinateur 4 à 5 fois par semaine et situe ses compétences informatiques comme bonnes. Dans sa fonction, la plateforme pourrait être utilisée pour partager avec ses collègues et ses supérieurs certains documents pour récupérer leur feedback.

Persona 8 :

La huitième personne est une étudiante en informatique âgée entre 18 et 24 ans. Elle utilise son ordinateur tous les jours et considère avoir de bonnes connaissances dans ce domaine. Comme pour les autres étudiants, elle pourrait utiliser la plateforme pour rendre ses travaux et obtenir un feedback de ses professeurs.

Persona 9 :

Le persona 9 est une femme âgée de plus de 65 ans qui est à la retraite. Elle utilise son ordinateur une à deux fois semaine et considère qu'elle n'est pas à l'aise dans son utilisation. Elle se limite à consulter les mêmes sites internet récurrents. Cette personne fait partie d'un cercle horticole. Dans ce cadre, le président de ce cercle est régulièrement amené à partager des documents avec les autres membres. Cela lui permet de prévenir des différents points abordés à la prochaine réunion ou encore de la documentation sur le jardinage. La plateforme pourrait alors être utilisée par le président ce qui permettra de recevoir des feedbacks des membres.

5.5 RÉSULTATS DE L'ENQUÊTE

5.5.1 L'inscription

La première fonctionnalité devant être essayée par les participants était l'inscription. Tous les participant ont facilement réussi à s'inscrire et n'ont rencontré aucune difficulté pour cette étape. Ils ont cependant émis quelques remarques.

La première est le fait que lorsqu'on se trompe dans la confirmation du mot de passe, toutes les informations précédemment remplies sont effacées. Il faut alors recommencer l'étape

depuis le début. Il est vrai que c'est assez pénible pour un utilisateur de devoir remplir une nouvelle fois tout le formulaire d'inscription lorsqu'il fait une simple erreur dans la confirmation de son mot de passe.

La deuxième est sur le choix de son pays. Lors de l'inscription, il est demandé d'indiquer le pays. Il doit être sélectionné au sein d'une liste déroulante et ne peut pas être écrit directement. Cela a gêné un des participants qui aurait préféré l'introduire directement pour gagner un peu de temps.

La troisième remarque est celle qui est revenue le plus. Elle concerne la protection des données. En effet, plusieurs participants ont mentionné le fait qu'il n'y avait pas d'explication quant à l'utilisation de leurs données. Il n'est plus possible, à l'heure actuelle, de récolter et d'utiliser les données en Union Européenne sans communiquer différents éléments aux utilisateurs. Ceux-ci sont explicités dans le cadre du RGPD (règlement général sur la protection des données) créé par l'Union Européenne et mis en application depuis le 25 mai 2018. Cette remarque est extrêmement pertinente pour la suite du développement de la plateforme et c'est un élément indispensable devant être réfléchi si celle-ci est mise à disposition du public.

5.5.2 Création d'un projet

La création du projet s'est dans l'ensemble réalisée facilement avec une moyenne de 4.89/5 pour l'accès à la page et de 4.56/5 pour la facilité de création d'un projet. Cependant une remarque a été énoncée presque unanimement par les participants de cette enquête. Ils ont remarqué que l'ajout d'autres personnes au projet n'est pas pratique. En effet, il est mentionné que celui-ci doit se faire avec l'adresse mail de chacune des personnes. Le problème est qu'ils ne connaissent pas forcément le mail de chaque personne. C'est un point qu'il faudra évidemment améliorer pour faciliter l'utilisation de la plateforme.

Certains suggéraient également de laisser la possibilité au créateur du projet d'ajouter ou de supprimer des utilisateurs. Effectivement, le groupe de personnes sélectionnées est souvent sujet à changement. Il suffit qu'une personne veuille devenir membre et il faut recréer tout un projet. Ce qui implique également que les documents postés et leurs commentaires ne seront plus visibles dans ce nouveau groupe.

5.5.3 Ajouter un rapport

L'ajout d'un rapport a été une fonctionnalité un tout petit peu plus difficile à réaliser que les deux précédentes. En effet, l'accès à la page d'ajout de rapport est noté d'un 4.44/5 tandis que la facilité d'ajout a obtenu 4.44/5.

Pour ajouter un rapport, l'utilisateur devait entrer dans un projet et ensuite ajouter le rapport dans celui-ci. La remarque principale émise par plusieurs participants est que le bouton pour « Entrer » dans un projet n'est pas suffisamment instinctif. Pour pallier à ce problème, il faudrait peut-être trouver un nom de bouton plus approprié plutôt que simplement « Entrer » ou rendre ce bouton plus visible.

Une deuxième suggestion faite par un participant est d'avoir une confirmation que l'ajout d'un document a bien été effectué. Cette confirmation pourrait être affichée directement sur le site ou envoyée par mail. Il propose également d'alerter par mail l'ensemble des participants au projet lors de l'ajout d'un document.

