Dossier Projet

Développeur Web et Web Mobile
session 2020-2021

Rapport de stage : Création d'un site vitrine en React et Typescript
&
Projet personnel : Création d'une boutique en ligne en PHP orienté-objet

par

Florent BOUTONNET

Afpa

Lab-STICC

Coordonnées

Organisme de formation :

AFPA Bretagne – Centre de Brest
15 rue du Petit Spernot, 29200 Brest
09 72 72 39 36

Directeur des formations :

Christophe HOUTIN
christophe.houtin@afpa.fr

Stagiaire :

Florent BOUTONNET
florent.boutonnet@gmail.com

Entreprise :

Lab-STICC
ENSTA Bretagne
2 rue François Verny, 29806 Brest
SIREN : 192 901 254
www.ensta-bretagne.fr

Équipe :

Team OSE (Observations Signal & Environnement)

Encadrant de stage :

Dorian CAZAU
Enseignant-chercheur
dorian.cazau@ensta-bretagne.org

Tuteur technique :

Erwan KERIBIN
Ingénieur logiciel
erwan.keribin@gmail.com

Résumé

Dans le cadre de mon stage de clôture de la formation « Développeur Web et Web Mobile » dispensée à l'AFPA de Brest, j'ai travaillé au sein d'une équipe du Laboratoire Lab-STICC basée sur le site de l'ENSTA Bretagne. Cette équipe est spécialisée en océanographie acoustique, c'est-à-dire dans l'analyse des sons sous-marins. Certains membres sont plus précisément spécialisés dans l'étude des cétacés.

J'y ai développé un site de présentation des projets de l'équipe et d'exploration de leur données devant s'adresser à la fois à leurs collègues chercheurs et au grand public. Pour participer à l'organisation des données, j'ai également travaillé sur une ontologie des espèces de cétacés et de leurs différents types de sons.

D'autre part, et pour satisfaire les exigences de contenu du présent rapport, j'ai aussi développé un projet *back-end* en parallèle. Il s'agit d'une application de e-commerce programmée en orienté-objet et adoptant le patron MVC. Cette application propose un contenu dynamique généré à partir de l'accès à une base de données.

Abstract

During my internship, I worked with a Lab-STICC team located in the ENSTA Bretagne school. That team specializes in acoustic oceanography, i.e. underwater sounds analysis. Some of them are cetacean specialists.

I developed a website to introduce the team's projects and explore their datas. That website targets fellow workers and general public. I also worked on an ontology about cetacean species and their sounds.

Furthermore, to fit exam's needs, I also worked on a back-end development project. It's an e-commerce website using object-oriented programming with MVC pattern. That application displays dynamic content produced from a database.

Compétences couvertes par le projet

- Identifier les besoins clients
- Collaborer avec une équipe technique
- Proposer puis choisir une solution technique
- Réaliser un cahier des charges
- Créer une charte graphique
- Réaliser une maquette
- Réaliser des schémas techniques
- Créer des composants d'affichage *front-end*
- Réaliser une interface utilisateur
- Intégrer un affichage *responsive*
- Utiliser une API externe à l'application
- Créer une base de données
- Représenter une base de données
- Créer des composants d'accès à une base de données
- Créer une application au contenu dynamique
- Créer une application de e-commerce
- Intégrer un système de création de compte et de connexion
- Mettre en place le modèle MVC
- Créer une application en Orienté-objet
- Intégrer des recommandations de sécurité
- Sécuriser les données entrant sur le serveur
- Sécuriser l'accès aux bases de données
- Travailler en autonomie
- Prendre en main une nouvelle technologie
- Utiliser une documentation technique
- Résoudre un problème technique via une recherche sur internet
- Lire et comprendre des documentations en langue anglaise
- Réaliser des tests unitaires
- Optimiser la vitesse d'exécution d'une application
- Créditer les auteurs
- Effectuer une veille technologique et de sécurité
- Rédiger un document de description technique
- Rédiger un rapport

Table des matières

I. Projet en entreprise : création d'un site vitrine en React et TypeScript

A. Introduction

Ce stage en entreprise a été effectué dans le cadre de la formation « Développeur Web et Web Mobile » dispensée à l'AFPA de Brest du 29 Septembre 2020 au 2 juin 2021 sous la direction de l'enseignant M. Patrick LOPES-REGO et l'autorité du directeur des formations de l'AFPA de Brest M. Christophe HOUTIN. Le stagiaire, Florent BOUTONNET, a débuté son apprentissage des technologies web principalement au début de cette formation.

Ce stage d'une durée de 6 semaine a été effectué du 1er avril au 14 mai au sein de l'école d'ingénieurs ENSTA sise à Brest ; et plus précisément dans la section « acoustique sous-marine » du laboratoire de recherche Lab-STIC du pôle recherche de l'ENSTA Bretagne, sous la direction du docteur Dorian CAZAU, enseignant-chercheur au sein de l'ENSTA Bretagne, et de l'ingénieur logiciel Erwan KERIBIN, prestataire de service pour l'équipe de recherche, qui fut l'encadrant technique du stagiaire.

Cette équipe travaille sur l'observation de la biodiversité marine autour du globe par l'analyse de données acoustiques captées par des hydrophones (l'homologue sous-marin du microphone) placés à différents endroits du globe. Elle se compose principalement d'acousticiens, de scientifiques de la donnée, de biologistes marins et d'ingénieurs logiciels. Une des thématiques fortes de cette équipe est l'étude des cétacés.

L'objectif de ce stage a été de réaliser un site vitrine de présentation des activités, membres et résultats de cette équipe de recherche.

Ce site a été réalisé en majeure partie en télétravail en raison d'un confinement dû au contexte de crise sanitaire décidé début avril par le Président de la République Emmanuel Macron. La communication entre membres s'effectuait déjà de longue date via la messagerie Slack que j'ai rapidement adopté. J'ai également participé à plusieurs rendez-vous en visioconférence notamment pour communiquer avec l'équipe technique. Mais j'ai aussi pu passer plusieurs jours en présentiel à l'ENSTA Bretagne au plus proche des membre de l'équipe.

Ce rapport de stage présente une description du travail effectué en vue de l'examen du titre professionnel « Développeur Web et Web Mobile » mais aussi une documentation technique à l'attention des développeurs reprenant le projet après moi.

B. Cahier des charges

1. Présentation de l'entreprise

J'ai effectué mon stage au sein de l'école ENSTA Bretagne, sise à Brest. L'ENSTA Bretagne est une école d'ingénieurs dépendant du ministère de la défense et dispensant des formations allant du post-bac à la thèse. Ses domaines d'expertise incluent l'océanographie, les systèmes d'observation, la robotique, l'architecture véhicule et les systèmes embarqués.

Les locaux de l'école hébergent les étudiants mais également un pôle de recherche qui accueille plusieurs unités de recherche du laboratoire Lab-STICC au sein duquel évolue l'équipe qui m'a accueilli. Cette équipe nommée OsmOSE regroupe une demi-douzaine de membres et est spécialisée dans le traitement et l'analyse de données sonores sous-marines captées de manière passive.

Elle rassemble, au sein du laboratoire ou via des prestataires extérieurs très investis, des profils aux compétences diverses, notamment : traitement du signal, analyse de données, *machine learning*, étude des mammifères marins, développement logiciel et calcul distribué. L'équipe est très liée à l'institut Ifremer qui leur donne accès à leur supercalculateur situé à Brest. Celui-ci leur permet de traiter rapidement des données massives. L'équipe a de nombreux échanges avec plusieurs groupes de recherches internationaux et accueille régulièrement de jeunes stagiaires venant du monde scolaire.

L'équipe travaille principalement dans la recherche publique et transmet les résultats de ses travaux dans des publications scientifiques. Au vu de son expertise en détection de présence de faune marine, elle répond également à des demandes d'études d'impacts commandés par des institutions ou des grands groupes pour leurs projets dans le domaine maritime.

Le principal projet logiciel de cette équipe de recherche est le développement d'une plate-forme d'annotation en ligne d'évènements acoustique de la faune marine appelée Aplose. Cette plate-forme, fonctionnelle et utilisée par les membres de l'équipe, a aussi été pensée pour être utilisée par des utilisateurs non-experts dans un cadre de développement par la science participative. Si cette plate-forme est fonctionnelle, le site de présentation de l'équipe et de ses travaux était pour le moins sommaire.

