Rapport de Stage

Fablab Cohabit

Élaboration, développement et déploiement sur un serveur Debian, d'un site internet avec le framework PHP Laravel

---

Stagiaire : Lukas BLOUIN
Maitre de stage : Pierre GRANGÉ-PRADERAS
Tuteur : DELECROIX Vincent
Année : 2022 – 2023

Licence Professionnelle ADSILLH

Sommaire

Remerciements

Je remercie Pierre Grangé-Praderas, mon maître de stage, pour l'aide et la confiance qui m'a apporté tout au long du stage. Il a su me canaliser et me faire confiance afin de mener à terme les projets qu'il m'a confiés.

Je remercie également Jean Baptiste Bonnemaison qui a été patient et à l'écoute pour répondre et m'aiguiller au mieux sur mes questionnements à propos du Fablab.

Sans oublier toutes les personnes que j'ai pu rencontrer au cours de mon stage, et notamment les membres et personnes avec qui j'ai pu travailler pour leur sympathie et leur professionnalisme.

I) Introduction

Du 1er Avril au 21 Juillet 2023, dans le cadre de la licence professionnelle ADSILLH, j'ai réalisé un stage de 4 mois dans l'association Fablab Cohabit présent à L'IUT de Bordeaux sur le site de Gradignan. Ma mission a été de continuer le développement du nouveau site de l'association, dans l'optique de le rendre exploitable début septembre 2023.

J'ai choisi cette association, car je cherchais un stage me permettant d'avoir une grande liberté sur ma manière de travailler, ainsi que sur la diversité des sujets. L'avantage des petites structures, c'est qu'on est appelé à travailler sur une plus grande diversité du sujet faisant appelle à différents métiers. Par exemple, durant le stage, j'ai été principalement développeur fullstack mais aussi administrateur système et réseaux, technicien support, mais j'ai aussi effectué la réparation et la maintenance des appareils, développer sur de l'embarquer...

Je souhaitais pouvoir mener un projet de A à Z, de la conception jusqu'au déploiement, sur des sujets faisant appel à différentes compétences. De plus, l'esprit du fablab, c'est-à-dire l'entraide et l'apprentissage par l'expérimentation, est en adéquation avec mes valeurs.

C'est donc sous la direction de Pierre GRANGÉ-PRADERAS et l'appui de Jean-Baptiste BONNEMAISON que j'ai participé à l'établissement d'un cahier des charges, ainsi qu'à la réflexion sur l'ergonomie et l'expérience utilisateur (UX). En parallèle, j'ai également participé en support à la maintenance, la mise en place de solution informatique et la résolution de problème au niveau du serveur ; Cette tâche étant attribuée à Bastien Boineau, un autre étudiant de la promo effectuant le stage en même temps que moi.

Tout d'abord, je commencerai par présenter l'association et leurs organisations, puis je détaillerai les objectifs du stage, en expliquant plus en détail le sujet et les technologies utilisées. À la suite de ça, j'exposerai le cheminement qui m'a mené à la résolution des différents problèmes que j'ai rencontré durant le stage. Pour finir, je dresserai un bilan technique et personnel de ce stage.

II) Présentation de l'association

Figure 1 - Photo de présentation du Fablab

Un fab-lab contraction de l'anglais, fabrication laboratory, est un tiers-lieu cadré par le Massachusetts Institute of Technology (MIT) et la FabFoundation. Le principe est le partage libre d'espaces, de machines, de compétences et de savoirs afin de permettre la création et l'accompagnement de projet.

Le Fablab cohabit situé à l'IUT de Bordeaux sur le site de Gradignan est une association ouverte à tous les publics (allant de l'enseignant chercheur au collégien). Il est géré au quotidien par Jean-Baptiste Bonnemaison et Pierre Grangé-Pradéras. De par la nature du lieu et de ces racines, la mentalité du fablab est orientée vers le partage des connaissances, le « do it yourself (DIY) » et la documentation ouverte afin de permettre la reproductibilité et l'ouverture des projets.

C'est dans cet esprit que le Fablab encourage et met en avant des logiciels et solution informatique libre dans le but de sensibiliser le plus de monde sur les problématiques lié à l'utilisation de logiciel propriétaire. C'est donc pour cela que j'ai privilégié l'utilisation de code et logiciel informatique libre et éviter au maximum l'utilisation de solution et/ou service propriétaire.

Par ailleurs, l'utilisation de logiciel libre offre certain avantage comme :

- Une certaine garantie au niveau de la pérennité et de la portabilité étant donné que le code du logiciel libre et sa documentation est accessible à tous.
- Une limite des risques de sécurité, car la découverte et la correction des failles est souvent plus rapide

- Moins de dépendance vis-à-vis d'un éditeur ou d'une entreprise et la possibilité de passer par d'autres acteurs pour reprendre le logiciel libre concerné
- la gratuité d'un grand nombre d'entre eux

C'est pour cela que le matériel informatique du Fablab Cohabit, en grande partie constitué de matériels récupérés, est sous la distribution Debian. L'association possède un grand de nombre de matériels parmi lesquelles ont peu trouvé (liste non exhaustive) :
des pc (bureau et portable), câble en tout genre, outils d'électronique de base (poste à souder, oscilloscope, composant électronique ...), électronique embarquée (raspberry pi, arduino, esp32 ...), machines (imprimante 3D, découpeuse laser ...) ...

Il est d'ailleurs à noter que les locaux sont partagés avec deux entités bien distinctes, le Fablab et le Technoshop. Chacun remplissant un rôle précis :

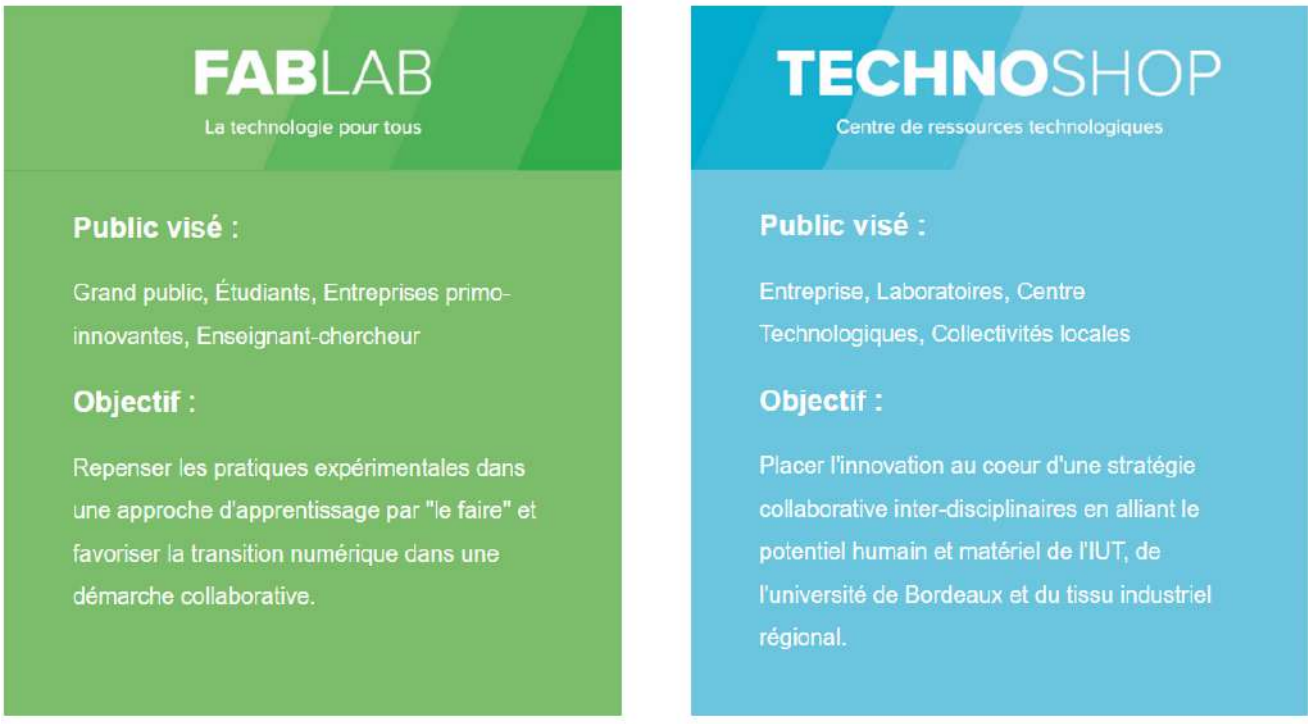


Figure 2 - Schéma d'explication des 2 entités

Le Fablab est organisé de la manière suivante :

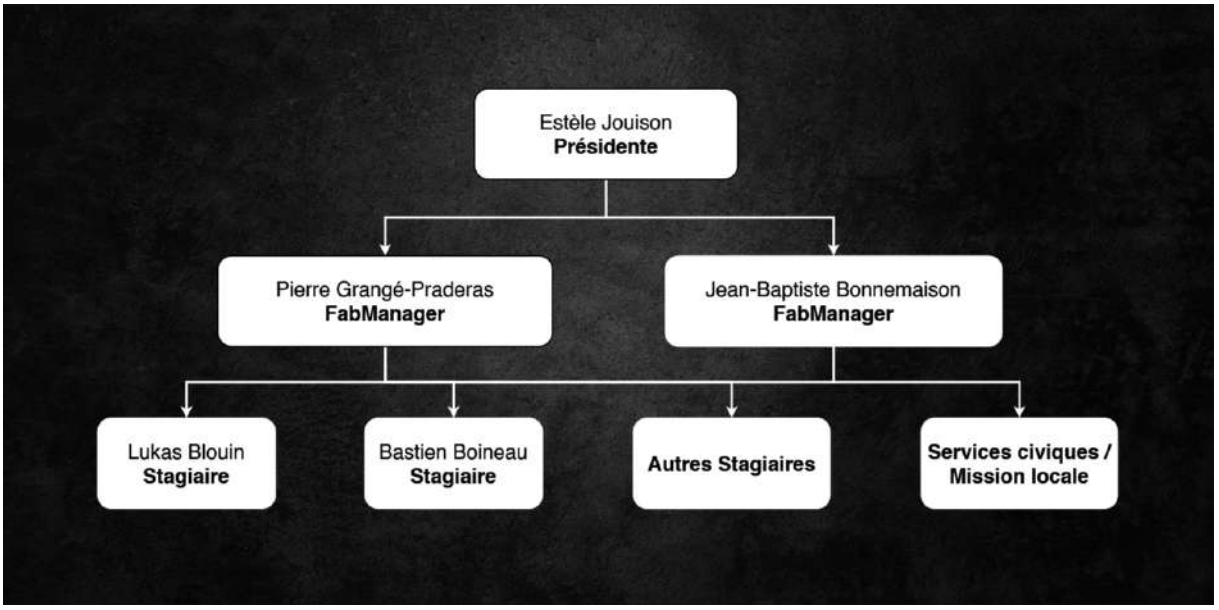


Figure 3 - Organigramme du Fablab Cohabit

La présidente de l'association est Estèle Jouison et le Fablab est géré par Pierre Grangé-Praderas et Jean-Baptiste Bonnemaison. Le Fablab fonctionne principalement sur le travail des stagiaires, services civiques, missions locales ou encore bénévoles. Par exemple, durant le stage, en simultané, ont été 7 à 8 stagiaires, 2 personnes en services civiques et 2 bénévoles actifs.

III) Présentation de la mission

1) Contexte

Le site internet actuel du Fablab (voir Annexe 1) a été créé avec WordPress et le but de ce site est de présenter le Fablab et les services qu'il propose afin de retrouver toutes les informations utiles en un seul endroit.