Une dernière suggestion serait d'insérer une option afin de n'accepter qu'un format de document pour, par exemple, obliger de ne soumettre que du format PDF ou CSV.

Le persona 9 qui est la personne qui utilise le moins l'ordinateur entre tous les participants a fait part de sa difficulté à poster un de ses fichiers. En effet, son utilisation de l'ordinateur se limite à consulter quelques sites internet mais c'était la première fois qu'elle devait poster un document à partir de ses fichiers. Malgré ce fait, cette personne a quand même réussi à effectuer la tâche. Il serait peut être utile d'ajouter une explication supplémentaire pour les personnes n'ayant jamais utiliser ce système.

5.5.4 Noter un rapport

Pour essayer cette fonctionnalité, l'utilisateur devait noter un rapport. Cette étape s'est déroulée avec facilité pour tout le monde avec une moyenne de 4.89/5 pour l'accès à la page et 4.67/5 pour la notation. Néanmoins, plusieurs participants ont avancé une remarque concernant la notation. Ils suggèrent qu'il serait plus pratique pour eux de mentionner que la note attribuée au rapport est sur 5.

Une autre remarque était de prévenir par mail lorsqu'une note est produite par un utilisateur.

5.5.5 Voir les critiques d'un rapport

La dernière étape était d'aller sur la page qui affiche toutes les critiques d'un document. Celle-ci s'est réalisée très facilement pour tous les participants. Ils ont, par contre, donné deux suggestions principales.

La première est d'ajouter une date et une heure à chaque note. Cela permettrait d'une part de savoir quand les différentes notes ont été envoyées et d'autre part de faire un tri en fonction des dates.

La deuxième est de pouvoir répondre directement à un commentaire spécifique. Cela permettrait d'ajouter de l'interactivité et de la lisibilité à cette page.

5.5.6 Plateforme en général

Dans la dernière partie de l'enquête, les participants ont noté le site en général. Ils ont également proposé des améliorations pour rendre le site plus intuitif et plus complet. La plateforme s'est vue attribuer une note moyenne de 4.22/5.

La remarque principale prononcée par quasiment tous les participants de cette enquête est que l'esthétique mériterait d'être améliorée. En effet, la plateforme possède encore un design très basique qui doit être amélioré que ce soit pour le fond d'écran, pour les boutons, pour les couleurs, …

Une suggestion supplémentaire était de changer l'ordre des projets soit en classant ceux-ci du plus récent au plus ancien soit de faire des filtres pour permettre à l'utilisateur de personnaliser l'agencement à sa manière. Une barre de recherche pourrait également être utile pour retrouver rapidement un projet précis.

Une dernière remarque a été émise au sujet du profil utilisateur. Il serait bien de pouvoir charger et utiliser une photo personnelle. Il serait également utile de pouvoir modifier ses différentes informations car pour l'instant ce n'est pas modifiable.

5.5.7 Utilité de l'enquête

Cette enquête a permis de passer en revue une grande partie de la plateforme par les participants. Cela a donné une indication sur la facilité d'utilisation et l'intuitivité de la plateforme. Les remarques des différents participants sur les différents points du site sont également très utiles pour la suite du développement. C'est un avis utilisateur à ne pas prendre à la légère. Cela permettra une nette amélioration entre la preuve de concept décrite tout le long de ce mémoire et la plateforme mise en ligne. Le fait que ce soit des personnes provenant de domaines différents permet de s'assurer que la plateforme est suffisamment intuitive pour tous types de personnes.

5.6 INTÉGRATION DE DIFFÉRENTES SUGGESTIONS

Dans la section précédente, différentes suggestions d'améliorations ont été citées. Celles-ci ont été établies par les utilisateurs lors de l'enquête. Ces propositions d'améliorations sont donc importantes à mettre en place pour faciliter l'intuitivité et l'utilisation de la plateforme.

Dans le cadre de ce mémoire, l'objectif n'est pas de créer une plateforme finie et prête à l'emploi donc il n'y a pas besoin d'affiner la plateforme jusque dans les moindres détails. Néanmoins, certaines des suggestions proposées par les utilisateurs ont été implémentées pour montrer comment l'architecture de ce prototype peut être étendue afin de répondre aux suggestions.

Dans les améliorations réalisées, il y a, par exemple, le respect du RGPD. Pour mettre en place cette suggestion, une page est ajoutée avec les différentes règles de confidentialité du site. Pour y accéder, l'utilisateur aura un lien en bas de la page d'inscription. Les différentes règles n'ont pas été poussées au point de les faire vérifier par un avocat cela ne constitue qu'un exemple pour démontrer que la suggestion était réalisable.

Une autre amélioration mise en place est la possibilité d'ajouter des utilisateurs sans connaitre leur adresse mail mais uniquement leur nom et prénom. Ce système ressemble à celui d'une boîte mail lors de l'ajout d'un destinataire.