2. Environnement technique en place

Ce projet de site vitrine vient s'ajouter à un historique de développement logiciel déjà conséquent. Pour les besoins de l'équipe, ont déjà été développé, entre autres, un outil de ré-échantillonnage des donnés audio, un outil de génération de spectrogrammes, une interface d'annotation de spectrogrammes et divers scripts de mise en forme des données. Ce travail logiciel est centralisé sur le répertoire GitHub public Project-ODE (https://github.com/Project-ODE) et est partagé sous différentes licences ouvertes.

Mon arrivé dans l'équipe a, peu ou prou, coïncidé avec un projet de migration du site vitrine et de l'application Aplose depuis un serveur souscris chez ovh vers l'infrastructure numérique Datarmor de l'Ifremer. Celle-ci était déjà la principale plate-forme de traitement des données et cette migration devait permettre d'y intégrer les composantes *front-end* du projet.

La principale application de l'équipe disponible via navigateur, nommée Aplose, constitue principalement un outil d'annotation d'évènements acoustiques (sifflements de dauphin par exemple). Cette application a été développé coté client avec le framework Javascript React et fonctionne coté serveur sous le framework HyperSwitch.

HyperSwitch est un framework développé par la Wikimedia Foundation (qui héberge notamment Wikipedia) dans le cadre de leur besoin logiciel. C'est un framework spécialisé dans la scalabilité des applications qu'il héberge. Mais il s'agit aussi d'un framework peu documenté et très peu populaire parmi les développeur d'applications en ligne. Pour cette raison, l'équipe technique travaillait au moment de mon stage au remplacement du framework HyperSwitch par le framework python Django, beaucoup plus connu parmi les développeurs.

La plupart des données textuelles descriptives des jeux de données audio sont dans un premier temps transmises dans le format .CSV puis les informations nécessaires à la bonne organisation et au bon archivage des données sont stockées dans une base de donnée gérée en PostGresQL.

Pour faciliter l'utilisation des différents éléments logiciels, l'équipe technique utilise des conteneurs Docker permettant de définir l'environnement logiciel adéquat.

3. Définition des objectifs

L'équipe OsmOSE est en demande de diverses fonctionnalités logicielles mais promeut l'autonomie de ses membres. C'est pourquoi je n'ai pas reçu de cahier des charges à mon arrivée ni non plus de direction très arrêtée sur le contenu du travail à fournir.

Pour définir le travail à effectuer lors de ce stage, j'ai donc pris du temps en amont du stage et lors des premiers jours pour discuter avec les différents membres de l'équipe et consulter les documents de fonctionnalités demandées les plus récents afin d'effectuer un recensement des principaux besoins. J'ai par la suite partagé ce document récapitulatif à l'ensemble des membres de l'équipe. Lors de discussions, j'ai également pu récolter nombre d'informations techniques sur l'architecture des applications développées au sein du projet. Sur la base de celles-ci, nous avons convenu avec mon encadrant de stage ainsi que les référents techniques un ensemble de tâches de développement adaptées à mes compétences et à la durée du stage.

En raison de la faible documentation du *framework back-end*, on m'a clairement indiqué qu'il ne serait pas judicieux pour moi de passer du temps à l'apprendre pour pouvoir travailler dessus. C'est pourquoi je ne ferais, dans le cadre de mon travail au sein de l'équipe OSE, que peu de développement coté serveur.

Nous avons donc décidé que l'objectif pour ce stage serait de créer un site vitrine présentant le projet OsmOSE et son équipe dans un premier temps. Puis ensuite de travailler sur l'ontologie des types de son étudiés et plus particulièrement sur les sons des cétacés. Cette ontologie devra servir de base pour développer une table dans la base de données ainsi qu'une page du site dédiée à chaque tag et présentant des informations qui seront par la suite tirées de la base de données.

Il s’agira d’un site vitrine à destination des universitaires, des institutions, des grands groupes ou du grand public qui aurait envie d’en savoir plus sur les travaux de l’équipe.

4. Contraintes et solution retenue

Pour réaliser ce site vitrine, l’équipe technique m’a indiqué rapidement les technologies à utiliser. Elle souhaitait que je développe à l’aide du Framework React. Ce framework constitue une option de choix dans le cadre de ce projet :

- il s’agit d’un framework largement connu des développeurs *frontend,* qui sera donc facilement modifiable par les membres de l’équipe actuelle ou future.
- issu de Facebook, ce framework bénéficie d’un suivi de qualité et probablement de long terme.
- il est aisément compatible avec nombre de framework *backend*, et notamment HyperSwitch et Django.
- il est déjà utilisé par Aplose, l’application principale de l’équipe. Cette convergence réduit le nombre de technologies à maîtriser pour intervenir sur le projet. Cela pourrait aussi permettre un rapprochement éventuel futur entre cette application et le site vitrine.

L’équipe m’a également demandé de travailler avec le logiciel Git qui améliore la gestion du code développé collaborativement.

Le stage devait aboutir sur une application fonctionnelle au terme de sa durée de 6 semaines.

5. Présentation des technologies utilisées

NodeJS :

NodeJS est un environnement d’exécution JavaScript proposant, via son gestionnaire de paquets NPM l’accès à une riche bibliothèque de fonctionnalités pouvant s’exécuter coté serveur.

React :

React est un framework JavaScript populaire permettant de créer des applications web qui peuvent modifier les données des pages, sans avoir à les recharger. React a l’avantage d’être évolutif et de proposer des applications rapides d’exécution. Il peut être utilisé facilement avec d'autres bibliothèques ou frameworks JavaScript.

TypeScript :

TypeScript est un sur-ensemble de JavaScript. Il s’agit d’une version fortement typée de JavaScript améliorant largement les possibilités du langage de créer de grosses applications. TypeScript ajoute d’importantes possibilités en programmation orienté-objet dans JavaScript.

JSX :

JSX est une extension à JavaScript permettant d'écrire et de manipuler facilement le DOM en utilisant une syntaxe proche du XML, bien connue des développeurs. Ayant fortement gagnée en popularité ces dernières années, cette extension constitue la manière claire et moderne d'écrire les composants React.

Bootstrap :

Bootstrap est une librairie CSS permettant de rapidement créer un site uniforme entre les navigateurs et *responsive* à l'aide de *classes* aux noms explicites. Elle inclut nombre de composants préprogrammés à l'apparence moderne et claire pour l'utilisateur. Bootstrap est un standard dans la création de la structure CSS des sites web.

CSV :

Le CSV est un format de données structurées en tableau. Il est facilement modifiable depuis la plupart des logiciels pouvant éditer un fichier tableur (comme Excel) et aisément transformable en une table de base de données.

Leaflet :

Leaflet est une bibliothèque JavaScript permettant d'inclure des cartes interactives dans des pages web. Les cartes sont proposées par différents fournisseurs spécialisés. Une version reconstruite de Leaflet sous forme de composants React et appelée « React Leaflet » a été utilisée pour le projet.

6. Prestations attendues

- Création d'une maquette fonctionnelle
- Définition d'une charte graphique
- Recherche et intégration de contenus graphiques
- Création d'une page d'accueil avec 2 panneaux
- Création d'une page de description du projet
- Création d'une page de présentation de l'équipe
- Proposer une ontologie des tag utilisés sur la plate-forme d'annotation
- Présenter cette ontologie dans un format structuré facilement transposable dans une base de données
- Intégration d'une carte permettant la localisation des jeux de données
- Intégration d'un tableau permettant le listage et l'organisation temporelle des jeux de données

- Création d’un arbre HTML affichant automatiquement les tag enfants depuis un fichier d’ontologie
- Création d’une page exploration contenant une carte et une liste de tags sous forme d’arbre
- Création d’une page pour chacun des tags avec informations dédiées et fil d’Ariane pour la navigation dans l’arborescence
- Proposer une application *responsive*
- Proposer une application résistante aux attaques malveillantes
- Fournir une application aisée à maintenir et à développer pour les intervenants suivants
- Transmettre le code à l’équipe technique sur GitHub

C. Maquette

Navigation sur le site :