L'un des problèmes majeurs du site est qu'il a été fait via des "drag and drop" de composants Wordpress, ce qui a pour conséquence d'avoir un site qui manquer de cohésion graphique. Un autre problème qui a été soulevé, est que la vidéo de présentation a mal été optimisée en taille et en performance, ce qui a pour effet de ralentir le temps de chargement de la vidéo sur les connexions internet à bas débit, mais aussi qu'une fois lu, cela ralentit les performances de l'appareil. À titre

d'indicatif, elle utilise 80% des performances des pc portable du Fablab. Le site de surcroît n'a pas été mis à jour depuis quelques années.
En outre, L'IUT de Bordeaux héberge le site internet sur leurs serveurs et possède également le nom de domaine, réduisant ainsi le champ d'action et ralentissant les procédures.

C'est pour toutes ces raisons que la décision a été prise de développer un tout nouveau site internet sans Wordpress et de l'auto-héberger sur leur serveur. D'une part, cela permet à des stagiaires d'en apprendre davantage sur le développement web et l'administration système et réseaux, mais aussi d'avoir un plus grand contrôle sur la gestion des données pouvant être récolté par les GAFAM. Cela permet aussi de réduire les impacts écologiques en développant un site qui utilise moins de ressources et qui est adapté à leurs besoins.

À terme, le but est aussi de se séparer du logiciel Redmine qui permet actuellement la documentation des projets, des machines, rédaction de tutoriels en tout genre, présentation des utilisateurs du Fablab et la rédaction du carnet d'expérience.

Le cahier des charges du site internet a été constitué tout au long du stage, en voici une synthèse :

- Présenter de manière accessible le concept et les activités du Fablab.
- Fournir les informations cruciales, comme les horaires, les coordonnées et les modalités d'adhésion.
- Établir la possibilité pour les utilisateurs de créer et de gérer leur compte personnel.
- Exposer les projets en cours et les projets achevés au sein du Fablab.
- Mettre à disposition des fonctionnalités de réservation des équipements (principalement la découpeuse laser).
- Instaurer un système de diffusion d'actualités afin de tenir les utilisateurs informés.
- Autoriser la création et la consultation de portfolios individuels par les utilisateurs.
- Faciliter la constitution d'un journal d'expériences pour chaque utilisateur.
- Concevoir les outils nécessaires à l'administration complète du site.
- Élaborer une documentation dans le but d'assurer le suivi du développement.
- Minimiser les requêtes du site vers des serveurs externes.
- Réduire au maximum l'utilisation de codes JS
- Permettre l'utilisation du site sur plusieurs plateformes (responsive design)

- (facultatif) Intégrer un mécanisme de suivi des paiements relatifs aux adhésions.
- (facultatif) Rédaction des descriptions des machines
- (facultatif) Élaborer un design en adéquation avec les valeurs du Fablab.
- (facultatif) Système de réservation des formations
- (facultatif) Ajout d'un système pour commenter et de liker les projets.
- (facultatif) Unification du système d'authentification des services du Fablab

Une première tentative de site a été faite par Pierre, il y a 3 ans, mais n'étant pas développeur web d'une part et n'ayant pas assez de temps, le site a été mis de côté (voir Annexe 2)

Pierre à ensuite confier la tâche, l'année dernière, à un stagiaire, de développer le site internet. C'est ce projet-là que j'ai repris lors de mon stage. Je me dois d'évoquer, en premier lieu, que ce stagiaire était non-voyant, il a donc fait de son mieux sur la partie frontend. De plus, il a effectué son stage à distance, il n'a donc pas tout fait compris le cahier des charges que Pierre et Jean-Baptiste avaient établies avec lui. Ce qui a eu pour conséquence de rendre un produit qui ne correspondait pas tout à fait aux attentes.

Et pour finir, ce stagiaire n'a ni documenter son travail (carnet d'expérience), ni créer une documentation du projet. Ce qui m'a énormément ralenti durant mon stage, en plus de la découverte du framework Laravel, que je n'avais jamais utilisé.

2) Environnement

J'ai effectué le stage au sein même des locaux du Fablab, ce qui m'a permis de mieux cerner et comprendre le Fablab. J'ai travaillé en compagnie de Bastien Boineau, ainsi que d'autres stagiaires, bénévole et usager du Fablab, ce fût très enrichissant. Il est à noter que tout au long du stage, avec Bastien, nous nous sommes entraidés, il se peut donc que l'on évoque certain sujet ou situation commune.

Le projet que j'ai repris utilise principalement le framework PHP Laravel qui est basé sur l'architecture Model View Component (MVC). Ce framework possède plusieurs avantages tels que la prise en charges des normes de sécurité, le système de routage ou biens encore l'ORM (Object-Relational Mapping) Eloquent

qui facilite l'interaction avec la base de données. Le site utilise également le framework CSS Boostrap.

Pour ce qui est de mon environnement de développement, j'ai travaillé depuis mon pc portable sous Windows 11 via l'éditeur de code Visual Studio Code, afin de simplifier le développement qui a été effectué via WSL (Debian). Le projet utilise la Base de Donnée (BDD) MySQL qui a ensuite été migré durant le stage vers PostgreSQL.

J'ai par ailleurs administré le serveur du Fablab qui est hébergé par le fournisseur d'accès à internet associatif, Aquilnet. Le serveur est un pc de récupération sous Debian qui a été placé dans les locaux d'Aquilenet, situé dans le centre de Bordeaux.

Mon carnet d'expérience a été consigné sur Redmine et le code ainsi que la documentation se trouve sur le git du Fablab (Forgero). L'accès au code et à la documentation du site internet du Fablab est pour l'instant non publique.

IV) Mise en Œuvre et Développement du Projet

1) Découverte du projet

Lorsque je suis arrivé, on m'a présenté assez succinctement le Fablab, les membres et, les activités qui y sont faites. C'est ensuite que j'ai découvert l'environnement informatique (service, infrastructure, matériels…) du Fablab et par la suite que l'on m'a présenté plus en détail le contexte et l'état du site.

Le site internet était déjà hébergé sur le serveur du fablab sur un nom de domaine de test. De par la configuration du site, il s'avère que le site n'était accessible que lorsqu'on ajoutait une redirection du nom domaine vers l'IP du serveur dans le fichier host. C'était dû à un problème lié au renouvellement du certificat d'autorité.

Bien que fonctionnel, le site internet n'était pas attrayant visuellement, l'utilisation d'une template permis néanmoins de garder une certaine cohérence. Malgré le contexte, je trouve que le précédent développeur à fait un travail plus que correcte.

Malheureusement l'état le site n'était pas en état d'être publié, comme vous pouvez le voir dans l'Annexe 3. Le site présente de nombreuses fautes d'orthographe, des textes de remplacement ou originaire de la template sont encore présents ("un text", "lorem ipsum" …), l'agencement des éléments du site n'est pas homogène, le style graphique a besoin d'être revu…

Le travail était donc immense et ce n'était pas la fin de mes déconvenues, car après avoir récupéré le code du site, aucune documentation n'était présente, ni même quelques notes afin de lancer le projet en local.

2) Préparation de l'environnement de développement

J'ai donc dû tâtonner et me renseigner sur le fonctionnement du framework dans le but de faire fonctionner le site en locale et pouvoir développer. On m'a d'ailleurs proposé de développer le site directement depuis le serveur, mais je n'étais pas à l'aise à l'idée de développer de cette façon. Certes, cela représente des avantages, comme l'environnement qui était déjà configuré, mais j'avais peur des potentiels problèmes de latence et cela voulait dire que j'étais dépendant d'internet pour développer. Je n'étais aussi pas à l'abri de problème de connexion d'internet, d'où moins choix évident de développer en local.

Le code du site avait été déposé sur Gitea avant le déploiement sur serveur et il n'y avait pas d'historique des versions qui aurait pu m'aiguiller sur le fonctionnement du site. Après quelques heures, à installer les dépendances avec composer, installer PHP et corriger les nombreuses erreurs que j'ai rencontrées, comme l'utilisation d'une vielle version de composer, ce qui générait pas mal d'erreur, ou encore l'installation successive des modules PHP manquant au fur et à mesure des erreurs composer ou bien la génération de la clé de chiffrage nécessaire au fonctionnement du site.

Après avoir résolu ces problèmes, j'ai pu lancer le serveur local de développement, à l'aide d'une commande que j'ai trouvé dans la documentation de Laravel, et je me suis retrouvé face à une erreur Laravel :

Figure 4 - Capture d'écran de la trace d'exécution de Laravel

Je me suis rendu compte que j'avais oublié d'installer une BDD et de la peupler avec les tables nécessaires au fonctionnement du site. À ce moment-là, j'ignorais qu'il existait des scripts écrit en PHP avec l'ORM Eloquent dans le projet qui permettait de générer les tables avec la commande "`php artisan migrate`". Cela n'aurait d'ailleurs pas était suffisant, car la version que j'utilisais pour développer n'était pas la dernière. Des hot fixes, sur le code et la BDD avait été directement fait sur le serveur et n'avait pas été commit sur Gitea. J'ai donc décidé de faire une copie de la base de donnée du serveur sur mon PC. Cette manœuvre me permit d'enfin avoir une version en apparence fonctionnelle sur mon PC.

3) Nettoyage et factorisation du code Frontend

3.1) Contexte

Comme expliqué précédemment, la version du code que j'utilisais était légèrement différente, ce qui provoquait des erreurs SQL sur une partie du site. Pensant dans un premier temps que le site n'avait pas été fini et n'ayant pas pu établir un cahier des charges avec Pierre et Jean-Baptiste, j'ai durant les 3 premières semaines du stage corriger le frontend du site (CSS/HTML/JS). D'une part, cela m'a permis de faire une première entrée dans le projet et comme je n'avais pas à agir sur le back-end, cela permettait de gagner du temps en attendant l'établissement du cahier des charges.

De cette manière, je pouvais aussi repousser le moment où je devrais rentrer plus en profondeur dans le projet. Je ne connaissais ni le framework et n'avait aucune documentation pour comprendre l'architecture du projet, ce qui avait tendance à me refroidir.

C'est sur cette voie que j'ai corrigé le style des pages une à une, tout en mettant à jour les informations concernant le fablab, telle que les horaires, les informations de contact...
Je n'avais aucune ressource qui rassemblait toutes les informations au sujet du Fablab . J'ai dû récupérer les informations à partir de différentes sources (Wordpress, prototype de sites, Redmine, Pierre et Jean baptiste) afin de produire une synthèse que je pourrais retranscrire sur le site.

Cela n'a pas été simple, à force de chercher et de me renseigner, je suis parvenu à obtenir les informations qui me manquaient. Par exemple, il m'a fallu un certain temps pour retrouver et trier les réseaux sociaux encore utilisés du Fablab. C'est pour cela qu'il était important que je comprenne en profondeur ce qu'est le fablab dans le but d'au mieux le retranscrire sur le site web.

3.2) Les dépendances

Le précédent stagiaire a utilisé 3 templates différentes pour constituer la page de login, le site principale et la page d'administration. Le problème est que chaque template utilisait différentes technologies.

Site principal	**Page de login**	**Page d'administration**

Bootstrap 4.6 FontIcon SASS Plugin JS en tout genre (principalement non utilisé) Axios (non utilisé)	Tailwind CSS (utiliser par les template Laravel Jetstream)	Bootstrap 4.6.1 (intégrer au CSS de la template) FontAwesome 5 CSS Beaucoup de code et de plugin JS non utilisé

Comme vous pouvez le voir, nous avons la page de login qui utilise Tailwind CSS qui est une à Bootstrap. J'ai donc réécrit les 4-5 templates utilisés par la page de login avec Bootstrap. Nous avons également les bibliothèques d'icônes, FontIcon d'un côté et FontAwesome de l'autre. J'ai donc tranché et utilisé FontAwesome, car il proposait plus d'icônes et j'étais plus familiarisé avec celui-ci. Pour les plugins JS, j'ai dû faire des tests, consistant à enlever les plugins 1 à 1, pour vérifier s'ils étaient utilisés par le site internet.