D'autres suggestions ont été également mises en place comme l'ajout d'explications supplémentaires à certains endroits par exemple.

Certaines suggestions n'ont par contre pas été réalisées lors de la nouvelle version du site internet. L'exemple le plus marquant est le design de la plateforme. Beaucoup de participants ont mentionné le fait qu'il serait appréciable d'améliorer le design du site. Il est tout à fait possible de modifier le graphisme du site plus ou moins simplement en fonction de la demande. Cependant, cette amélioration n'a pas été effectuée. La raison principale est que faire un design est un métier à part entière qui demande des compétences que je ne possède pas.

5.7 CONCLUSION

Dans ce chapitre, une enquête est réalisée pour mesurer l'intuitivité du site internet et de ses fonctionnalités. Tout d'abord, les objectifs et le déroulement de l'enquête sont expliqués. Cela permet de comprendre exactement pourquoi et comment l'enquête est réalisée. Ensuite, les participants de celle-ci sont présentés sous forme de persona. Cette présentation est importante car elle montre que les différents personas ne sont pas dans le même domaine d'activité et n'ont pas le même âge. Cela veut dire que tous les types de personnes peuvent utiliser la plateforme. Enfin, un résumé des résultats obtenus est présenté pour comprendre ce que les participants ont pensé de l'intuitivité du site en général mais également de certaines fonctionnalités.

Après une analyse complète de ces résultats, le chapitre suivant va expliquer les différentes améliorations possibles mais également les perspectives que la plateforme peut avoir.

Chapitre 6: PERSPECTIVES

Le sujet présenté dans ce mémoire ouvre de nombreuses possibilités tant sur les améliorations possibles à la plateforme que pour ses perspectives d'utilisations. Ce chapitre vise à démontrer ces différents points.

Dans la première partie du chapitre, une série d'améliorations seront explicitées afin de rendre la plateforme plus intuitive. Les suggestions des participants à l'enquête explicitées au chapitre précédent vont également être passées en revue.

La deuxième partie du chapitre présente les différentes perspectives de cette plateforme. Celles-ci pourront se concentrer sur la mise en ligne de la plateforme. Quelles sont les différents éléments nécessaires pour la mettre en ligne mais également les différentes voies dans lesquelles cette plateforme pourrait s'orienter.

6.1 AMÉLIORATION

Cette plateforme décrite tout au long de ce mémoire n'a, à ce stade, pas la prétention d'être complète ou finie. C'est une preuve de concept créée pour démontrer le développement d'une telle plateforme mais également pour récolter les premiers feedbacks d'utilisateurs au travers d'une enquête.

Sachant cela, il est évident que cette preuve de concept n'est pas parfaite et qu'il y a des éléments à ajouter, modifier ou supprimer. Les feedbacks récoltés constituent une première liste de modifications afin de perfectionner cette plateforme. Une autre liste plus technique peut aussi être dressée pour perfectionner la sécurité, la rapidité, …

6.1.1 Esthétisme

Le manque d'esthétisme(50) est la remarque émise unanimement par l'ensemble des participants à l'enquête. Bien que ce ne soit pas obligatoire pour le bon fonctionnement d'un site internet, il est important de soigner l'esthétisme car c'est la première caractéristique jugée par l'utilisateur.

En effet, le design n'était, dans un premier temps, pas la préoccupation principale lors du développement de cette preuve de concept. C'est pour cela que des éléments de Bootstrap sont utilisés. Ces éléments ont l'avantage d'avoir été utilisés par de nombreux utilisateurs et ont donc moins de chances de posséder un bug. Le problème est qu'ils sont toujours assez semblables et n'apportent pas réellement un design unique à la plateforme. Pour améliorer

ce point de la plateforme, il n'est pas nécessaire de se séparer totalement de Bootstrap mais il faut pouvoir personnaliser quelques peu le design de ce site.

La plupart des grandes entreprises du net disposent de plusieurs employés uniquement chargés du design de leur site internet. Pour la plateforme, il pourrait être envisagé de demander à un web designer d'y apporter une touche singulière.

6.1.2 Intuitivité

Bien que l'intuitivité du site en général et de ses différentes fonctionnalités ont obtenu une note plutôt élevée par les participants de l'enquête, elle peut encore être perfectionnée. Certaines remarques émises sont très pertinentes et méritent d'être approfondies.

Tout d'abord, le fait de devoir rentrer le mail de la personne pour l'ajouter à un projet n'est pas intuitif. Cela est acceptable dans une preuve de concept mais doit absolument être amélioré par la suite. L'amélioration utilisée est de trouver les utilisateurs par leur nom et prénom. Il y a une barre de recherche spécifique où on peut soit mettre le nom et prénom soit écrire directement l'adresse mail. L'ajout de cette fonctionnalité ne signifie pas qu'il n'est pas possible d'encore améliorer celle-ci.