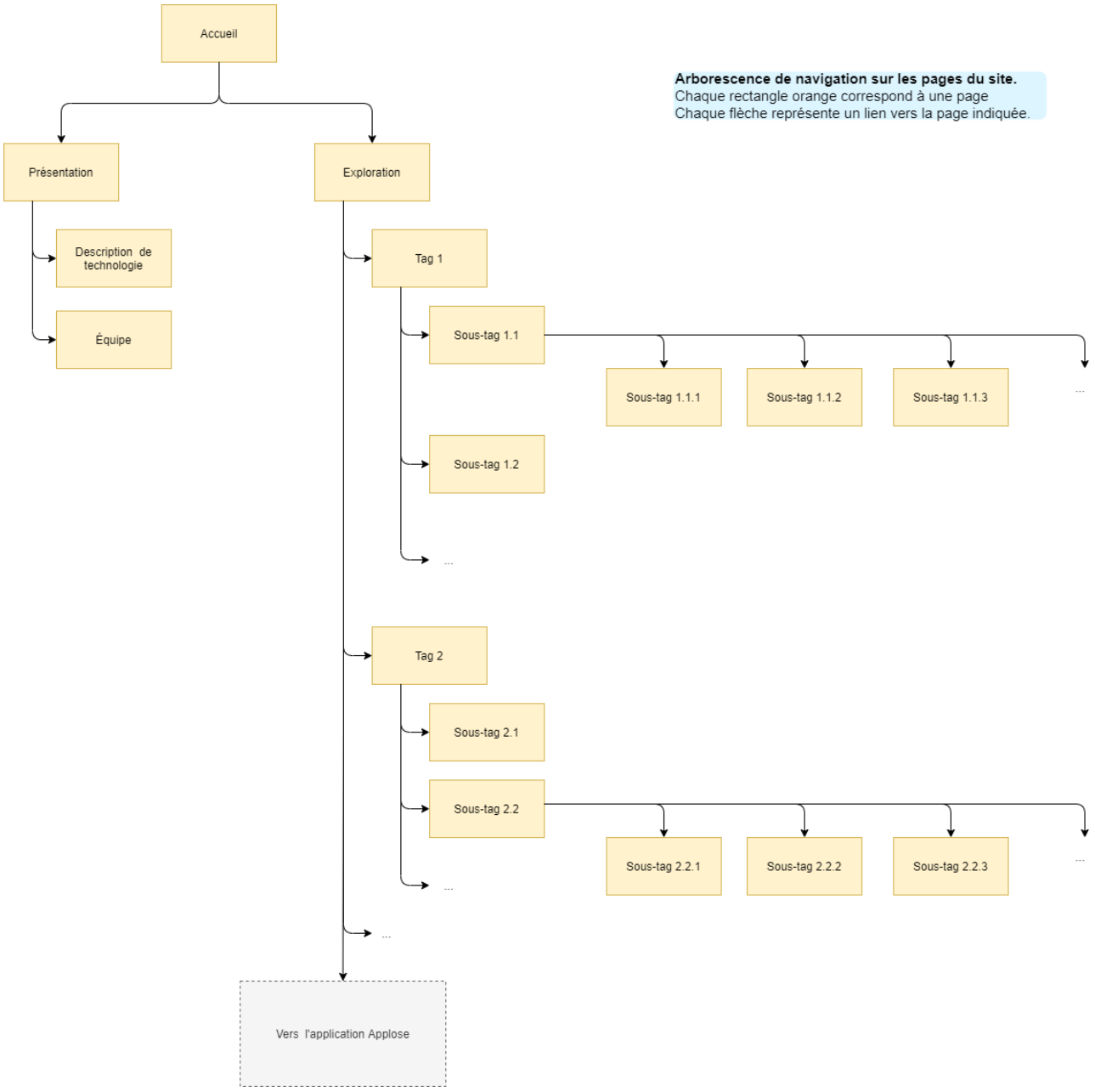


Fig 2 : Plan des pages du site

Le site proposera un petit nombre de pages au contenu fixe dans la partie de présentation rédigée d'une part (sur la gauche de la figure 2) puis un nombre variable (selon les besoins de l'équipe) de pages de *tags* générées et hiérarchisées automatiquement d'après des informations prélevées depuis un fichier de données dédié à l'ontologie des espèces et leurs sons (dans un premier temps) ou depuis une table de la base de données (dans un second temps).

Page d’accueil :

Fig 3 : Maquette de la page d’accueil

Le client a clairement indiqué sa volonté de bien distinguer deux parties sur le site : une partie centrée sur la présentation rédigée et une autre sur l’exploration de données possédées. Cette distinction devait être claire dès la page d’accueil. Une présentation sous forme de deux images-liens carrées côte à côte pour la version de bureau du site a été demandé. Pour la version mobile, ces deux images-liens devraient être mises l’une au dessus de l’autre dans un format plus rectangulaire pour les distinguer sans avoir besoin de faire dérouler la page.

Page-type de présentation rédigée :

Fig 4 : Maquette d’une page de contenu rédactionnel

Il s’agit ici d’une page de présentation type qui sera utilisée pour la page de présentation du projet mais pourra aussi l’être dans le futur pour diverses pages de présentation. Des pages sur la technologie ou des valeurs du projet ont été évoquées. Une page de présentation des membre de l’équipe devra utiliser une présentation proche en incluant des variations adaptées.

Page de présentation d’un tag :

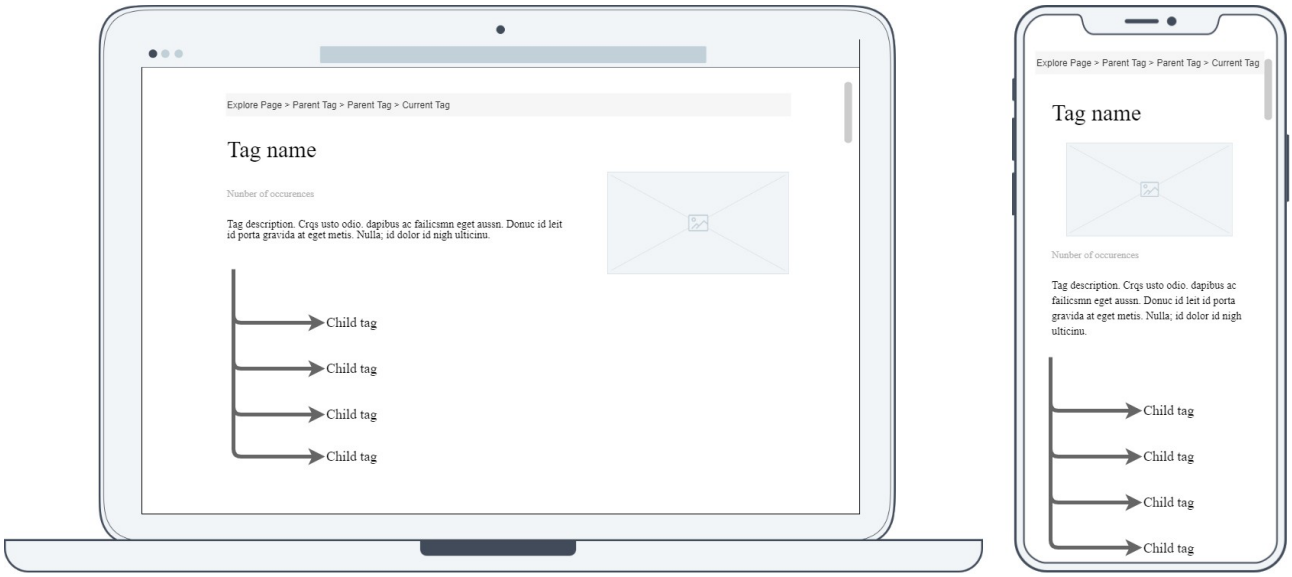


Fig 5 : Maquette d’une page de présentation d’un tag

Le site devra proposer des pages de présentation des *tags*. Chaque page est générée automatiquement d’après le même *template* rempli automatiquement à partir d’un fichier de données définissant l’ontologie choisie. Cette page contient notamment :

- un fil d’Ariane (*breadcrumb*) pour visualiser et se déplacer rapidement dans la hiérarchie des pages de *tag*
- un arbre présentant les *tags* enfants directs
- diverses informations liées au *tag*, comme une description écrite, une illustration ou le nombre des observations du *tag* effectuées par l’équipe via la plate-forme Aplose.
- une carte de visualisation des emplacement de ces observations

D. Développement

1. Formation sur la technologie demandée

N'ayant jusque là pas eu l'occasion de travailler avec les technologies React et TypeScript, je me suis rapidement plongé dans leur apprentissage. Le site officiel du framework **React** (https://fr.reactjs.org/docs/getting-started.html) propose une riche documentation en langue française ainsi qu'un tutoriel qui abouti à la création d'un jeu de morpion. Ayant saisi l'utilisation des possibilités de base après quelques jours passés sur la documentation officielle, je décide de rapidement poursuivre sur les technologies suivantes.