Durant tout le stage, je n'ai cessé de faire le tri dans les dépendances afin d'éliminer celles qui sont inutilisées ou superflues. Il m'est arrivé à de nombreuses reprises de devoir faire marche arrière dans le but de réparer mes erreurs. Le développement du site a été rempli d'obstacles à franchir, mais je n'ai jamais abandonné et ce fût très enrichissant.

Voici un tableau récapitulant les modifications que j'ai apportées :

Site principal	**Page de login**	**Page d'administration**
Bootstrap 5 **FontAwesome 6** SASS **Plugin JS (réduction significative)**	**Boostrap 5** **FontAwesome 6**	Bootstrap 4.6.1 **(Suppression de JQuery)** **FontAwesome 6** **Suppression du code JS**

Vous trouverez en gras les changements qui ont été effectués. J'ai principalement mis à jour les dépendances du site principal, en commençant par Bootstrap, qui est passé à la nouvelle version majeure 5.

Il est important de peser dans la balance la quantité de travail supplémentaire que peut apporter le passage des dépendances vers leurs nouvelles versions majeures. Généralement, cela signifie qu'il y a eu un nombre de modifications pouvant rendre obsolète en partie ou totalement le code du projet.

Le passage vers Bootstrap 5 a permis de simplifier le CSS, le code JS exécuté par le site via la suppression de JQuery et facilité le développement par l'ajout de nouvelles fonctionnalités. La mise à jour, n'a provoqué que très peu dégât, en ne cassant que les menus déroulants et le système de marges. Ce qui a été rapidement réglé en remplaçant l'ancienne nomenclature par la nouvelle.

Seule la page d'administration est restée sur l'ancienne version, car le template a été développé pour fonctionner uniquement avec cette version de Bootstrap. Cela aurait nécessité de redévelopper dans son intégralité la template or, je n'avais pas le temps.

J'ai dû privilégier l'UX (user experience) à l'UI (user interface), si je voulais parvenir à rendre un site fonctionnel à la fin du stage.

3.3) Laravel Blade

À tout cela s'ajoute la mauvaise utilisation des layout et componement Blade, c'est-à-dire que le code HTML était copier-coller de fichier en fichier (y compris le header). On avait donc des différences notables entre deux components présentant des informations similaires. Il était donc nécessaire de factoriser le code afin de faciliter la maintenance ainsi que l'ajout de contenu.

Laravel Blade est le moteur par défaut de template de Laravel. Il permet de fragmenter le code d'une page web dans le but de faciliter la maintenance et l'édition de celui-ci. Pour faire simple, le "layout" est la première couche du site (contenant par exemple la barre de navigation et le footer), auxquelles on va ajouter dynamiquement le contenu qui est présent dans un autre fichier qu'on appelle "componenent". Il est également possible dans ces componements d'inclure du code PHP, à l'aide de la syntaxe du framework. À la fin, les différentes couches sont assemblées et les instructions écrites avec la syntaxe Blade sont interprétées et convertis en PHP, puis insérées dans la page finale.

Le header HTML aurait dû être généré dynamiquement à partir d'un seul fichier, ce que j'ai fait et j'y ai ajouté la génération automatique des titres, selon la page qui est générée. Il s'agit des titres affichés sur l'onglet du navigateur, avant, il n'affichait que "Laravel", maintenant, il affiche "Nom de la page - Fablab Coh@bit". Le site nécessita également quelques modifications afin de pouvoir être utilisé sur mobile (responsive design).

J'ai donc eu un gros travaille d'uniformisation des technologies utilisées, de factorisation des templates et de refonte graphique. Je devais par ailleurs faire le tri dans le code JS dans le but de réduire au maximum le code inutilisé ou facilement remplaçable par des règles CSS. C'est ainsi que j'ai refait les différentes pages statiques du site, notamment la page de contact en changeant Google Maps par une solution open source : OpenStreetMap, pour la localisation du Fablab. J'ai par ailleurs vérifié et actualisé les informations sur les moyens de transport disponible pour se rendre Fablab.

J'en ai profité pour supprimer les appels externes, en rendant les ressources accessibles en local (tels que Bootstrap, FontAwesome ...). C'est à la suite de ce travail que j'ai eu à intégrer une nouvelle fonctionnalité sur le site internet.

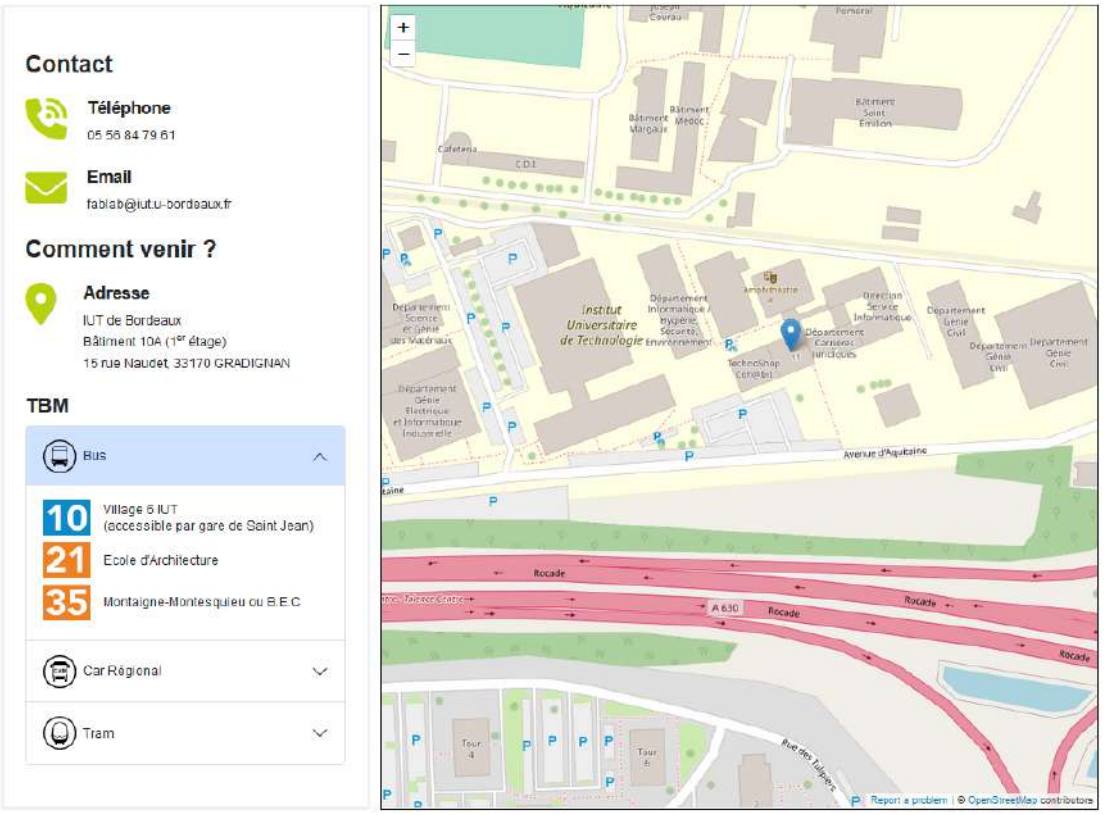


Figure 5 - Page de contact

4) Ajout d'un éditeur de texte

4.1) Contexte

À partir du moment où j'ai pu faire le point sur le site avec Pierre et Jean-Baptiste. Il a été évoqué que le site devait progressivement remplacer Redmine. Redmine est utilisé pour la rédaction de tutoriels, fiches techniques des machines,

présentation des projets, des carnets d'expériences et des portfolios. C'est ce qui constitue la principale source de documentation du Fablab.

Cependant, il présente certains problèmes significatifs :

- L'organisation de l'information a évolué de manière non structurée au fil des années, ce qui complique la recherche et la consultation.
- Le système de recherche s'avère peu efficace, rendant l'accès à certaines ressources plus complexe.
- L'éditeur de texte manque de fonctionnalités essentielles, comme la possibilité de créer facilement des sommaires ou de masquer des paragraphes sous forme d'accordéons, ce qui entrave la navigation et la lecture des pages.
- Seules une infime partie des fonctionnalités de Redmine sont utilisées, ce qui suscite des interrogations quant à l'utilisation des ressources par rapport à l'utilisation réelle. Ce qui permet d'"envisager la possibilité de migrer vers un logiciel plus léger.

Encore aujourd'hui, j'ai du mal à retrouver les informations afin de m'aider à la rédaction de ce rapport. C'est donc dans le but d'amorcer la migration vers un autre système de documentation qu'il m'a été demandé d'intégrer un éditeur de texte de type WYSIWYG (What you see is what you get), dans le but de permettre la rédaction et la consultation des portfolios et des carnets d'expérience des utilisateurs depuis le site.

C'est à ce moment-là que je me suis rendu compte que les nombreuses erreurs PHP présentes sur ma version de travail n'était pas présente sur celle du serveur. C'est pour cela qu'il a été nécessaire de faire une copie du code présent sur le serveur afin de l'intégrer à mes modifications. Forte heureusement, la fusion n'a provoqué que très peu de conflit.

Malencontreusement, j'ai commis une erreur lors de l'archive du code, ce qui eu pour conséquence de compresser chaque fichier du projet sur le serveur, rendant le site inopérant. Ce problème fût découvert et résolu seulement 1 à 2 semaines après et n'a eu qu'un impact minime.