Ensuite, lors du dépôt d'un rapport, il n'y a pas de confirmation visuelle mis à part le fait que le rapport s'ajoute bien dans la liste des rapports du projet. Cette méthode n'est sans doute pas la meilleure surtout lorsque le projet compte un grand nombre de rapports. L'utilisateur ne verrait pas forcément si son document a bien été ajouté. Une des suggestions d'un participant était d'ajouter une popup ou un message à l'écran pour signaler que le document est bien déposé.

Puis, pour faciliter la compréhension des utilisateurs, des noms plus explicites sur certains boutons pourraient être ajoutés. Une explication plus complète de ce que peut faire l'utilisateur à chaque étape peut être utile également. Ces deux points permettraient d'améliorer l'intuitivité de la plateforme.

Enfin, les commentaires émis par des utilisateurs à un rapport ne comportent pas de date. Cet ajout peut permettre une plus grande lisibilité dans les différents commentaires postés.

6.1.3 Fonctionnalités

Les fonctionnalités principales de la plateforme sont développées. Celles-ci permettent de répondre aux objectifs principaux de la gestion de rapports. Cependant, des fonctionnalités complémentaires pourraient être ajoutées pour faciliter la vie des utilisateurs. La plupart des suggestions ont pour but de faciliter ou d'améliorer les fonctionnalités principales du site internet.

Un élément primordial est de faire gagner du temps à l'utilisateur. Lors de l'inscription, tous les champs s'effaçaient quand la confirmation du mot de passe n'est pas la bonne. Ce genre d'événement est assez frustrant pour l'utilisateur. Il doit recommencer le processus ce qui lui fait perdre son temps. Il est donc important que les champs ne s'effacent pas mais plutôt d'indiquer sur lequel est l'erreur. Une autre perte de temps mentionnée est lors du choix du pays. Ce choix est réalisé dans une liste déroulante de près de deux cents choix. Une amélioration possible serait de laisser un champ écrit avec éventuellement une suggestion en dessous. De ce fait, l'utilisateur noterait les deux ou trois premières lettres de son pays et tomberait sur la bonne suggestion. Ce principe peut être utilisé dans toutes les listes déroulantes de la plateforme.

Le gain de temps se fait également lors de recherche de documents ou de projets. Lorsque le nombre de projets ou le nombre de documents dans un projet devient important, il est plus difficile pour les utilisateurs de s'y retrouver. Pour pallier à cela, des filtres et une façon de classer les éléments doivent être mis en place. Par exemple, dans la liste des projets, il faut ajouter une possibilité de classer ces éléments du plus récent au plus ancien ou l'inverse mais également pouvoir filtrer en fonction du créateur du projet, du nom du projet, … Il est évident que les éléments cités peuvent également s'appliquer sur la liste des documents présents dans un projet ou encore sur les commentaires postés. Tous ses petits ajouts peuvent vraiment faciliter la vie de l'utilisateur.

6.2 MISE EN LIGNE

La preuve de concept présentée démontre la possibilité de développer une telle plateforme. Bien entendu, celle-ci ne peut pas être ouverte à tous les utilisateurs potentiels dans son état actuel. Il faut d'abord analyser différents éléments.

6.2.1 Hébergement

L'hébergement d'un site web consiste à placer celui-ci sur un serveur afin de le rendre disponible à tous. Pour commencer, il y a plusieurs façons d'héberger un site internet sur son serveur. Il est possible par exemple de créer un serveur directement sur un ordinateur personnel ou une carte type Raspberry. L'inconvénient de cette méthode est qu'il faut que l'appareil reste allumé continuellement. Il faut également que son débit internet soit suffisant sinon de gros problèmes de latences seront ressentis chez les utilisateurs. Le nombre d'utilisateurs simultanés ne doit pas dépasser la puissance du serveur sinon il risque de planter. Mais le plus grand problème avec cette méthode est qu'il nécessite une vigilance accrue d'un point de vue sécurité. Il faut en réalité ouvrir un port de sa propre box internet ce qui n'est vraiment pas conseillé. De plus, lors d'un problème quelconque sur ce serveur, la personne l'ayant créé ne sera pas aidée. Pour éviter ce genre d'inconvénient, la plupart des

sites internet sont hébergés chez un hébergeur dédié. Ce sont des sociétés mettant à disposition un serveur continuellement allumé et relié à internet. Il existe un nombre très important d'offres pour héberger un site internet. Chacune a des avantages et des inconvénients suivant le site à héberger. Il est nécessaire de vérifier les différentes offres et ne pas en prendre une au hasard. Ces hébergeurs garantissent la sécurité du serveur, un service client et, en fonction de l'offre souscrite, une sauvegarde du site et de ses données.

Pour l'hébergement du site, il est préférable d'utiliser un nom de domaine. Il est le masque de l'adresse IP du serveur contenant le site. Il permet de communiquer et de retenir plus facilement l'adresse de celui-ci. Il est plus aisé de mémoriser « monsite.com » que « 216.32.74.51 ». Un nom de domaine peut directement être acheté chez un hébergeur.