Le sur-ensemble **TypeScript** bénéficie lui aussi d'une riche documentation proposé par l'éditeur sur le site officiel (https://www.typescriptlang.org/docs/). Cette documentation n'est pour l'instant qu'en langue anglaise mais cela ne m'a pas posé soucis : le niveau de langue est très abordable et le vocabulaire technique m'était déjà familier. J'ai notamment débuté par la page « TypeScript for JavaScript Programmers » (https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html) qui présente les principales nouveautés du langage.

J'ai retrouvé dans ce langage plusieurs concepts et syntaxes que j'avais déjà pu manipuler d'autres langages, comme le typage des variables, les interfaces et plus généralement la programmation orientée-objet basée sur les classes.

Ayant à cœur d'utiliser les technologies populaires dans l'environnement React, je me suis rapidement orienté vers l'adoption de la technologie **JSX** (https://facebook.github.io/jsx/) qui a été récemment introduite dans celui-ci et qui propose une nouvelle manière d'écrire les composants React. Il s'agissait ici d'apprendre directement à écrire du React moderne.

Dans le cadre d'un bon support du framework React par TypeScript, j'ai pu trouver les documentations officielles « TypeScript pour React » (https://react-typescript-cheatsheet.netlify.app/docs/basic/setup) ou la section « JSX » du *handbook* TypeScript (https://www.typescriptlang.org/docs/handbook/jsx.html).

Avec les contraintes de temps liées au projet, je me suis principalement concentré sur les connaissances permettant d'être rapidement opérationnel afin de réaliser le projet. Pour pouvoir atteindre un bon niveau de maîtrise des technologies pré-citées, il me faudra bien plus approfondir les documentations ainsi que la pratique.

2. Mise en place

Transmission de l'existant :

L'équipe OsmOSE disposait jusqu'ici d'un site vitrine réalisé avec un *template* de HTML5 Up. Ce site avait bénéficié de peu d'attentions tant en terme de développement que de rédaction. Pour bien maîtriser la transformation du site et faciliter mes débuts avec les nouvelles technologies, l'équipe technique m'a fournit le contenu de ce site sous la forme d'un projet React déjà écrit en TypeScript et fonctionnant avec un serveur de développement nodeJS.

Configurations et dépendances :

L'environnement nodeJS en place ne contenait qu'un nombre limités de dépendances. Le projet a été initié avec le populaire pack d'initialisation « *Create React App* ». Le *bundler* Webpack était déjà inclus et bien configuré. Je n'éprouverai pas le besoin d'y revenir lors de ma phase de développement. TypeScript a été configuré pour accepter le JSX ainsi que renvoyer du code ECMAScript 5. Il a aussi été configuré en « mode strict ».

3. Lancement et premier blocage

J'ai rencontré mon premier blocage lors du lancement du projet sur ma machine. Après avoir *forké* le projet depuis Github puis installé les dépendances nodeJS, l'application refusait de se lancer via la commande 'npm start' malgré la documentation fournie.

Fig 6 : Message d'erreur nodeJS au lancement du projet

J'ai donc recherché la source du problème sur internet en utilisant le nom de l'erreur associé au nom de la commande entrée. En parcourant les résultats fournis, je compris que cette erreur 'cb() never called !' semble apparaître dans différentes circonstances, ce qui ne facilita pas la recherche. Parmi les résultats, les fils de discussion de forum de développeurs (comme StackOverflow) ou de post de blogs de développeurs me sont apparus comme les plus pertinents.

Grâce à leur lecture attentive, il m'est apparu que ma version de nodeJS ne permettait pas de lancer le projet. Il ressortait de manière évidente que le projet n'utilisait pas exactement la version de nodeJS que je venais d'installer. Après vérification dans les variables d'environnement de mon ordinateur il m'apparaît que celui-ci possède une installation en *global* de nodeJS. Je décide de désinstaller celle-ci et modifier les variables d'environnement associées. Mais le problème persiste. Je retourne sur les forums pour apprendre que ce genre de problème se règle par la suppression du cache de l'installation nodeJS qui peut persister malgré les désinstallations puis redémarrer l'ordinateur.

Ce premier contretemps aura été pour moi l'occasion de mes premières recherches de solutions sur internet. Le forum spécialisé sur le développement StackOverflow m'aura permis de rapidement trouver les raisons et les solutions à mes problèmes. C'est un site sur lequel je suis régulièrement revenu et qui propose des réponses triées d'une communauté active de développeurs.

4. Charte graphique

L’équipe m’a demandé de proposer une identité visuelle au projet en m’accordant une grande liberté créative. Bien que je ne sois pas graphiste, j’ai mis en place certaines idées pour diriger la création visuelle.

Le site vise à présenter un projet récent d’une équipe jeune et dynamique alliant expertise scientifique, travail collaboratif basé sur le volontariat et développement logiciel. Un visuel moderne, clair et attrayant mais sérieux m’est apparu comme une direction adéquat.

En demandant des références visuelles, j’ai pu avoir accès à un travail de charte graphique réalisé il y a environ 2 ans par un graphiste mais qui n’a que très peu été utilisé jusqu’alors. Ce travail comportait un logo, ainsi que des couleurs dominantes. J’ai repris en grande partie ce travail et y ai ajouté ma part en m’inspirant également de site de projets scientifiques similaires que m’ont indiqué les membres de l’équipe.

Fig 8 (ci-dessus à gauche) : Image d’inspiration pour l’univers graphique.

Fig 9 (ci-dessus à droite) : Logo.

Fig 10 : Couleurs majeures de la charte graphique.

Prenant en compte ces travaux et réflexions, je décide de réaliser le site sur fond blanc, pour la sobriété et l’aspect moderne, avec un univers graphique marqué par une dominante bleue qui rappelle la mer. Les liens utilisent un bleu azur très lumineux et facilement repérable par les visiteurs que j’ai repris des éléments graphiques récupérés.

Pour illustrer le site j'ai fait principalement mes recherches sur la base d'images en utilisation libre Pexels (www.pexels.com) et ai crédités les auteurs dans un fichier humans.txt situé dans le dossier '/public'. J'ai effectué une sélection d'images de belle qualité des mammifères marins les plus étudiés par l'équipe, notamment baleines et dauphins.

Pour des questions visuelles et d'optimisation, j'ai effectué plusieurs recadrages, redimensionnements et compressions de ces images avec le logiciel XnConvert.

Pour la typographie, j'ai opté pour la police d'écriture « Exo 2 » trouvé sur le site fonts.google.com. Il s'agit d'une police *sans serif* qui, selon l'auteur, « tente de transmettre un sentiment de technologie tout en restant élégante ». J'ai opté pour la graisse 300 pour l'usage de base.

Fig 11 : Police de caractères « Exo 2 » utilisé pour le projet.

5. Mise en page de la partie de présentation

Pour la partie de présentation rédigée du site, je me suis tourné vers une organisation en paragraphes illustrés chacun par une image. Cette association paragraphe-image, souvent appelé *card,* a été réalisé à partir d'un squelette en Bootstrap modifié selon mes besoins. J'ai réalisé des variations de *cards* pour les sections de description, pour les membres principaux et pour les membres plus anciens. Ces *cards* proposent un affichage *responsive* automatique en colonne pour les écrans de petite largeur et un autre en ligne pour les écrans plus grands.

On m'a demandé une présentation en bandeau permettant de mettre en avant certaines informations dans la page et briser la monotonie des suites de *cards*. Le composant créé, que j'ai nommé *Banner* permet de mettre l'accent sur de courts textes ou images disposées en colonnes.

En m'inspirant d'un site dédié au même sujet, j'ai réalisé un composant permettant de présenter le titre de la page au sein d'une image fine s'étalant sur toute la largeur de l'écran.

Enfin, j'ai développé un composant *Header*, contenant une barre de navigation, un autre *Footer*, proposant des liens d'information, ainsi qu'un *Layout* utilisant les deux composants précédents pour encadrer le contenu des composants dits « *page* ».

L'ensemble du site a été pensé pour être adaptable à tous types d'écrans et d'orientations. L'usage extensif de la librairie CSS Bootstrap a grandement facilité ce travail.

6. Ontologie

Les membres de l'équipe ont vu apparaître, avec l'expérience de l'utilisation de l'application Aplose, des problématiques de nommage, doublons et organisation des labels annotés. On m'a demandé, suite à la création de la partie graphique du contenu de présentation, de me pencher sur la question de l'ontologie des *tags* utilisés.