4.2) Choix et intégration de l’éditeur

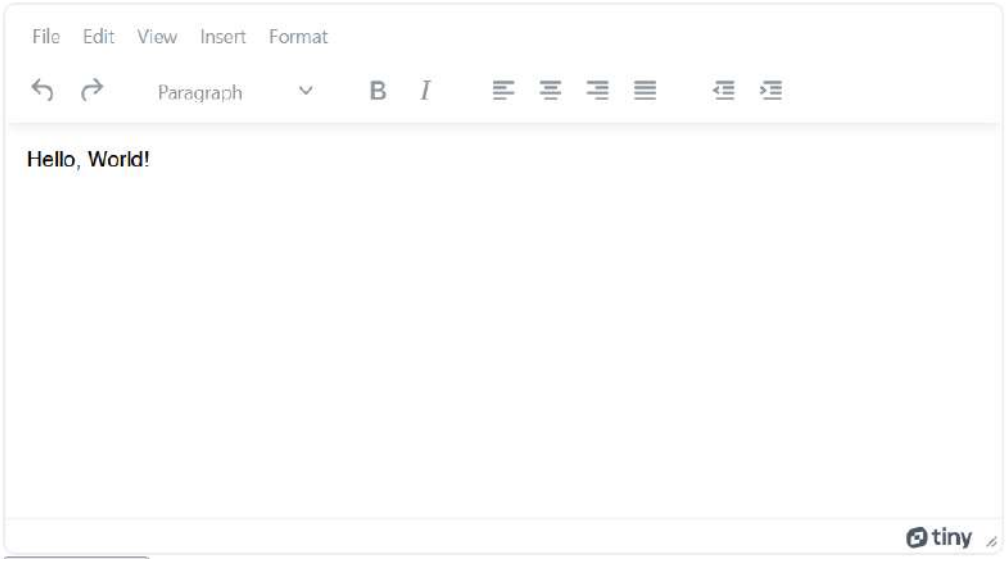


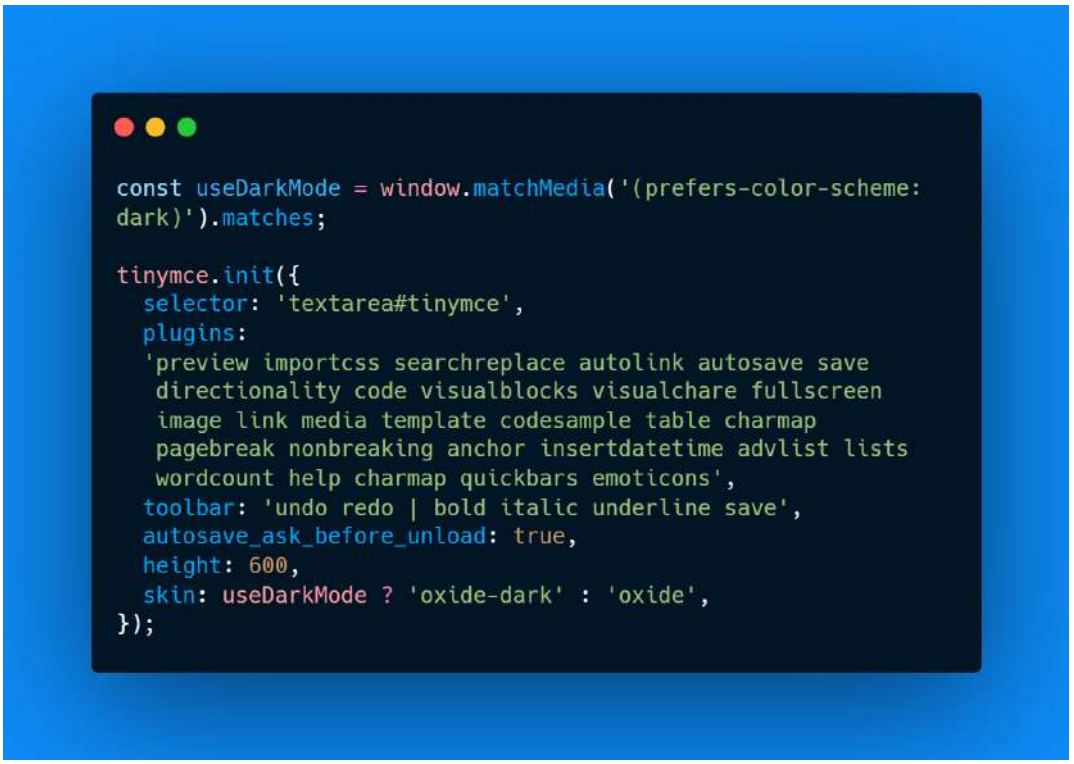
Figure 6 - Éditeur de Texte TinyMCE (configuration minimal)

Après quelques recherches sur les éditeurs disponibles, mon choix, c'est très vite porté sur TinyMCE. J'ai exploré divers éditeurs et j'ai pu constater que TinyMCE offre la plupart des fonctionnalités présentes dans Redmine. Cette similitude facilitera l'adoption par les adhérents. De plus, TinyMCE est hautement personnalisable et open source, ce qui constitue un avantage. Un atout supplémentaire est qu’il propose une intégration avec Laravel, ce qui m'a convaincu quant à son utilisation dans le projet.

Cependant, j’ai très vite déchanté lorsqu’il a fallu l’intégrer au projet. Ce qui devait me prendre 2 à 3 jours, m’a pris environ 2 semaines. D’une part, je découvrais encore tous juste le framework Laravel et d’une autre part, je n’avais pas encore trop touché au backend.

En suivant la documentation de TinyMCE, je parvins sans difficulté à l’installer. Le plus compliqué a été d’intégrer TinyMCE de manière à pouvoir enregistrer le texte et de pouvoir la restaurer. Pour ce faire, il faut faire le lien entre PHP et JavaScript afin de récupérer le texte.

4.2.1) Fonctionnalité de sauvegarde



```
const useDarkMode = window.matchMedia('(prefers-color-scheme:
dark)').matches;

tinymce.init({
  selector: 'textarea#tinymce',
  plugins:
  'preview importcss searchreplace autolink autosave save
   directionality code visualblocks visualchare fullscreen
   image link media template codesample table charmap
   pagebreak nonbreaking anchor insertdatetime advlist lists
   wordcount help charmap quickbars emoticons',
  toolbar: 'undo redo | bold italic underline save',
  autosave_ask_before_unload: true,
  height: 600,
  skin: useDarkMode ? 'oxide-dark' : 'oxide',
});
```

Figure 7 - Exemple de fichier de configuration TinyMCE

Le fichier de configuration de TinyMCE n'est ni plus ni moins qu'un dictionnaire associant un mot clé à une valeur. J'ai grandement réduit le nombre d'options pour l'exemple, mais on peut voir qu'il n'est pas difficile à configurer. C'est de cette manière que j'ai activée et configurée les différentes fonctionnalités de l'éditeur.

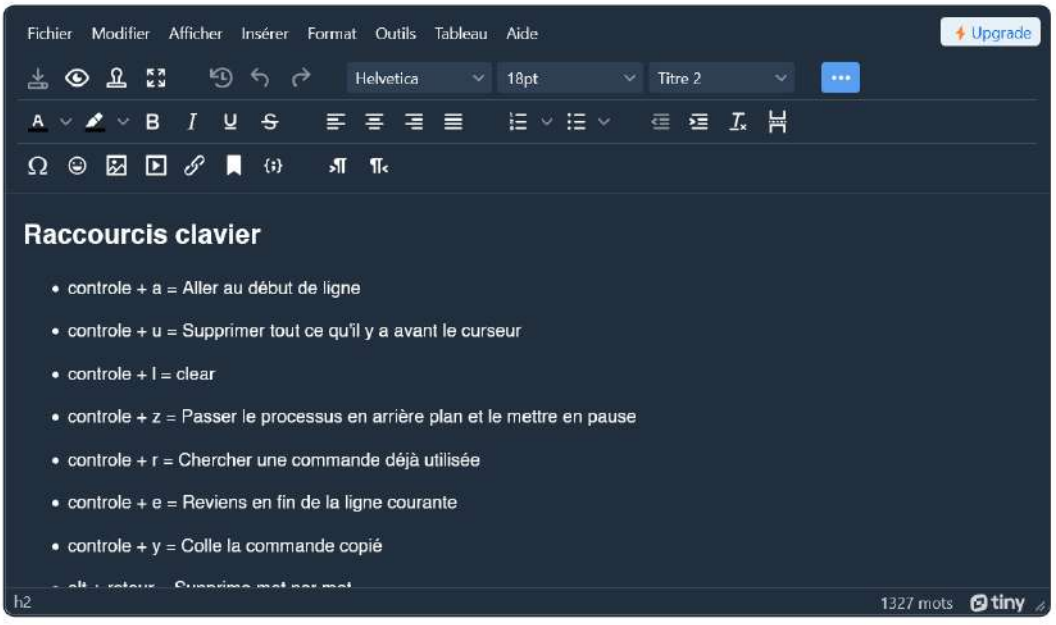


Figure 8 - Éditeur de Texte TinyMCE (configuration avancée)

Au départ, l'éditeur est assez minimal (voir au-dessus) en termes de fonctionnalités et n'est disponible qu'en anglais. Afin d'éviter de lire toute la documentation pour activer tous les plugins et ainsi gagner du temps, j'ai trouvé par hasard dans la documentation, un fichier de configuration avec tous les plugins open-source déjà activés, ainsi que de nombreux préréglages (sauvegarde automatique, thème, bar de menu prêt à l'emploi ...).

Aussi étonnant qu'il puisse le paraître, le plugin permettant la sauvegarde n'est pas activé de base. Une fois activé, j'espérai que l'intégration serait facile, mais ce ne fut pas le cas. J'ai dû utiliser l'une des fonctionnalités du framework Livewire, permettant de faire du binding.

Le terme "binding" fait ici référence à la liaison ou à la connexion entre deux éléments, la BDD et la variable PHP contenant le texte écrit dans l'éditeur. Livewire est une bibliothèque PHP conçue pour simplifier le développement d'interfaces utilisateurs interactives en temps réel.

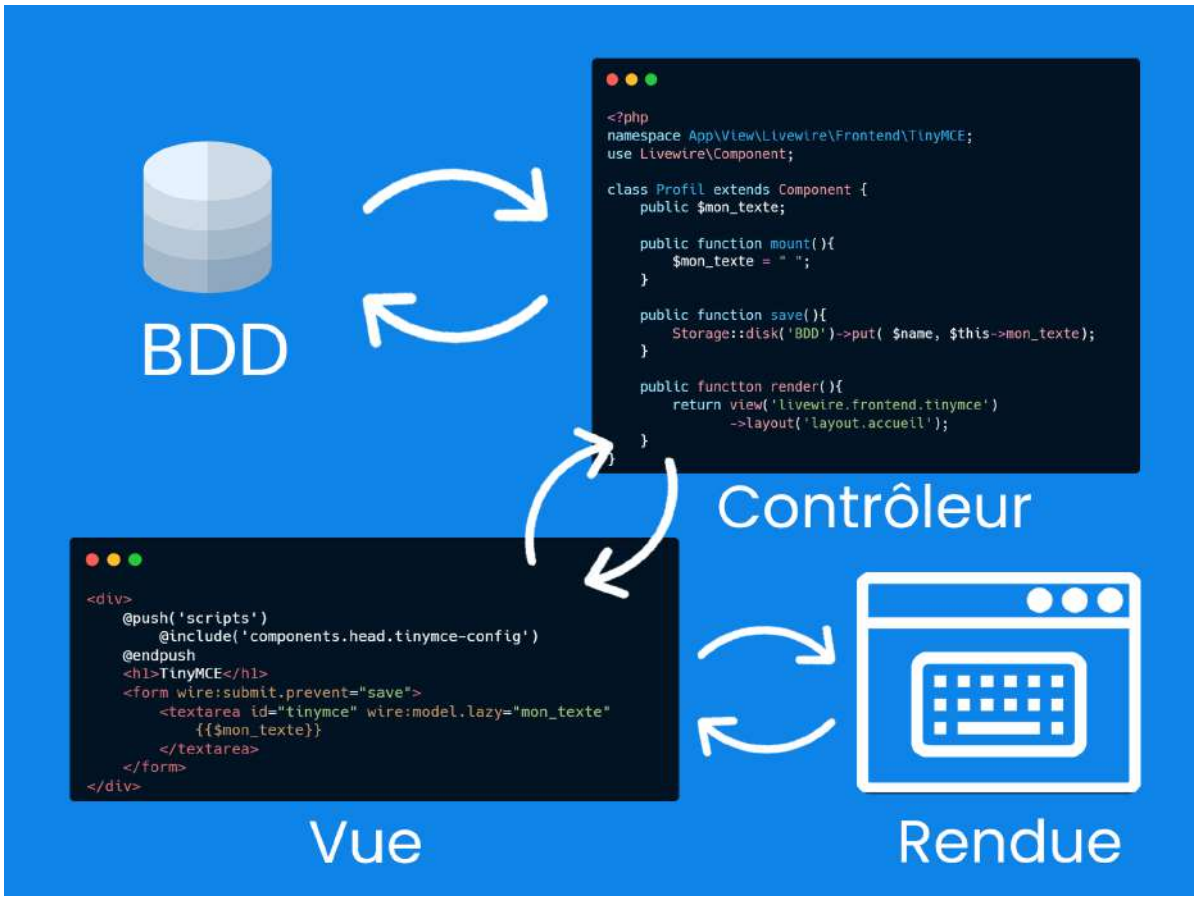


Figure 9 - Schéma expliquant la sauvegarde du texte sur la base de l'architecture MVC

Il serait donc possible de configurer la vue pour qu'elle envoie au contrôleur (patron d'architecture MVC) de manière périodique, le contenu de l'éditeur et pour qu'ensuite cela puisse être enregistré dans la base de données.