Dans le cas de la plateforme, elle est disponible à l'adresse : https://gentle-wave-24743.herokuapp.com

Il est possible que le site ait un peu de latence au départ car il est héberger sur une version gratuite de Heroku qui met le serveur en veille.

6.2.2 Sécurité

La section précédente évoquait la question de la sécurité. Dans cette section, la question va être approfondie en expliquant les différents problèmes qui peuvent survenir mais également en expliquant comment s'en protéger. Tout le monde sait qu'internet est un endroit dangereux. Il est d'ailleurs assez fréquent que des sites internet soient piratés. Il existe plusieurs raisons qui poussent certaines personnes à pirater un site mais bien souvent, c'est l'argent qui les motive. Les méthodes de hacking ne cessent de se développer. C'est impossible d'avoir une plateforme sécurisée à cent pourcent. Malgré ce fait, il est possible de mettre en place des stratégies pour éviter de faciliter la vie de la personne mal intentionnée.

Pour parfaire la sécurité du site, il est préférable de commencer par utiliser un logiciel qui va scanner automatiquement et donner les premières grosses failles de sécurités. Ces logiciels sont très utiles car ils sont à la fois utilisés par des personnes voulant sécuriser des systèmes que par des personnes voulant les pirater. Le fait que ce genre de scanner ne trouve pas de failles de sécurité est déjà un très bon point. Ce scan est le minimum avant de mettre son site en ligne afin d'éviter toutes surprises.

D'autres méthodes de sécurité telles que les tests de pénétration sont possibles mais demandent un investissement plus conséquent. Il est donc important de savoir à quel point il est nécessaire de sécuriser le site.

Dans le cas de la plateforme, deux groupes de données peuvent être prisés par d'éventuels hackers.

Le premier contient les données personnelles des utilisateurs de la plateforme enregistrés lors de l'inscription. Bien qu'il n'ait pas demandé des informations extrêmement sensibles comme un compte bancaire ou un numéro de carte d'identité il faut rester vigilant. En effet, la base de données contient les adresses mails ainsi que le nom et l'âge des personnes inscrites. Certaines sociétés sont prêtes à débourser de l'argent pour ce genre d'informations. Un des éléments les plus sensibles dans les données enregistrées est sans doute le mot de passe des utilisateurs. Beaucoup de personnes utilisent le même mot de passe pour plusieurs sites internet. Une fuite de mot de passe et d'adresse mail pourrait alors avoir des conséquences dramatiques puisqu'il serait possible de se connecter à d'autres sites qui peuvent éventuellement contenir des informations encore plus sensibles. Pour protéger les utilisateurs, les mots de passe contenus dans la base de données sont chiffrés et donc illisibles. Malgré cela, il est plus prudent de vérifier si réellement ces données ne sont pas accessibles.

Le deuxième groupe de données sont tous les articles déposés sur la plateforme. Certains de ceux-ci pourraient être confidentiels ou contenir des informations sensibles. Il est donc préférable de vérifier s'il est possible d'obtenir ces fichiers.

6.3 LES POSSIBILITÉS DU SITE

Cette plateforme peut être utilisée dans beaucoup de domaines. Lors des enquêtes, certains participants ont trouvé des utilisations possibles dans certaines sphères de leur vie telles que les études, les activités sportives, les conseils communaux et le partage de documents dans les entreprises. L'avantage est que cette plateforme est une meta-plateforme qui recense plusieurs instances. Chaque instance est alors plus spécialisée dans un domaine précis. Cette spécialisation permet d'améliorer les fonctionnalités pour chaque domaine d'utilisation. Elle est, en réalité, un des avantages à utiliser cette plateforme et elle constitue la principale voie d'amélioration. Le site pourrait faire l'objet de nombreuses autres enquêtes pour l'utilisation de chaque domaine en particulier. Avec celle-ci, il serait possible d'améliorer une partie précise du site et de le rendre plus personnalisé pour chaque domaine. D'autres méthodes d'enquêtes beaucoup plus poussées pourront également être utilisées pour améliorer l'intuitivité. Par exemple, une étude de type « eye tracking »(51) pourrait être utilisée pour repérer exactement ce que les utilisateurs regardent sur le site et ainsi améliorer l'emplacement de certains éléments mais également supprimer certains éléments qui pourraient perturber les utilisateurs.

Chapitre 7: CONCLUSION

La gestion de rapports apparait dans beaucoup de domaines de la vie. En effet, les conférences scientifiques sont l'exemple le plus connu mais des conseils d'administrations, conseils communaux font également cette gestion. A la différence des conférences scientifiques, ceux-ci ne possèdent pas vraiment de plateformes adaptées à leurs besoins. Ils sont souvent obligés de passer par des outils tiers tels que les mails ou les réseaux sociaux. Le sujet de ce mémoire a pour but de montrer qu'il est possible de créer une plateforme de gestion de rapport qui se veut générique. Cela permet aux différents domaines d'avoir une plateforme utilisable et en adéquation avec leurs besoins.