Après plusieurs discussions avec les membres de l'équipe pour bien évaluer tous les aspects du sujet, j'ai commencé par créer une première version de l'ontologie des tags sous la forme d'un fichier au format JSON.

Fig 12 : Exemple d'entrée du fichier d'ontologie en JSON.

Ce format permet de hiérarchiser facilement les données entrées. Mais je me suis rapidement rendu compte que ce format ne pouvait avoir qu'une utilité temporaire pour plusieurs raisons :

- certains tags n'ont pas une hiérarchie simple et peuvent appartenir à plusieurs parents,
- l'organisation que je propose pourrait être modifiée à l'avenir, ce qui pourrait s'avérer fastidieux dans un format comme le JSON
- cette ontologie a vocation à être transférée par la suite dans une base de données, ce qui n'est pas simple avec ce format.

Je décide donc de réécrire l'ontologie dans le format CSV qui me paraît bien plus adapté au vu des contraintes soulevées.

7. Développement de la partie exploration

L'équipe était en demande d'une page permettant au visiteur de visualiser facilement les informations concernant les jeux de données qu'utilisent les annotateurs d'une part et d'exposer l'ontologie présentée sous forme hiérarchisée d'autre part.

De plus, l'objectif était de pouvoir accéder à toutes les entrées de l'ontologie pour lesquels l'équipe possédait des données d'observation intéressantes. C'est pourquoi je me suis attelé à la construction d'une page de *layout* dont le contenu serait peuplé automatiquement par les informations se trouvant dans le fichier (et par la suite la table de la base de donnée) définissant l'ontologie.

Pour répondre à ces problématiques j'ai donc développé 3 composants principaux qui vont chacun récupérer des informations depuis le fichier d'ontologie :

Liste en arbre :

Inspiré d’une présentation similaire trouvée sur un site de recherche, cet arbre propose une liste automatique des *tags* enfants au *tag* appelé. Cette liste peut présenter une description du tag, une image d’illustration, certains tags enfants selon des critères définis et le nombre d’observations déjà faites du *tag*. Ce composant a été pensé pour être modulaire et être utilisé sur la page générale (dans laquelle il présente peu d’informations) et sur les pages dédiées à chaque *tag* (dans lesquelles il affiche beaucoup plus d’informations).

Carte de monde avec marqueurs de position :

Afin de fournir un aperçu direct de la distribution spatiale des jeux de données étudiées par l’équipe, une carte du monde paraissait toute indiquée. La librairie Leaflet et le projet React Leaflet ont permis de réaliser cette carte de manière aisée. La recherche de cartes de qualité m’a permis de trouver des cartes bathymétriques qui ont plu aux membres de l’équipe.

***Layout* de page de présentation des tags :**

Les pages de présentation des tags seront générées automatiquement à partir de ce composant de type *page.* Il contient, en plus de la liste en arbre avec illustration et de la carte du monde, un fil d’Ariane pour naviguer facilement dans la hiérarchie des tags parents.

E. Exemple de fonctionnalité : la liste en arbre

La liste en arbre, mentionnée à la section précédente a requis des développements tant graphiques que fonctionnels :

Développement fonctionnel :

Pour ce composant, j'avais besoin de générer automatiquement des éléments de liste à partir du fichier de définition de l'ontologie (utilisé sous le nom de variable 'jsonObject' dans l'extrait ci-dessous).

```
// Add taglist of children with at least 1 occurrence in DOM
function addChildrenTagList() {
  if (tag !== undefined && tagChildren !== null){
    let ulTree = document.getElementById(tag);
    if (ulTree != null && ulTree.children[0] === undefined){
      // créer un élément li puis l'ajouter dans le document
      for (let i = 0; i < tagChildren.length ; i++) {
        let tagChildrenOccurence = jsonObject[tagChildren[i]].occurrence;
        if (tagChildrenOccurence > 0){
          let liElem = document.createElement('li');
          let linkElem = document.createElement('a');
          linkElem.href = "/ontology?" + tagChildren[i];
          linkElem.innerText = jsonObject[tagChildren[i]].engName + " ";
          let abbrElem = document.createElement('abbr');
          abbrElem.innerHTML = "occ.";
          abbrElem.title = "occurences";
          let spanElem = document.createElement('span');
          spanElem.classList.add('badge', 'badge-pill', 'text-secondary');
          spanElem.innerHTML = tagChildrenOccurence + " ";
          spanElem.appendChild(abbrElem);
          liElem.appendChild(linkElem);
          liElem.appendChild(spanElem);
          ulTree.appendChild(liElem);
```

Fig 13a : Extrait de code d'ajout automatique des éléments <li> dans le DOM.

Partie graphique :

Le composant affiche des éléments de liste (<li>) liés entre eux par une barre verticale symbolisant la hiérarchie et une barre horizontale symbolisant l'entrée.

```css
div.treelist > ul > li::after{
    content: '';
    width: 40px;
    height: 0;
    position: absolute;
    left: -38px;
    top: 1.2rem;
    border-top: solid 3px #002d80;
}
```

Fig 13b : Extrait de création graphique de la liste en arbre en CSS

Utilisation :

L'usage, sous une syntaxe JSX, dans les composants *pages* se voulait simple.

```jsx
<TreeList
tag="phocoeninae"
titleLevel="h3"
/>
```

Fig 13c : Code d'usage du composant dans les pages.

Rendu :

(données d'exemples)

Fig 14 : Rendu du composant « TreeList ».

F. Mode d'emploi

1. Lancement

L'application se lance de manière classique pour un projet en nodeJS, à l'aide des commandes suivantes :

```
# Install the modules from the package.json file
npm install
# Run development server
npm start
```

Fig 15 : Commandes de lancement du projet.

2. Fonctionnement

Organisation du code :

Un projet en React correspond à la couche de visualisation sur un modèle de type MVC.

Le dossier « *public* » regroupe les fichiers accessibles aux visiteurs. Le cœur de l'application React se trouve dans le dossier « *src* ». En React, les éléments de présentation sont appelés « *components* » et ont été rangés dans le dossier du même nom. Il s'agit de blocs de code permettant d'afficher du contenu sous une forme prédéfinie.

Ces *components* sont appelés dans les fichiers « *pages* » regroupés dans le dossier éponyme. Chaque fichier *component* ou *page* importe un fichier CSS associé situé dans le même dossier pour plus de clarté.

Les fichiers TypeScript racines 'App.tsx' et 'index.tsx' se trouvent à la racine du dossier 'src'.

Je me suis attaché, lors du développement à garder une organisation du code conforme aux bonnes pratiques de React. J'ai aussi veillé à commenter correctement mon code pour les développeurs qui reprendront mon application.

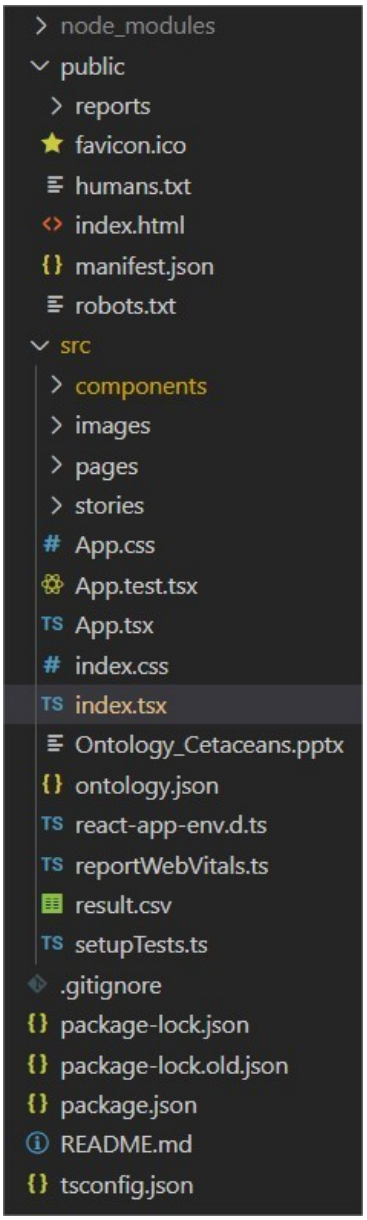


Fig 16 (ci-contre à droite) : Arborescence des dossiers du projet

Hiérarchie de l'application :

Lorsque le visiteur arrive sur une page du site, il appelle la page 'index.html' située dans le dossier '/public'. Celle-ci, en plus de la

structure HTML de base des pages et des appels aux fichiers externes par CDN (Bootstrap par exemple), contient l'élément suivant :

```
<div id="root"></div>
```

Fig 17 : Emplacement cible de l'application React dans la page 'index.html'

Cet élément est ensuite ciblé depuis le fichier 'index.tsx' du dossier 'src' qui lui-même appelle le fichier 'App' :

```
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Fig 18 : Extrait de code du fichier racine 'index.tsx'

Le fichier appelé, 'App.tsx', contient lui le routage de l'application et le contenu des fichiers « *pages* ». Les fichiers « *pages* » contiennent eux-mêmes les fichiers « *components* », à la base des contenus affichés.

Interventions sur le contenu des pages :

Les membres de l'équipe volontaires pour rédiger le contenu du site n'ayant pas forcément de formation en langages de programmation web, j'ai tenu à réaliser des composants React simples d'utilisation pour les interventions concernant la rédaction et l'agencement des pages.