Dans la pratique, c'est tout autre chose, je me suis heurté à un problème, d'une part au moment où je l'ai fait, j'avais moins de connaissance sur le framework et donc beaucoup moins de recule sur son fonctionnement. Il s'avère que le fonctionnement de TinyMCE fait que le binding ne pouvait marcher.

Pour inclure l'éditeur dans une page, soit on l'inclut directement avec une balise avec Blade, de la manière suivante **`<x.livewire.tinymce>`**, soit dans mon cas, il faut configurer au préalable un mot clé, qui sera détecté, soit par un id, soit par une classe. La balise qui contient ce mot clé sera remplacé par une frame contenant l'éditeur.

Figure 10 - Code HTML après chargement de la page

Sauf que moi, j'ai placé l'id sur une balise **`<textarea>`** puis j'ai mis l'option **`wire:model="ma_variable"`** pour bind le contenu de la balise. Cette balise est ensuite désactivée avec le mot clé et l'éditeur est inséré ensuite dans la page, et il n'y a donc aucun moyen de signaler à l'éditeur de texte d'envoyer le contenu au contrôleur. Pour permettre la sauvegarde, il fallait également l'insérer dans un formulaire HTML. C'est pour cela que j'ai mis du temps avant de comprendre pourquoi je n'obtenais pas le texte que j'écrivais.

J'ai donc essayé diverses méthodes et chacune d'entre elles ont échoué. À force d'essayer, j'ai fini par trouver dans la documentation de TinyMCE une fonction permettant de renvoyer le texte de l'éditeur.

Il ne me restais plus qu'à l'envoyer au contrôleur via l'événement suivant : `wire:submit.prevent="ma_fonction($mon_texte)"` mais pour une raison que j'ignore encore aujourd'hui, l'événement n'est pas toujours appelé. J'ai donc utilisé son équivalent Javascript : `onsubmit`. Il ne me restait plus qu'à transmettre le contenu. Pour ce faire, j'ai créé un événement dans le contrôleur et je l'ai appelé, côté Javascript avec Livewire. Ce n'est pas parfait et sûrement perfectible, mais sur le moment cela m'a permis de débloquer ma situation.

```
<div class="row">
    @push('scripts')
        @include('components.head.tinymce-config')
    @endpush
    <div class="col-8">
        <h3>Carnet d'expérience</h3>
    </div>
    <form wire:ignore onsubmit="Livewire.emit('contentSaved','carnet_experience.html',
          tinymce.get('carnet_editor').getContent());">
        <div>
            <textarea id="carnet_editor" class="tinymceinstance">
                  {{$this->get_file('carnet_experience.html')}}
            </textarea>
        </div>
    </form>
</div>
```

Figure 11 - Code fonctionnel permettant la récupération et la restitution du texte dans TinyMCE

Si je résume, lorsque l'on charge la page, le contenu du document est mis dans la `textarea`. Celui-ci est ensuite interprété et affiché par l'éditeur. Une fois les modification apportait, le formulaire est soumis via le bouton de sauvegarde de l'éditeur, l'événement `onsubmit` est appelé et l'événement Livewire de sauvegarde (`contentSaved`) est émis avec le nom du document et son contenu, il est ensuite récupéré par le contrôleur qui le sauvegarde en local sur le serveur.

Une fois ce problème résolu, j'ai permis l'ajout d'images, avec tout autant de difficulté, et après avoir effectué des tests en condition réelle, j'ai découvert un bug. Je n'ai pas réussi à résoudre ce bug, la cause est problème de déconnexion de l'utilisateur au bout d'une ½ heure. Cette déconnexion empêche l'utilisateur de sauvegarder, faisant perdre les modifications faites durant ce laps de temps.

5) La réservation des machines

5.1) Contexte

Une fois plus à l'aise avec le projet, je me suis occupé de la fonctionnalité la plus importante : la réservation des machines. Lorsque j'ai récupéré le projet, cette fonctionnalité était déjà présente et fonctionnelle. Le seul problème étant que les éléments de l'interface de réservation n'était pas ergonomique, nuisant à l'UX et que comme souligner au début du rapport, l'UI était à refaire (voir Annexe 5 et Annexe 6).

On retrouve parmi les exigences :

- L'affichage d'un catalogue des machines présentes au Fablab
- Fiche de présentation (courte description, image de présentation).
- Formulaire de réservation accessible aux utilisateurs connecté.
- Affichage des réservations sur une période de 2 semaines.
- Affichage des réservations des dernières réservations (objet de la réservation et personne).
- Possibilité de limiter le nombre d'heures réservable par semaine (découpeuse laser) sauf pour les administrateurs.
- Création d'outils permettant d'administrer les réservations.

5.2) Modification de l'interface

Horaire/Jour	Monday 2023-09-04	Tuesday 2023-09-05	Wednesday 2023-09-06	Thursday 2023-09-07	Friday 2023-09-08
10-11	RÉSERVER	RÉSERVER	RÉSERVER	RÉSERVER	RÉSERVER

Figure 12 - Capture d'écran de l'ancien calendrier de réservation

Dans un premier temps, il m'a été demandé de remplacer les boutons de sélections des horaires, car le choix des couleurs et la multiplication des boutons surchargeait visuellement l'œil et pouvez donner un mal de tête. Il y a d'ailleurs, un fait assez comique, en fixant 1 min le calendrier de réservation, par effet d'optique, cela faisait apparaître des point noirs entre les boutons de réservation.

J'ai donc changé le style et la disposition des éléments de la page, de cette manière, on voyait plus distinctement les champs du formulaire. Pour ce qui du

calendrier, j'ai mis une couleur moins agressive et j'ai remplacé les boutons par des checkbox qui apparaissent uniquement lorsqu'ils sont survolés avec la souris.

Figure 13 - Capture d'écran du calendrier de réservation

L'affichage des réservations était sur un fond gris, une des suggestions qui m'a été faite est d'afficher les réservations avec une couleur différente par utilisateur. Je l'ai donc fait à l'aide d'une fonction qui génère un nombre aléatoire compris en 0 et 255 pour les champs RVB (Rouge Vert Bleu), qui constituent la couleur. Pour éviter que la couleur change à chaque fois que l'on recharge, j'ai mis en seed l'id de l'utilisateur. Une seed (ou graine en français) est une valeur initiale spécifique que l'on fournit à l'algorithme de génération et qui détermine le point de départ de la séquence de nombres utilisé pour générer notre nombre. De cette manière, je n'ai pas à enregistrer la couleur dans le profil utilisateur et j'obtiens toujours le même résultat.

Un autre problème du calendrier, c'est que les jours était affiché en anglais. Afin de résoudre le problème, j'ai changé la langue interne de Laravel sans succès. J'ai également essayé de changer la variable globale PHP qui gère la langue, appliquer dans le constructeur des dates la timezone, vérifier que la variable globale du serveur gérant la langue est bien paramétré en français.

La solution qui à finalement fonctionner et qui est selon moi loin d'être idéale bien qu'efficace, a été de faire correspondre les jours et les mois en anglais avec leur traduction française via un dictionnaire. On récupère les textes générés pour l'affichage et on substitue juste avant les mots en anglais.

5.3) Analyse du code du contrôleur

Je ne l'ai pas précisé plus tôt, mais le code du contrôleur était assez complexe à comprendre. Le calendrier n'est pas le fruit d'une template ou d'un plugin, mais bien d'une production personnelle. Si j'avais eu une documentation, ou ne serait-ce qu'un commentaire, expliquant le but de chaque fonction, j'aurais sûrement perdu moins de temps. Étant donné que les réservations sont fonctionnelles d'un point de vue, logique, j'ai donc pris mon courage à deux mains et j'ai analysé le code pour comprendre son fonctionnement.

La méthode qui m'a permis de comprendre au mieux chaque fonction, dans un premier temps, est l'affichage, à chaque étape, des valeurs manipulées. Puis dans un second temps, la factorisation et l'optimisation de tout ou partie des fonctions, ce qui m'a permis de m'approprier le code. Je ne vais pas détailler les étapes qui m'ont permis d'obtenir le calendrier actuel, car je ne me souviens tout simplement pas de tous les détails, mais aussi que cela serait beaucoup trop long. Je me contenterai simplement d'évoquer le bug qui m'a fait perdre le plus de temps.

Je n'étais pas très loin de la version actuelle et j'avais besoin de changer la manière d'on est récupéré les informations des séances sélectionnées. Je ne me souviens plus exactement des raisons, mais je sais que c'était nécessaire, afin de réduire significativement le code et améliorer les performances.

Ce changement se trouve au niveau de la création de la structure de donnée qui accueille les données du calendrier. J'avais pour idée d'utiliser directement la date sous forme d'objet à la place de générer les différents textes utilisé dans le calendrier. Cela aurait permis de mettre moins d'informations dans le tableau et ainsi facilité le parcours de la structure.

La structure de donnée est une matrice à deux dimensions, en abscisse, on a les horaires et en ordonnée, on a les jours.

Figure 14 - *Structure de donnée du calendrier (en abscisse)*

Si l'on prend la première valeur, on a donc un tableau qui représente le calendrier des 2 prochaines semaine. Il contient des dictionnaires, qui eux-mêmes contiennent les informations de chaque jour pour l'horaire 10-11h.

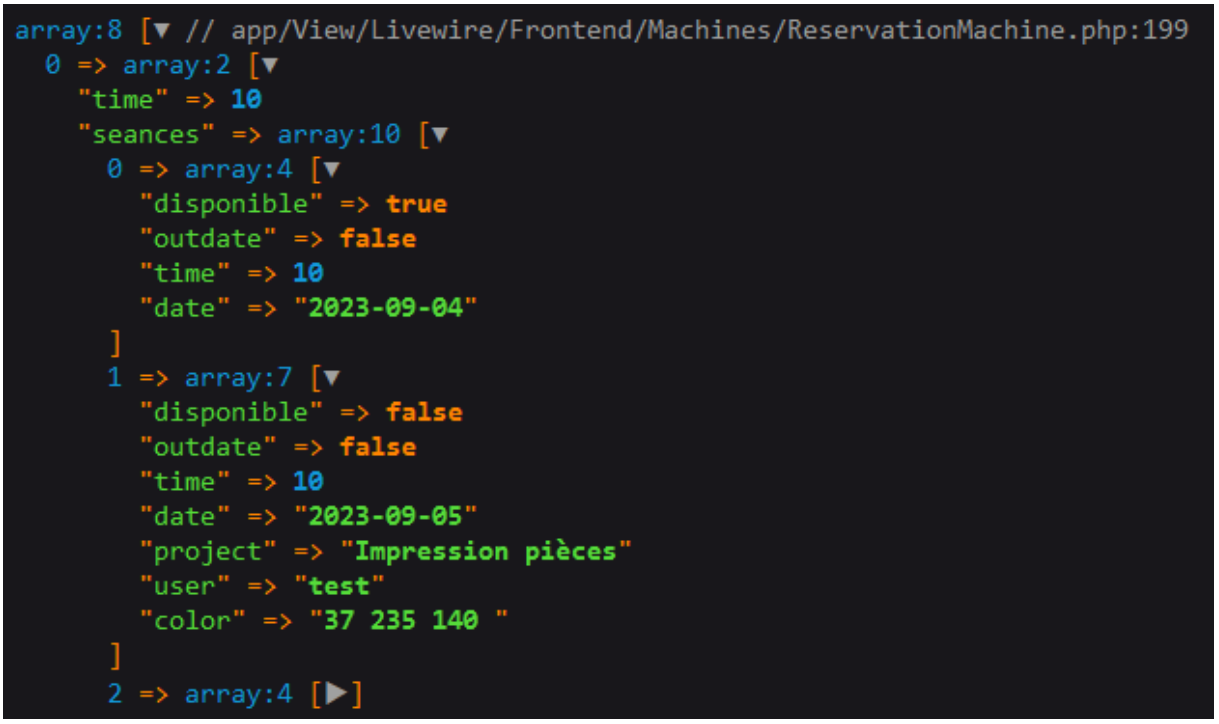


Figure 15 - Structure de donnée du calendrier (en ordonné)

Dans le dictionnaire “seances”, on a une entrée par jour. Chaque entrée contient les informations suivantes :

- Si cette horaire est disponible
- S'il est réservé par quelqu'un
- L'horaire et la date
- Nom et auteur de la réservation (si réserver)
- Couleur de la case (si réserver)

Vous noterez qu'un horaire peut ne pas être disponible soit dans le cas où il est réservé, soit dans le cas où il est indisponible pour diverse raison (Fablab fermé, machine en maintenance ...). La structure n'est pas évidente à comprendre et certaine information sont redondantes (comme l'horaire), mais c'est négligeable au vu du temps impartit et de l'impact minimum sur l'expérience utilisateur.