A travers ce travail, j'ai essayé de développer une solution intuitive et pouvant être utilisée dans tous les domaines. Pour ce faire, le chapitre sur la problématique abordée a permis de d'expliquer pourquoi développer une telle plateforme. Puis, le concept décrit, quant à lui, une première analyse de ce site.

Des recherches sur les différents outils de programmation ont été réalisées pour choisir ceux qui correspondaient le mieux aux caractéristiques mentionnées dans le chapitre sur le concept. Ensuite, l'analyse complète et les étapes de programmation ont décrit complètement la plateforme.

L'intuitivité de la plateforme est mesurée lors d'une enquête. Celle-ci a regroupé plusieurs participants de domaines différents pour couvrir un large panel d'utilisateurs. Ceux-ci ont permis de mettre en lumière les défauts de cette plateforme et les points d'amélioration possibles. Le dernier chapitre a décrit les améliorations à apporter pour améliorer ce site. Il a également expliqué quels étaient les points d'attentions pour mettre en ligne cette plateforme.

La plateforme développée ne correspond pas à une plateforme finie et prête à l'emploi. Elle correspond plutôt à une preuve de concept pour démontrer que les analyses expliquées dans ce mémoire sont utilisables dans la pratique. Par suite, le dernier chapitre jette des perspectives d'amélioration pouvant conduire à une plateforme complètement peaufinée.

BIBLIOGRAPHIE

(1) EasyChair, https://easychair.org/ ,(Dernière accès : 29/03/2021)

(2) Usabilis, Material Design, l'UI selon Google !, https://www.usabilis.com/material-design-lui-selon-google/ (last revised 17-04-2018) (Dernière accès : 04/04/2021)

(3) Mustafa Kurtuldu, Basics of UX, https://developers.google.com/web/fundamentals/design-and-ux/ux-basics , (Dernière accès : 02/04/2021)

(4) Visual Paradigm, What is Use Case Diagram?, https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ , (Dernière accès : 04/02/2021)

(5) W3, Web Development Roadmaps, https://www.w3schools.com/whatis/ , (Dernière accès : 15/02/2021)

(6) Duckett, J. (2011), *HTML & CSS: design and build websites (Vol. 15),* Indianapolis, IN: Wiley.

(7) MDN Web Docs, CSS : Feuilles de style en cascade, https://developer.mozilla.org/fr/docs/Web/CSS , (Dernière accès : 23/11/2020)

(8) W3, Cascading Style Sheets, https://www.w3.org/Style/CSS/#specs , (Dernière accès : 23/11/2020)

(9) Haverbeke, M. (2018), *Eloquent JavaScript: a modern introduction to programming,* No Starch Press.

(10) numendo, Savez-vous faire la différence entre un framework et une bibliothèque ?, https://www.numendo.com/blog/framework/quelle-est-la-difference-entre-un-framework-et-une-bibliotheque/#:~:text=Framework%20%3A%20une%20infrastructure%20logicielle%20cl%C3%A9s,inclut%20une%20ou%20des%20biblioth%C3%A8ques , (Dernière accès : 05/04/2021)

(11) Banks, A., & Porcello, E. (2017), Learning React: functional web development with React and Redux, " O'Reilly Media, Inc.".

(12) Freeman, A. (2014), *Pro AngularJS*, Apress.

(13) Bootstrap, Build fast, responsive sites with Bootstrap, https://getbootstrap.com/, (Dernière accès : 02/04/2021)

(14) Bootstrap, Learn more about the team maintaining Bootstrap, how and why the project started, and how to get involved, https://getbootstrap.com/docs/5.0/about/overview/, (Dernière accès : 02/04/2021)

(15) Vivian Cromwell, Evan You, https://web.archive.org/web/20170603052649/https:/betweenthewires.org/2016/11/03/evan-you/, (Dernière mise à jour : 03-11-2016) (Dernière accès : 06/04/2021)

(16) Macrae, C. (2018), *Vue. js: Up and Running: Building Accessible and Performant Web Apps,* " O'Reilly Media, Inc.".

(17) Java, What is Java technology and why do I need it?, https://www.java.com/en/download/help/whatis_java.html, (Dernière accès : 02/11/2020)

(18) Spring, https://spring.io/, (Dernière accès : 08/04/2020)

(19) Shaw, Z. A. (2013), *Learn Python the hard way: A very simple introduction to the terrifyingly beautiful world of computers and code,* Addison-Wesley.

(20) Django, https://www.djangoproject.com/, (Dernière accès : 02/11/2020)

(21) Ruby, https://www.ruby-lang.org/en/, (Dernière accès : 02/11/2020)

Flanagan, D., & Matsumoto, Y. (2008). The Ruby Programming Language: Everything You Need to Know. " O'Reilly Media, Inc.".