```
<CardMember
name="Dorian Cazau"
img={defautPortrait}
imgSide="left"
imgAlt="Dorian's portrait."
job="Data Scientist"
// url="https://www.google.com"
>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Quasi minus officia molestias, voluptate
    obcaecati ipsam consectetur error dolores iusto eius quaerat? Lorem ipsum dolor sit amet consectet
    officia molestias, voluptate
  </p>
</CardMember>
```

Fig 19 : Inclusion du composant « CardMember *» dans une page.*

G. Rendu visuel

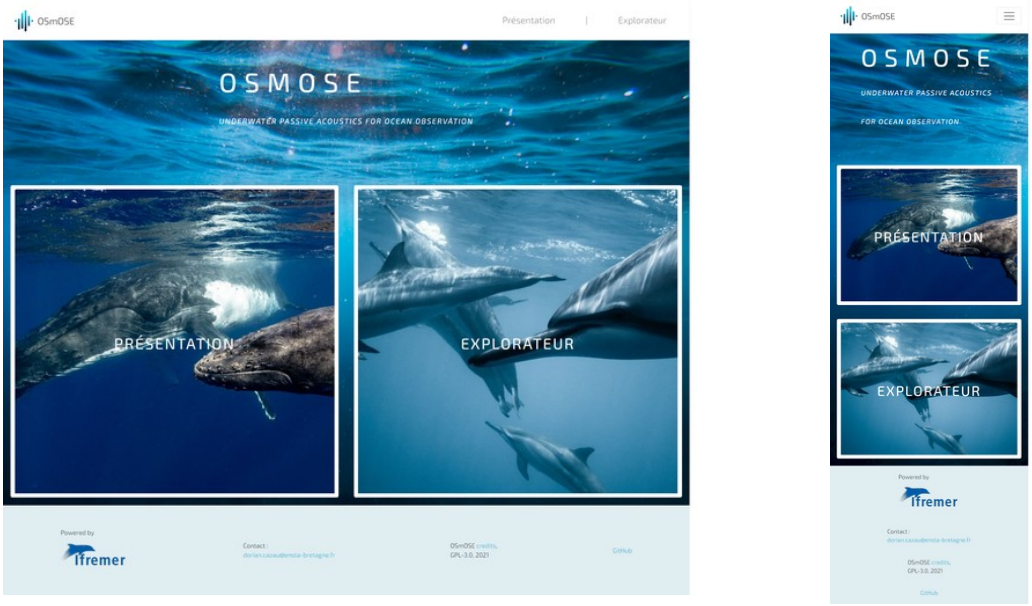


Fig 20a et 20b : Rendu de la page d'accueil sur un écran de bureau (à gauche) et sur un smartphone *(à droite).*

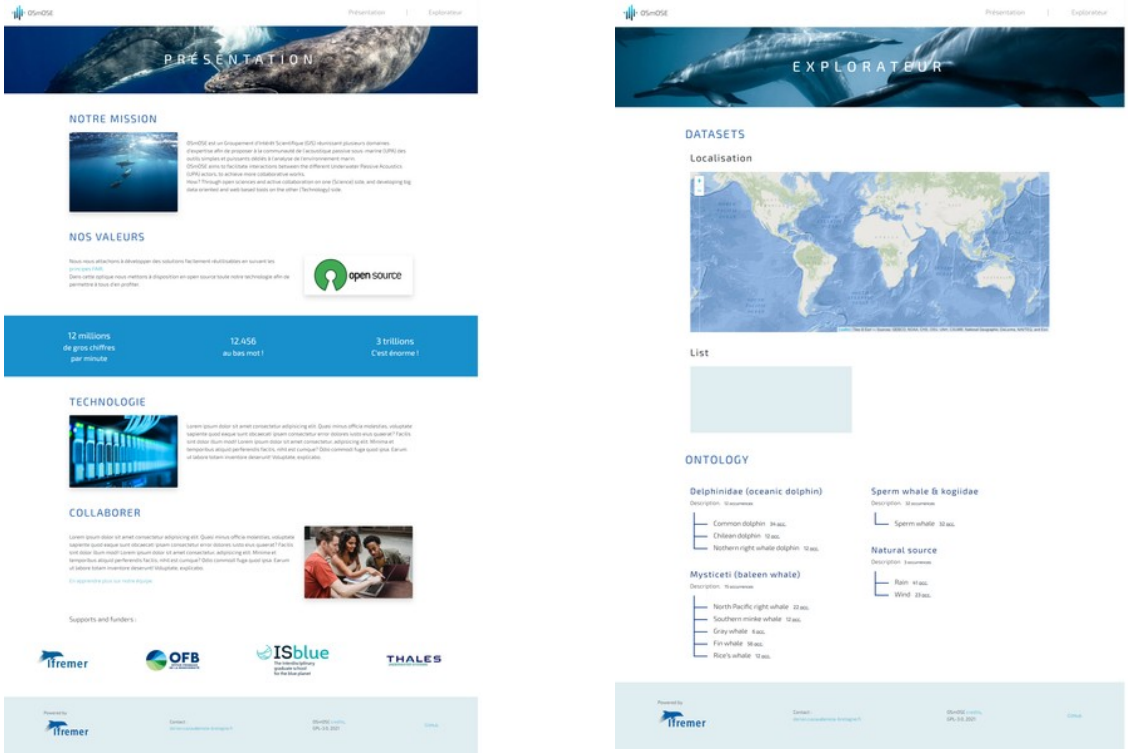


Fig 21a : Rendu de la page de présentation (à gauche).

Fig 21b : Rendu de la page d'exploration (à droite).

H. Bilan

Travail rendu :

L'application rendue en fin de stage correspond bien aux objectifs que je m'étais fixé d'après les demandes clients lors de la création de la maquette : un visuel attractif mais sérieux, une partie présentation du projet séparée d'une partie exploration des données proposant des fonctionnalités de contenu dynamique et un site entièrement *responsive*.

Limites :

En raison des contraintes de temps je n'ai pas pu finaliser la transformation du fichier d'ontologie créé en table de la base de donnée. De même, à cause de difficultés liés au *framework backend* utilisé, je n'ai pas pu faire de développement dans ce domaine, comme expliqué plus haut. Je n'ai donc pas eu la possibilité de véritablement lier mon application React avec la partie *backend*.

En raison de ma formation expresse sur React et TypeScript, il est possible, malgré le fonctionnement du code, que celui-ci présente des faiblesses de débutant dans cette technologie.

Suite :

J'ai assuré à l'équipe que j'effectuerai un suivi de l'application jusqu'à ce que, en accord avec l'équipe, elle soit validée sur la partie technique et achevée sur la partie rédactionnelle. L'équipe et moi visons une mise en ligne effective du site sur le site de l'Ifremer à l'adresse https://osmose.ifremer.fr/ durant l'été.

II.Exercice personnel : Création d'un site de e-commerce en PHP orienté-objet

A. Introduction

Avec le projet de site vitrine pour OsmOSE j'ai pu couvrir les compétences liées au développement *front-end*, comme l'affichage *responsive*, la création de charte graphique, le maquettage en fil de fer, mais aussi des compétences comme l'analyse des besoins clients, le travail en équipe ou encore l'utilisation d'une API externe.

Lors de mon développement logiciel au sein de cette équipe, il m'est apparu que je ne pourrais pas réaliser un projet qui couvre l'ensemble des compétences demandées lors de l'examen pour l'obtention du titre « Développeur Web et Web Mobile ». Je me suis donc attelé à la constitution d'un autre projet qui permettrait de couvrir ces manques.

Ce projet a dû être réalisé sur le temps très réduit dont je disposais d'environ 2 semaines.

B. Cahier des charges

1. Objectif

J'ai, pour ce projet, principalement voulu répondre aux demandes de compétences inscrites dans le référentiel Emploi-Compétence dédié au titre « Développeur Web et Web Mobile » que je n'avais pas pu réalisé au sein du projet OsmOSE.

Je me suis donc orienté sur la création d'une application me donnant l'occasion de créer et manipuler une base de données, créer des composants d'accès à cette base de données afin de réaliser des pages au contenu dynamique. La sécurisation de l'accès à cette base de données doit constituer un sujet d'importance.

Une boutique en ligne m'est apparue comme une application permettant de mettre en œuvre ces compétences et dont le savoir-faire est apprécié dans le domaine professionnel.