Pour revenir au sujet principal, je souhaitais mettre la date sous sa forme d'objet et non uniquement en texte. De cette manière, j'avais bien plus d'informations et j'avais également accès aux fonctions qui permettent de générer les différents textes à partir de la date (en toute lettre, jj/mm/aa ...). Il s'avérait que j'avais besoin d'avoir la date sous différente forme et je n'avais pas envie de multiplier les informations dans la structure de donnée.

C'est à ce moment-là que mon calvaire commença. Lorsque j'accède à la page et que le calendrier est généré et affiché, je n'ai pas de problème. C'est uniquement lorsque que j'appuie pour sélectionner un horaire, que j'obtiens une erreur pour le moins bizarre :

“Impossible d'accéder à la fonction to_format(“jj/mm/aa”) d'un tableau”

La ligne concernée par cette erreur se trouve dans la vue, plus précisément au moment où j'accède à la date (objet) pour générer le texte. Ce n'était pas la première fois que je l'ai rencontré, car durant la modification, j'avais oublié certaine variable qui était encore des tableaux contenant les textes pré-générés. Sauf que cette fois-ci, après plusieurs vérifications, il n'y avait aucune erreur, ce n'était donc pas un oubli. Fait encore plus étonnant, cela n'intervient qu'une fois que l'on clique pour réserver, pas avant. Et je tiens à préciser que ce n'est aucunement lié au code qui est exécuté à l'appui du bouton.

Vous êtes, sûrement, en train de vous dire que c'est une erreur anodine, ou alors que c'est trop long et que vous êtes complétement perdu dans mon récit, dans ce

cas-là, j'en suis désolé 😅. Je plaisante bien sûr, c'est toujours plus agréable à lire lorsqu'il y a des traits d'humour, mais je m'égare …

Reprenons ! Ce n'est pas anodin, car si le calendrier est généré une 1ère fois sans problème et que ce n'est pas la conséquence directe de l'appui sur le bouton. Pourquoi est-ce que j'obtiens une erreur sur un code qui a fonctionné quelque seconde plus tôt. L'erreur est donc générée durant l'exécution du code et concerne forcément un fonctionnement interne de mon code ou de Laravel. Je ne vais faire le suspense plus longtemps, il s'agit d'un problème assez particulier qui n'a été quasiment jamais été évoqué sur internet.

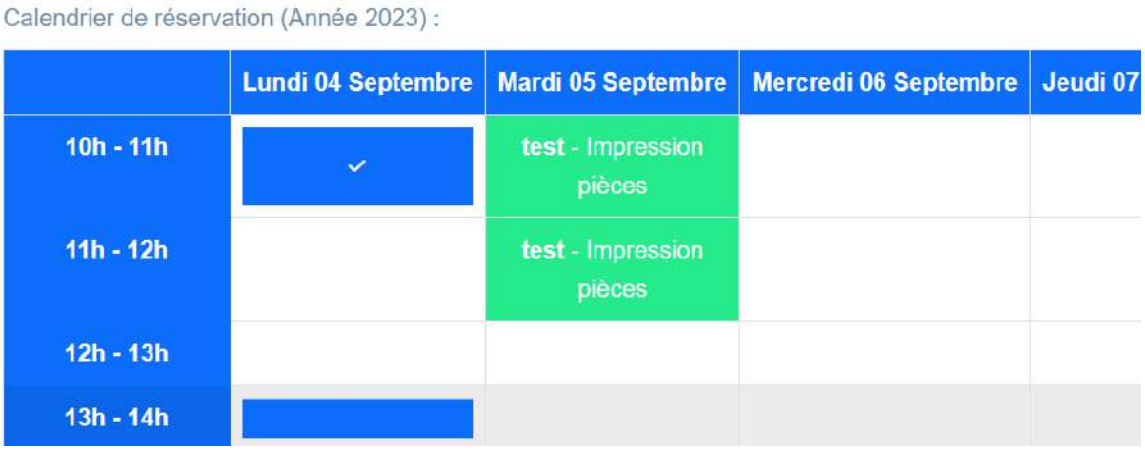


Figure 13 - Calendrier de réservation (de nouveau) pour l'avoir sous les yeux et ainsi mieux visualisé

Lorsque j'appuie sur le bouton, cela a pour effet de changer la case par une check-box pour montrer visuellement à l'utilisateur que la case a été sélectionnée. Cette action a pour effet de régénérer la portion de page concerné, ici le calendrier pour ensuite modifier dynamiquement la page. C'est tout simplement le fonctionnement de Livewire, cela permet de faire des pages dynamique avec PHP sans avoir à recharger complétement la page. Sauf qu'ici, précisément au moment où la page et régénérer une deuxième fois, la date qui était un objet est devenu un tableau.

Au début, je ne comprenais pas les raisons pour lesquelles Laravel ou Livewire convertissais l'objet en tableau. Je ne le comprends pas encore tout à fait, car il s'agit d'un fonctionnement assez fin du framework. Mais pour faire simple, cela permet de généraliser la manière d'interpréter et d'afficher les objets sur la page. Normalement, cela n'aurait pas dû impacter cette variable, étant donné que je la convertis en texte avant de l'afficher, mais ici, c'était le cas… J'ignore même s'il

s’agit d’un bug du framework ou uniquement d’une erreur de ma part, mais la conséquence est la même, je dois me résoudre à pré-générer mes textes.

Si j’ai refusé obstinément cette option, c’est parce que je ne comprenais pas pourquoi mon code pouvait fonctionner une première fois, mais pas une deuxième, sans que le code ait était modifié entre-temps. Cette erreur est pour moi incompréhensible et m’a demandé beaucoup de patience, d’analyse et de recherche afin de connaître les tenants et les aboutissants. Le pire étant que j’ai rencontré le même type d’erreur, à 3 reprise durant le développement, toutes différentes et incompréhensibles.

6) Déploiement sur serveur

6.1) Changement de serveur

Le Fablab possède un serveur qui est vieux un pc DELL de récupération qui a été placer au sein des locaux d'Aquilenet, un fournisseur d'accès à internet associatif bordelais. Le serveur est sous Debian et il héberge les services du Fablab tel que Redmine, Forjero et le site que je développe. La particularité de ce serveur est qu'il a deux alimentations, ce qui provoque une forte consommation (60W en sommeil) comparer à l'utilisation que l'on fait. La première alimentation alimente la carte mère et l'autre est dédié au SSD (ce qui est overkill).

Figure 16 et 17 - Ancien serveur et Câble d'alimentation SATA vers USB

Durant le stage, on a réfléchi à moyen de réduire la consommation, c'est pour cela que l'on a créé à partir de câble récupération des câbles USB vers SATA, afin d'enlever une alimentation et donc diviser la consommation. Il était question de venir les installer dans le courant de la semaine, mais l'opération a été un peu précipitée par la panne soudaine du serveur. Cette panne à paralyser ainsi les services et l'accès distant au serveur SSH, rendant impossible le dépannage à distance.

Lorsque nous sommes intervenus avec Bastien, il nous a été impossible d'identifier la panne, l'erreur provenait de l'impossibilité au démarrage du système de monter les SSD. Le système démarrait donc dans un mode de récupération. Les SSD fonctionnant parfaitement, on a d'abord mis la faute, à cause d'un message d'erreur au démarrage, sur une mise à jour Linux qui aurait pu être incompatible avec la version actuelle du BIOS du PC. Après avoir mis à jour le BIOS, le message a bien disparu, mais pas notre problème, on a donc

supposé que cela venait soit d'une mauvaise configuration du fichier fstab, soit d'un problème d'alimentation. Le fichier fstab est un fichier de configuration sous Unix/Linux qui contient une liste des disques utilisés au démarrage et des partitions de ces disques. On a donc installé les nouveaux câbles et retirer l'alimentation, mais le PC ne voulait plus démarrer, après quelques tests infructueux, on a commencé à voir un peu de fumée et une odeur de composant cramé. On a vite conclu, qu'il venait de tirer sa dernière révérence.

Il était donc temps de changer le serveur par un autre, pour cela, on a pris un pc du fablab et on a réinstallé les services un à un. Comme nous avions encore les données intactes de l'ancien serveur, nous avons restauré d'abord en local, les différents services. Ainsi que le site internet, pour ce faire, je me suis basé sur la documentation que j'ai pris le soin de faire tout du long du développement. J'eus sans trop de problème reconstituer les instructions pour déployer le site internet.

Une fois que l'on c'est assurer que les services était stable et pleinement opérationnelle, nous avons remis le PC dans les locaux d'Aquilenet. Je vous ai épargné les heures de débogage, de copie des données, de restauration qui fût fastidieuse, mais très formatrice. Si vous souhaitez plus de détail, veuillez vous référer au rapport de stage de mon collègue Bastien Boineau.

6.2) Migration BDD de Laravel

Durant la réinstallation du serveur, on s'est posé la question de l'utiliser, d'avoir deux BDD sur le serveur. PostgreSQL est utilisé par tous les services du Fablab, excepter le site web qui est sur MariaDB. Il serait plus simple d'administrer une seule base de donnée et on s'était également dit que cela réduirait la consommation des ressources du CPU. Ce qui après le coup, n'était pas vraiment fondé, car les BDD n'utilisait que très peu les ressources du serveur.

J'ai donc effectué la migration sans vraiment de difficulté. J'ai pour cela utilisé pgloader, après avoir renseigné les informations de connexion des 2 BDD, il a fait le travaille de conversion pour moi. J'ai ensuite modifié certain type qui n'était pas bon au niveau des tables.
Il ne restait plus qu'à modifier le site pour qu'il utilise la nouvelle BDD, pour une fois, j'étais content d'utiliser Laravel, car la couche d'abstraction permet le changement de BDD facilement. On a juste fait une erreur en modifiant le fichier d'environnement du projet et on a bloqué plusieurs heures sur une erreur d'accès à la BDD. La raison est toute simple et terriblement bête de notre part, on avait

renseigné “postgres” au lieu de “pgsql”. Du coup, Laravel n’utilisait pas le bon driver et la connexion ne pouvait pas s’effectuer.

Ce changement fut finalement une grosse erreur, car je n’ai jamais eu autant d’erreur de permissions d’accès à la BDD qu’avec PostgresSQL. Cela était dû à un manque de connaissance de la BDD et de la configuration des permissions. Malgré toute notre bonne volonté, il nous a été impossible de complétement régler ce problème de permissions. Mais cela ne m’empêcha pas de continuer le développement du site, les fichiers migration pour modifier la BDD nécessitaient plus d’attention qu’avant.