(22) RubyOnRails, https://rubyonrails.org/, (Dernière accès : 02/11/2020)

(23) Sébastien Doeraene, Scala.js no longer experimental, https://www.scala-lang.org/blog/2015/02/05/scala-js-no-longer-experimental.html , (Dernière mise à jour : 5-2-2015)(Dernière accès : 02/11/2020)

(24) Zandstra, M. (2010), *PHP Objects, Patterns and Practice,* Apress.

(25) Histoire de PHP, https://www.php.net/history.php, (Dernière accès : 02/11/2020)

(26) A brief history of Node.js, https://nodejs.dev/learn/a-brief-history-of-nodejs, (Dernière accès : 03/11/2020)

(27) Nodejs, https://nodejs.org/en/, (Dernière accès : 03/11/2020)

(28) Justin Carroll, A history of Node.js, https://builtinnode.com/2019/05/04/a-history-of-node-js/ , (Last revisited 4-5-2019)(Dernière accès : 03/11/2020)

(29) MDN Web Docs, Inner-browsing extending the browser navigation paradigm, https://developer.mozilla.org/en-US/docs/Archive/Inner-browsing_extending_the_browser_navigation_paradigm, (Dernière accès : 05/04/2021)

(30) MDN Web Docs, HTML5, https://developer.mozilla.org/fr/docs/Web/Guide/HTML/HTML5, (Dernière accès : 04/04/2021)

(31) IBM, ACID properties of transactions, https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions, (Dernière accès : 06/11/2020)

(32) MySQL, https://www.mysql.com/, (Dernière accès : 05/11/2020)

(33) MariaDB, https://mariadb.org/, (Dernière accès : 05/11/2020)

(34) Why is the Software Called MariaDB?, https://mariadb.com/kb/en/why-is-the-software-called-mariadb/, (Dernière accès : 05/11/2020)

(35) Oracle, https://www.oracle.com/en/database/, (Dernière accès : 05/11/2020)

(36) Obe, R. O., & Hsu, L. S. (2017), *PostgreSQL: Up and Running: a Practical Guide to the Advanced Open Source Database,* " O'Reilly Media, Inc.".

(37) MSSQL, https://www.microsoft.com/fr-be/sql-server/sql-server-downloads, (Dernière accès : 05/11/2020)

(38) Most Widely Deployed and Used Database Engine, https://www.sqlite.org/mostdeployed.html, (Dernière accès : 27/03/2021)

(39) Appropriate Uses For SQLite, https://www.sqlite.org/whentouse.html, (Dernière accès : 27/03/2021)

(40) Mike Butusov, Companies That Use Node.js for Backend: How Do Big Players Benefit from It?, https://blog.techmagic.co/companies-that-use-node-js-for-backend-how-do-big-players-benefit-from-it/ , (Dernière mise à jour : 1 – 10 – 2020) (Dernière accès : 11/11/2020)

(41) Deacon, J. (2009). Model-view-controller (mvc) architecture. *Online][Citado em: 10 de março de 2006.] http://www. jdl. co. uk/briefings/MVC. pdf*.

(42)Hainaut, J. L. (2018). *Bases de données-4e éd.: Concepts, utilisation et développement*. Dunod.

(43)Apache Friends, Linux Frequently Asked Questions, https://www.apachefriends.org/faq_linux.html, (Dernière accès : 20/04/2021)

(44) Apache Friends, https://www.apachefriends.org/index.html, (Dernière accès : 20/04/2020)

(45)Bootswatch, Cosmo, https://bootswatch.com/cosmo/, (Dernière accès : 06/03/2021)

(46)Twig, https://twig.symfony.com/, (Dernière accès : 11/03/2021)

(47)Express, Fast, unopinionated, minimalist web framework for Node.js, https://expressjs.com/, (Dernière accès : 06/02/2021)

(48)Npm, https://www.npmjs.com/, (Dernière accès : 01/03/2021)

(49) Google, Tips to improve your website, https://support.google.com/google-ads/answer/7653020?hl=en, (Dernière accès : 20/04/2021)

(50) Hartson, R., & Pyla, P. S. (2012). *The UX Book: Process and guidelines for ensuring a quality user experience*. Elsevier.

(51) Kate Moran, How People Read Online: New and Old Findings, https://www.nngroup.com/articles/how-people-read-online/ , (Dernière mise à jour : 5/04/2020), (Dernière accès : 3/05/2021)

(52) Fink, A. (2003). *The survey handbook*. sage.

ANNEXES

Annexe 1 :

Au départ, il faut produire une nouvelle base de données.