Ayant déjà eu l'occasion de développer des fonctionnalités similaires dans divers petits projets codés sous forme impérative, j'ai voulu pour cet exercice réaliser un projet en pur orienté-objet. Cette contrainte m'a rapidement ouvert la possibilité de mettre également en place le modèle MVC qui correspond à une séparation stricte du code.

Pour ce projet, j'ai décidé, pour développer mes compétences fondamentales dans ce domaine, de ne pas utiliser un framework existant mais de créer le code de l'application à partir de zéro. Le nombre des fonctionnalités sera probablement réduit, mais j'en maîtriserai tous les aspects.

2. Technologies

Pour réaliser cette application sans l'aide framework, je décide d'utiliser le langage PHP avec lequel j'ai déjà pu expérimenter plusieurs types d'applications. La base de données (BDD) sera gérée par MySQL adossée au moteur InnoDB. L'extension PHP PDO me permettra d'accéder et d'interagir avec la base de donnée.

Voici une rapide présentation des technologies retenues :

PHP :

PHP est un langage de script exécuté sur le serveur et offrant de riches possibilités pour construire des application faisant le lien entre BDD et contenu des pages web. PHP permet de s'insérer rapidement dans un environnement de technologies libres et gratuites. PHP est la technologie phare du développement *backend*.

Programmation orientée-objet :

La programmation orientée-objet (POO) est un paradigme de programmation dans lequel l'ensemble de la logique est contenu dans des objets. Dans le cas de PHP, ces objets sont des instances de classes.

PDO :

PDO est une extension de PHP permettant d'interagir avec une base de donnée via le langage SQL.

MySQL :

MySQL est un système de gestion de base de données (SGBD) gratuit supportant les BDD relationnelles. Il s'agit du SGBD le plus utilisé au monde.

SQL :

SQL est un langage permettant de décrire des requêtes destinées aux bases de données relationnelles. C'est un standard dans le domaine des bases de données.

MVC :

Le modèle MVC (Model-View-Controller) est une manière de concevoir une application web en séparant clairement l'affichage, la logique et les données.

Twig :

Twig est un moteur de *templates* populaire permettant de manipuler facilement la structure html d'une page à partir d'information envoyés par des fichiers PHP.

Bootstrap :

Bootstrap est une librairie CSS permettant de réaliser rapidement un site *responsive.* Il met également à disposition des composants préalablement stylisés prêts à l'emploi.

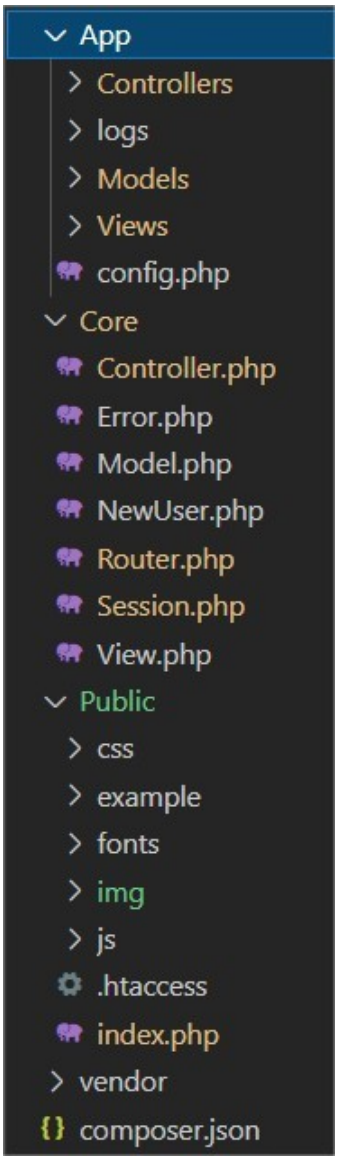
3. Fonctionnalités attendues

- réaliser un site de e-commerce
- créer une base de données
- stocker les informations produits dans une base de données
- créer des composants d'accès à une base de données
- afficher des pages web au contenu dynamiques
- sécuriser l'accès à la base de données
- créer un système d'inscription et de connexion utilisateur
- créer une page de profil

- créer une barre de navigation
- créer une page ‘Catalogue’ présentant tous les produits
- Créer une page ‘A propos’.

C. Description

1. Architecture



Cette application, développé selon le patron MVC, sépare de manière stricte des parties« *Controllers* », « *Models* » et « *Views* » : aucun code PHP ne doit apparaître dans les vues, et aucun appel à la base de données ne doit ignorer la classe ‘Core/Model’.

Une application PHP en orienté-objet utilisant le patron MVC se construit sous la forme de plusieurs arborescences de classes contenant la logique de l’application (la partie *Controller*) ou représentant les données utilisées (la partie *Model*) ainsi que des fichiers dédiés à la partie visuelle du site (la partie *View*).

Cette application en orienté-objet utilise l’héritage afin de transmettre un certain nombre de propriétés et méthodes qui peuvent être utiles à leurs descendants.

2. Hiérarchie et fonctionnement

Fig 24 (ci-contre à droite) : Arborescence des fichiers de l’application.