6.3) Nom de domaine et déploiement du site

Le déploiement du site était, après toutes ces péripéties, plus qu’une simple formalité. Il ne restait plus que la question du nom de domaine cohabit.fr. Ce nom de domaine est sous la gestion de l’IUT de Bordeaux, il a donc été nécessaire de faire appel à la DSI pour modifier le DNS et faire pointer la page et le nom de domaine sur le serveur du Fablab.

Historiquement, le nom Cohabit concernait les deux entités Fablab et Technoshop. Sauf que le Technoshop ne l’a jamais vraiment utilisé et au fil des années, dans les esprits, Cohabit faisait référence uniquement au Fablab. Il a donc fallu d’abord obtenir l’autorisation de s’approprier le nom de domaine afin de le faire pointer sur le nouveau site internet. Actuellement, les démarches ont été effectuées pour changer l’adressage et les derniers ajustements sont effectués dans le but de rendre disponible le site courant septembre 2023. Le site est accessible à l’adresse suivante : https://www.iut.u-bordeaux.fr/cohabit/

VI) Conclusion

Ce stage m'a permis de prendre en confiance en moi et m'a permis d'exprimer ma créativité à travers la conception de l'UX (user experience) et de l'UI (user interface) mais aussi par les problèmes auxquels j'ai été confronté tout au long du stage. J'ai pu mettre en pratique une grande variété de compétences que j'ai acquises durant mon cursus universitaire, mais aussi en grande partie par mes projets personnels. L'ambiance générale et la convivialité qui y règne est vraiment un plus.

Le stage m'a appris à avoir une meilleure hygiène de travail, au vu de l'énergie et de la concentration que cela m'a demandé d'avoir. J'ai une grande capacité d'analyse et de concentration, ce qui me permet d'être organisé et efficace, mais au vu de l'ampleur de la tâche, il était clair que je n'aurais pas pu en venir à bout avec le temps qui m'était impartis. Néanmoins, je suis content et fier du travail abattu, ainsi que de ma progression tant sur le plan personnel que professionnel.

En effet, je pense que malgré les précautions que j'ai mises en place pour éviter le surmenage, je ne me suis pas assez ménager, ce qui m'a contraint à mettre fin à mon stage plus tôt que prévu, à cause de problème de santé. Sans rentrer dans les détails, je traversais en parallèle une phase difficile sur le plan personnel et j'avais cumulé un emploi étudiant de 3 semaines, qui m'a nécessité également beaucoup d'énergie.

À la fin du stage, j'ai pu rendre un site internet proche du cahier des charges établi avec une documentation pratique pour déployer, mettre à jour et permettre la continuité du développement. J'ai pu approfondir mes connaissances en développement web avec PHP et Laravel et mettre au profit du Fablab mes connaissances en administration système, mais aussi sur des sujets et des domaines bien différents. J'en garde un souvenir positif et si c'était à refaire, je ne pense que je ne changerais rien.

VII) Lexique

(Définition générée à partir de ChatGPT et vérifié et corrigé par mes soins)

1. **Binding** : La liaison ou la connexion entre deux éléments logiciels ou entre un élément logiciel et un composant matériel pour synchroniser les données.
2. **Bootstrap** : Un framework CSS et JavaScript open source largement utilisé pour la création de sites web réactifs et esthétiquement plaisants.
3. **Commit** : Dans le contexte de la gestion de versions (comme Git), c'est l'action de valider et d'enregistrer les modifications apportées à un dépôt de code source.
4. **Éditeur Wysiwig** : Un éditeur qui permet aux utilisateurs de créer et de modifier du contenu web en utilisant une interface similaire à un traitement de texte, affichant le rendu final.
5. **Footer** : La section en bas d'une page web qui contient généralement des informations complémentaires, des liens ou des coordonnées.
6. **Forjero** : Un fork de Gitea résultant d'un désaccord sur le développement du logiciel. Forjero conserve néanmoins certaines caractéristiques de Gitea.
7. **Framework** : Un cadre de travail qui fournit une structure préétablie pour le développement d'applications, simplifiant ainsi le processus en utilisant des conventions et des bibliothèques.
8. **Gitea** : Une plateforme légère de gestion de développement de logiciels basée sur Git, offrant des fonctionnalités similaires à GitHub et GitLab. Vous pouvez accéder à celui du Fablab ici.
9. **<head>** : Une balise HTML utilisée pour inclure des métadonnées et des éléments de lien, généralement située dans la section d'en-tête d'une page web.
10. **Hot fixes** : Des correctifs de logiciel rapides conçus pour résoudre un problème spécifique ou une vulnérabilité critique dans une application.

11. **<iframe>** : Une balise HTML utilisée pour incorporer du contenu externe (comme une page web) dans une page web.
12. **JQuery** : Une bibliothèque JavaScript populaire qui simplifie la manipulation du DOM, les animations et les interactions utilisateur sur les pages web.
13. **Laravel blade** : Le moteur de modèle par défaut de Laravel, un framework PHP, qui simplifie la création de vues HTML en utilisant une syntaxe conviviale.
14. **Laravel Livewire** : Une bibliothèque qui simplifie la création d'interfaces utilisateur interactives en temps réel en utilisant PHP et Laravel.
15. **MVC (Modèle-Vue-Contrôleur)** : Un modèle d'architecture logicielle qui sépare une application en trois composants distincts pour gérer la logique de données (modèle), l'interface utilisateur (vue) et le contrôle des interactions (contrôleur).
16. **ORM (Object-Relational Mapping)** : Une technique permettant d'interagir avec une base de données en utilisant des objets et des relations plutôt que des requêtes SQL directes.
17. **Portfolio** : Une collection organisée de travaux, projets ou réalisations d'une personne, généralement utilisée pour démontrer ses compétences ou son expertise dans un domaine donné.
18. **Redmine** : Une application de gestion de projet open source qui offre des fonctionnalités pour la planification, le suivi et la gestion de projets, facilitant la collaboration au sein des équipes. Vous pouvez accéder à celui du Fablab ici.
19. **UI (User Interface)** : L'interface utilisateur, qui représente la partie visible et interactive d'une application ou d'un système.
20. **UX (User Experience)** : L'expérience utilisateur, qui englobe l'ensemble des émotions et des perceptions d'un utilisateur lorsqu'il interagit avec un produit ou un système.
21. **Wordpress** : Une plateforme de gestion de contenu (CMS) open source largement utilisée pour la création de sites web et de blogs.

VII) Annexes

Annexe 1 : Site du fablab (Wordpress)

COHaBIT, LE FABLAB DE L'UNIVERSITÉ DE BORDEAUX ET LE TECHNOSHOP

/// Le Fablab Coh@bit est un espace collaboratif de fabrication numérique///

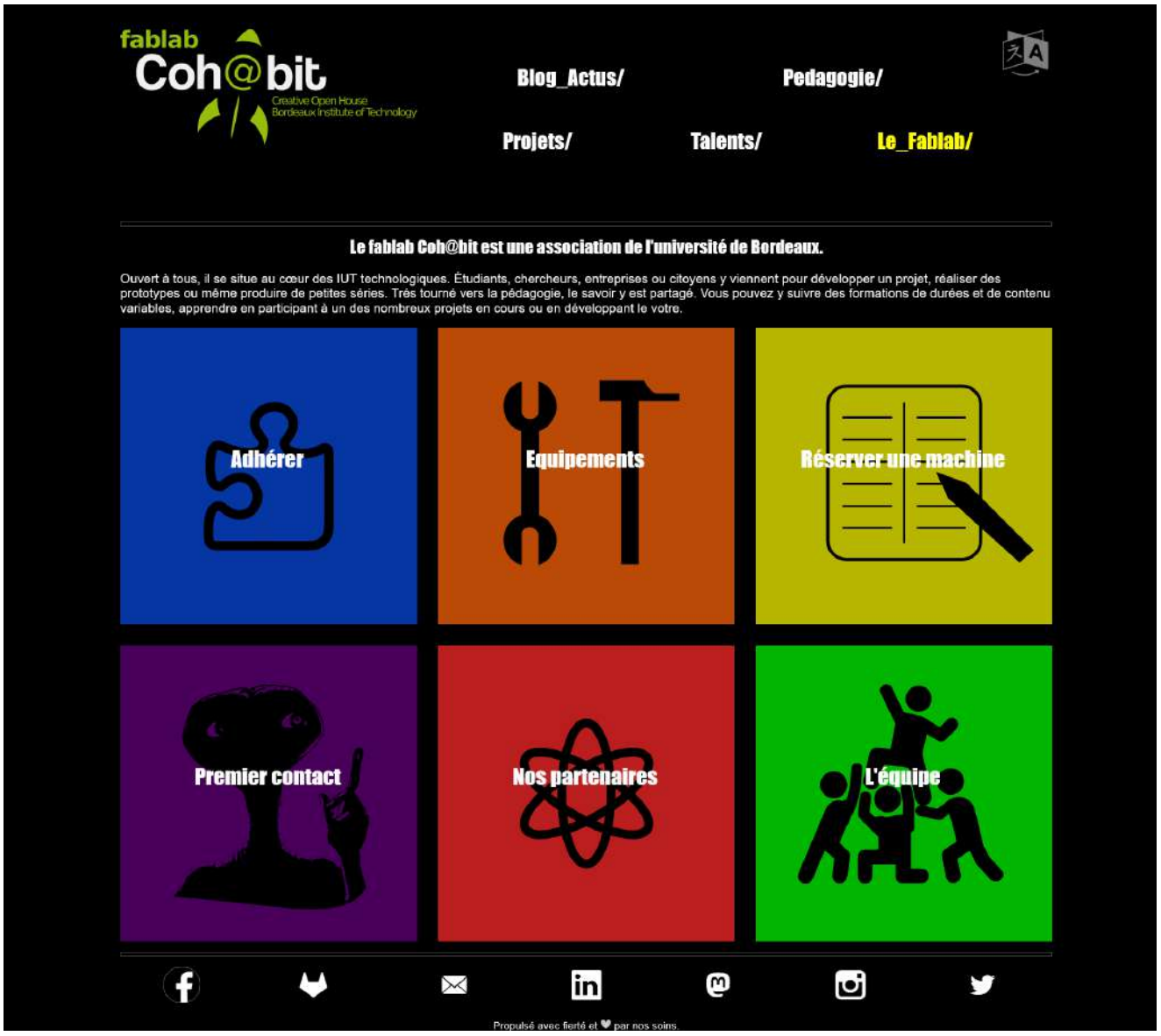
///Visite virtuelle : cliquez **ICI**///

/// Le **TechnoShop** pour le développement de projets technologiques innovants///