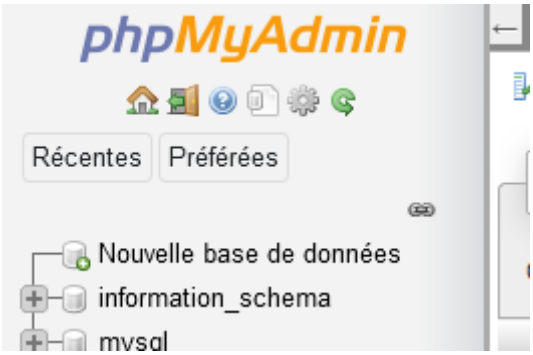

Figure 27 phpMyAdmin

Le nom de celle-ci est demandé et la base de données se crée. Une fois cette action réalisée, il est possible d'enregistrer les tables et toute la structure de cette base de données. Il existe deux alternatives.

La première est d'écrire en langage SQL afin de créer tous les éléments. Pour cela, sur la plateforme, il suffit d'aller sur l'onglet « SQL » et de noter toutes les requêtes dont il y a besoin pour au final les exécuter.

Figure 28 Requête SQL dans phpMyAdmin

La deuxième méthode est de créer manuellement les tables une après les autres en remplissant tous les noms des colonnes à chaque fois. Pour chacune des colonnes, il faut préciser le type de variable qui va être utilisé (Int, Text, …) et ajouter laquelle va être identifiant de chaque ligne.

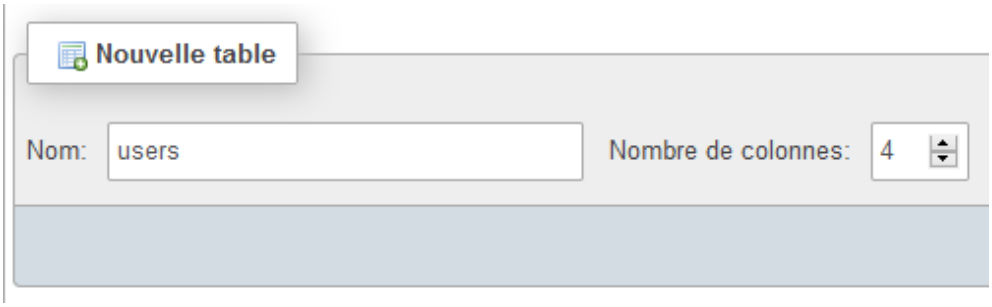

Figure 29 Création nouvelle table phpMyAdmin

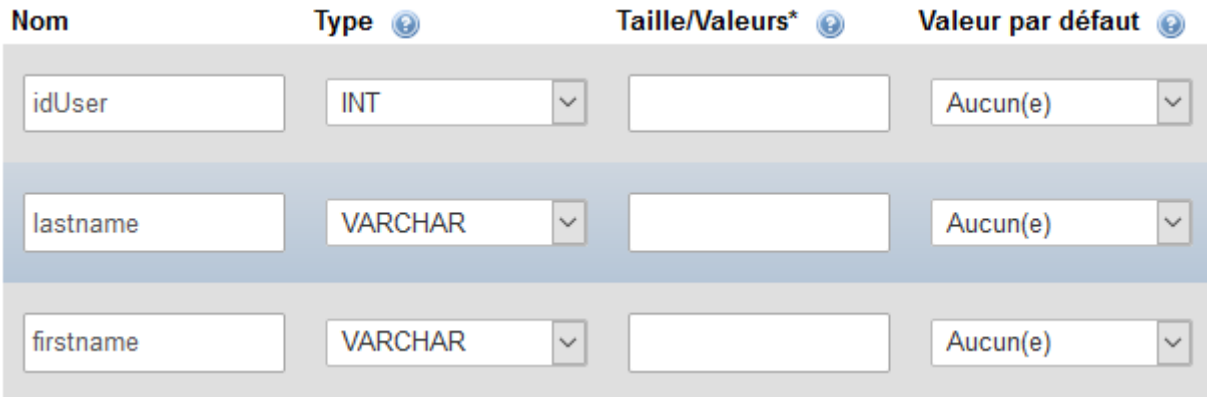

Figure 30 Ajout de colonnes à une table phpMyAdmin

Une fois que plusieurs tables sont créées, l'ajout de contraintes de clé étrangère est possible.

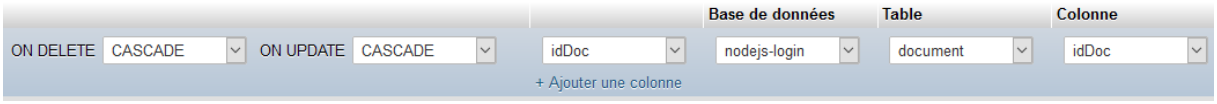

Figure 31 Ajout de contraintes à une table phpMyAdmin

Lorsque toutes les informations sont remplies, la base de données est prête à être utilisée et à répondre aux différentes requêtes. Evidemment, il est toujours possible de modifier, supprimer, ajouter des éléments (tables, attributs, contraintes, …) en utilisant une des deux méthodes expliquées juste avant.