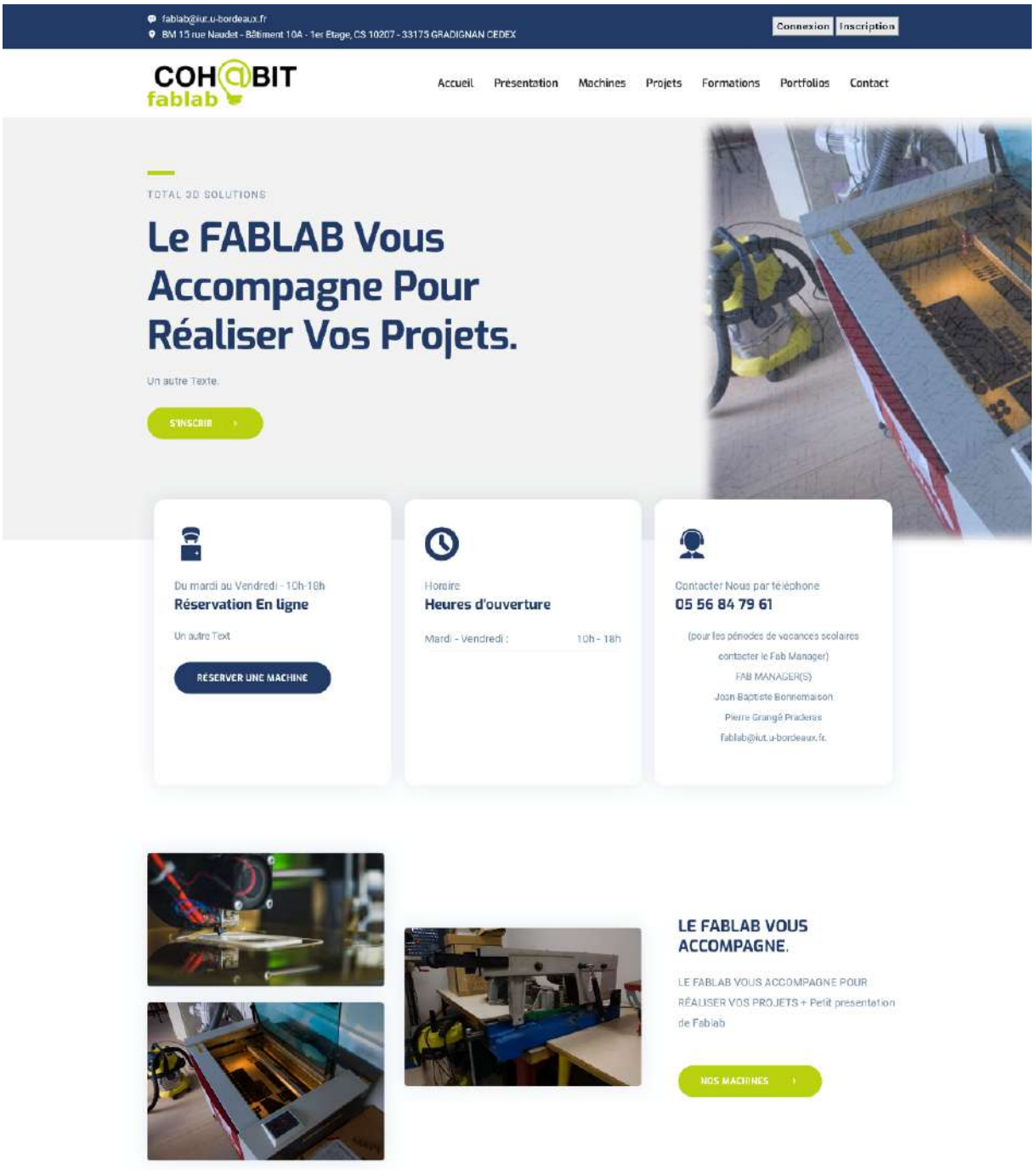
/// Devenez un Maker

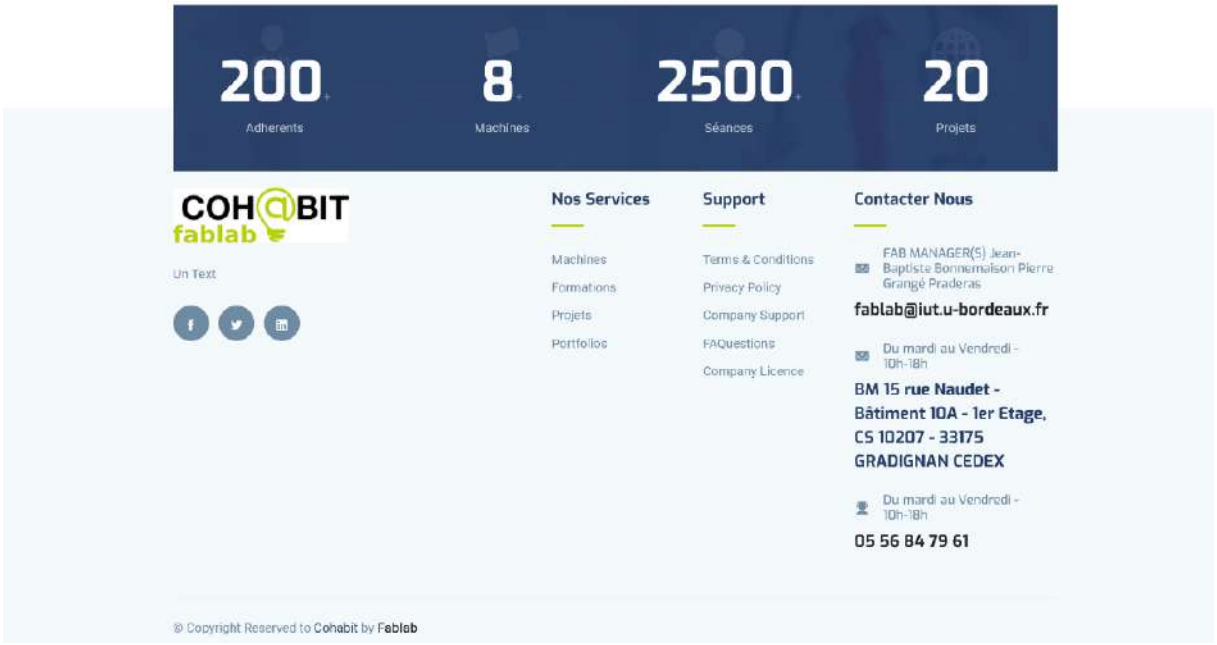
#Ils nous font confiance

Annexe 2 : Prototype de site internet



Annexe 3 : Site actuel Du Fablab (avant)





Annexe 4 : Site actuel du Fablab (après)

Les dernières annonces

Le 29 août c'est la rentrée 2023-2024

rentrée 2023-2024

Écrit par
Dernière mise à jour : 9 jour(s)

Réservation en ligne

Vous êtes déjà membre du Fablab et vous souhaitez réserver une machine ?

Réserver Une Machine

Heures d'ouvertures

Lundi :	Fermé
Mardi :	10h - 18h
Mercredi :	10h - 18h
Jeudi :	10h - 18h
Vendredi :	10h - 18h

*En périodes de vacances scolaires (**voir le calendrier**), veuillez nous contacter par mail*

Informations

Ouvert à tous !

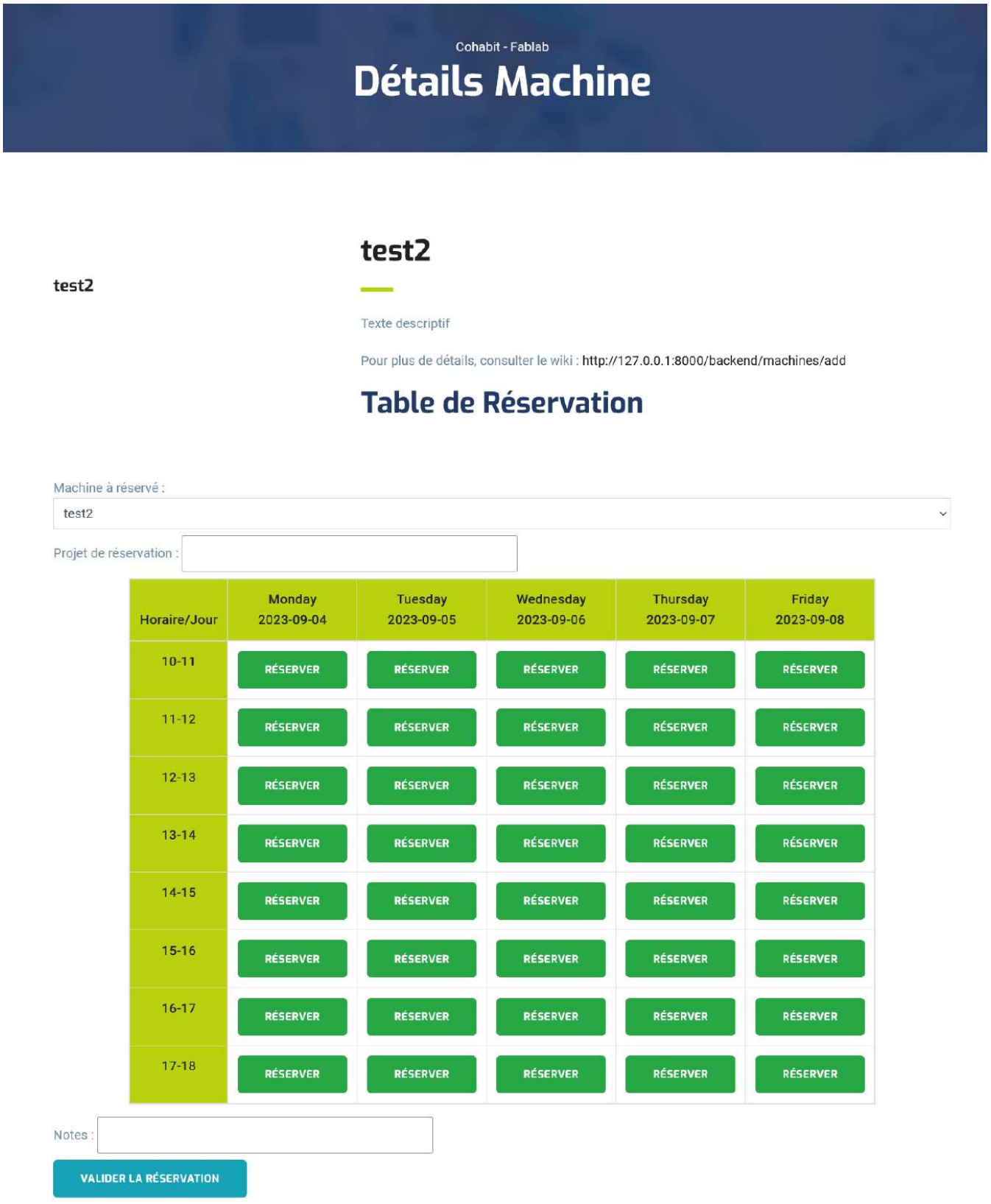
IUT de Bordeaux, Bâtiment 10A
15 rue Naudet, 33170 GRADIGNAN

05 56 84 79 61

fablab@iut.u-bordeaux.fr

Capture d'écran de la page d'accueil du nouveau site internet (02/09/2023)

Annexe 5 : Page de réservation (avant)



Annexe 6 : Page de réservation (après)

COHABIT fablab

Machines Projets Portfolios Contact FAQ test test

✓ Votre réservation a été bien enregistré. Vous pouvez la consulter à tout moment dans votre profile.

Réservation

Machine réservable :

Imprimante 3D Curiosity

Imprimante 3D Curiosity

Imprimante 3D à dépot de fils chaud Creality Ender-3
Dimension de travail : 20x20x30 cm
Matériaux acceptés : PLA, PETG
⚠ Attention : La carte d'origine à été remplacer et le micro-logiciel à été remplacer par marlin

Pour plus de détails, consulter le wiki :
https://projets.cohabit.fr/redmine/projects/documentation-machines/wiki/Ender_3

Formulaire de réservation

Nom du projet :

Calendrier de réservation (Année 2023) :

	Lundi 04 Septembre	Mardi 05 Septembre	Mercredi 06 Septembre	Jeudi 07 Septembre
10h - 11h	✓	test - Impression pièces		
11h - 12h		test - Impression pièces		
12h - 13h				
13h - 14h				
14h - 15h	test - Test d'impression			
15h - 16h	test - Test d'impression			
16h - 17h				
17h - 18h				

Valider la réservation