Le projet est découpé en quatre principaux dossiers :

- le dossier ‘Core’ regroupe les classes racines du programme. Ce sont elles qui, par héritage, vont transmettre un certains nombre de propriétés et méthodes à des classes enfants dédiées à une fonctionnalité précise situées dans la partie ‘App’.
- Le dossier ‘vendor’ regroupe les quelques bibliothèques externes utilisées pour le projet (notamment Twig)
- le dossier ‘App’ regroupe plusieurs dossiers :
  - le dossier ‘Models’ regroupe les classes correspondant à des fonctionnalités requérant l’accès à la base de données.
  - le dossier ’Views’ regroupe l’ensemble des fichiers de structure des pages affichées sur le site. Ces fichiers contiennent du code HTML ainsi que du code Twig.
  - le dossier ‘Controllers’ regroupe des fichiers de classes qui définissent le comportement des fonctionnalités dont ils sont en charge. Ils héritent de la classe ‘Controller’ située dans ‘Core/Controller.php’. Le plus souvent, ces classes appellent des données via une classe ‘Model’ et ceux-ci sont renvoyées d’après le schéma d’un fichier vue situé dans ‘Views’.

- le dossier ‘Public’ enfin, correspond à la racine de l’adresse web de l’application. Elle regroupe donc les fichiers pouvant être accessibles par les visiteurs du site. Cela correspond principalement
  - aux fichiers de style CSS,
  - aux fichiers JavaScript permettant des interactions avec l’utilisateur
  - aux fichiers images utilisés sur le site.
  - au fichier ‘index.html’ qui définit les routes et lance l’exécution de l’application

Routage :

Le routage de l’application se fait via la classe Router définie dans le fichier ‘Core/Router’. Cette classe contient la méthode ‘add()’ permettant de créer un tableau associatif entre des chemins demandés via l’URL et des contrôleurs capable de générer des pages dynamiques. Cette classe contient aussi une méthode ‘dispatch()’ qui va trouver la route demandée depuis ce tableau puis instancier le contrôleur associé

3. Base de données

Pour ce projet j’ai pu mettre en place une base de données pour stocker et accéder rapidement aux informations dont j’avais besoin. Ces données se concentrent principalement autour des produits et des utilisateurs.

Dans le cadre de la méthode Merise, j’ai réalisé des représentations en diagramme de la base de donnée, bien que celle-ci soit encore inachevée.

Modèle conceptuel de données :

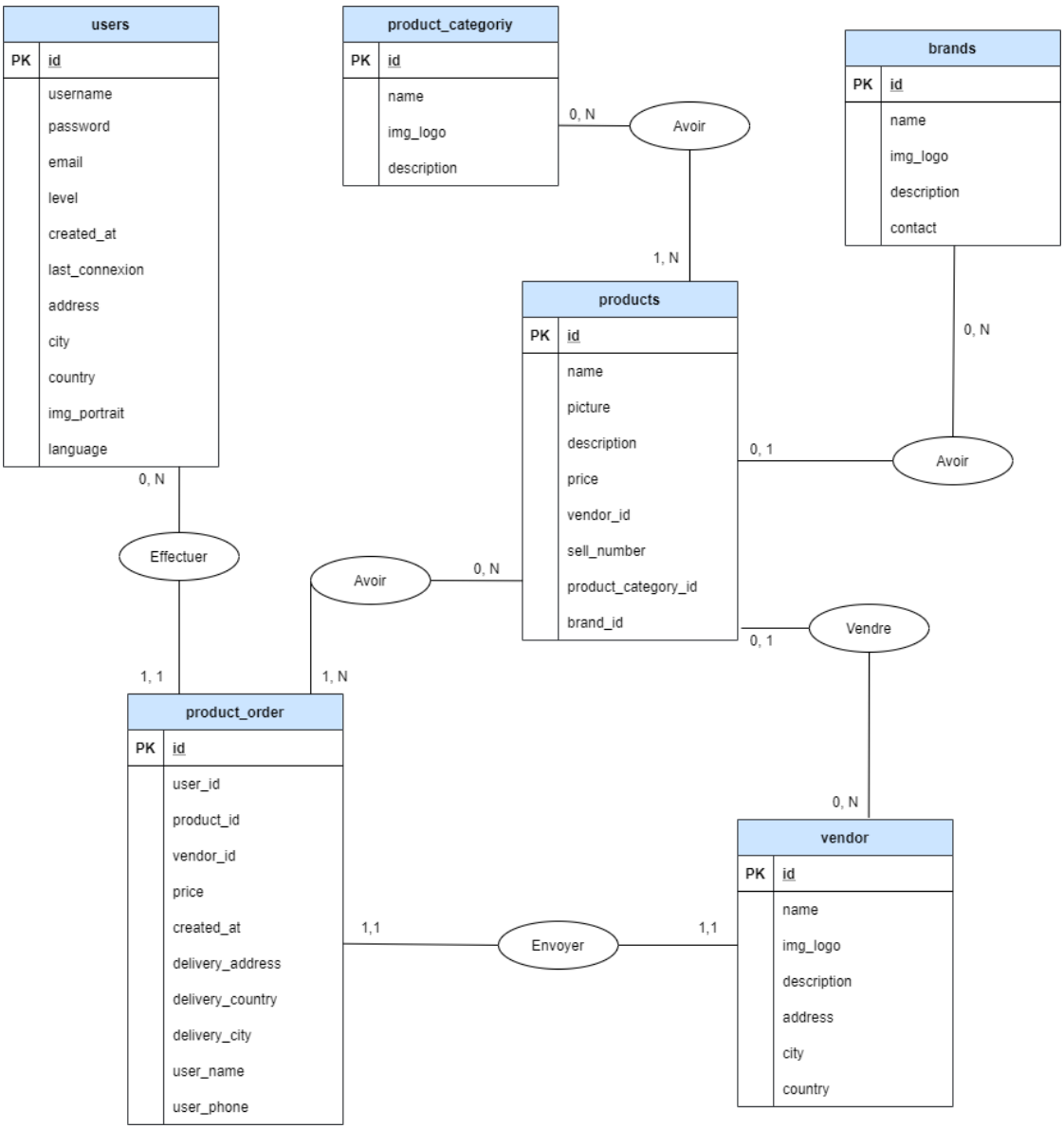


Fig 26 : Diagramme du Modèle Conceptuel des Données.

Modèle logique de données :

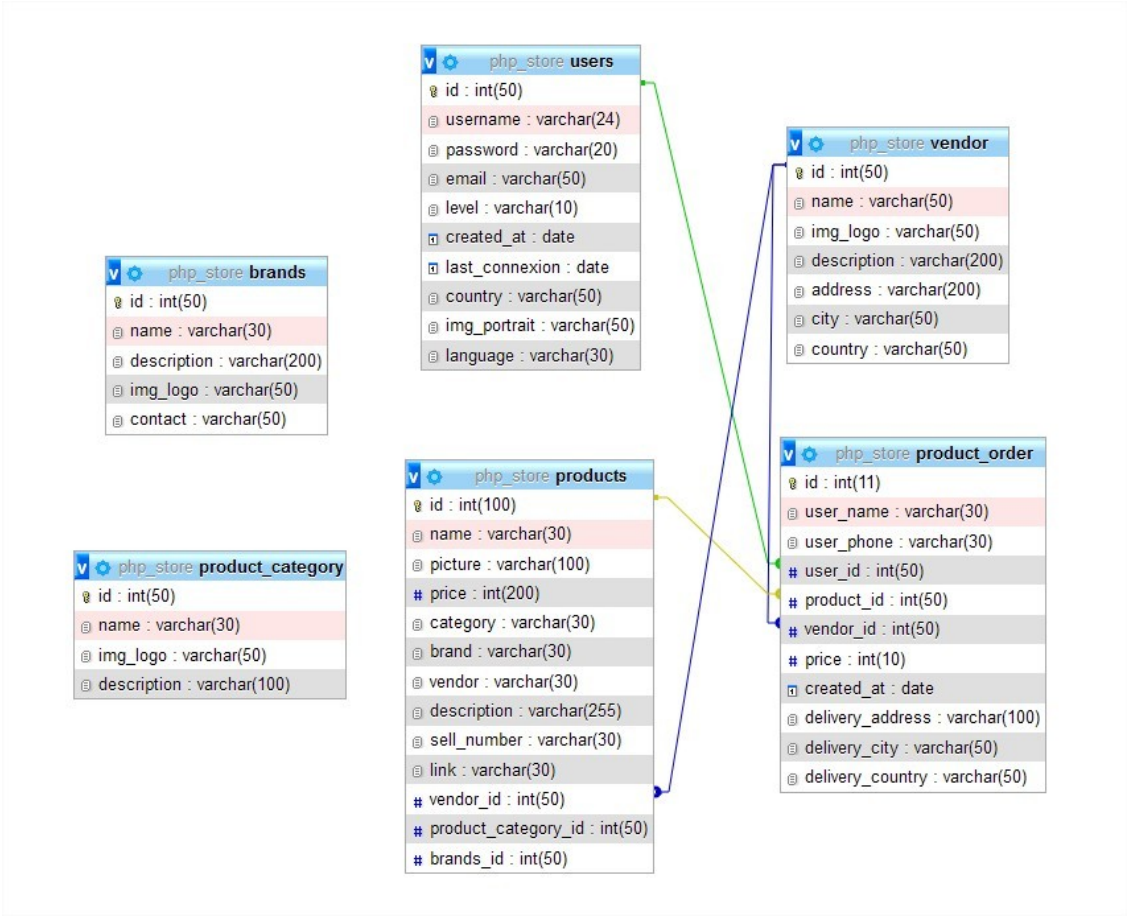


Fig 27 : Diagramme du Modèle Logique des Données

Codage :

La connexion à la base de données est définie dans une classe parente à toutes celles qui auront besoin d'accéder à celle-ci. Elle se trouve dans le fichier 'Core/Model.php'

```php
// Get the PDO database connection using Config file
protected static function getDb(){
    static $db = null;
    if ($db === null){
        $dsn = "mysql:host=" . Config::DB_HOST . ";dbname=" . Config::DB_NAME . ";
        charset=utf8";
        $db = new PDO($dsn, Config::DB_USER, Config::DB_PASSWORD);
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    return $db;
}
```

Fig 28 : Extrait du fichier 'Core/Model.php' établissant la connexion à la BDD.

Pour la page de catalogue par exemple, les résultats de nombreuses requêtes à la base de données sont envoyés dans un fichier vue.

```php
public function catalogAction(){
    View::renderTemplate('Catalog/index.php', [
        'all_products'=>Product::getAll(),
        'topsellers'=>Product::getquery('topsellers'),
        'hats'=>Product::getquery('hats'),
        // 'displayhats'=>Product::getquery('hats'),
        'scarfs'=>Product::getquery('scarfs'),
        'socks'=>Product::getquery('socks'),
        'lacoste'=>Product::getquery('lacoste'),
        'laredoute'=>Product::getquery('laredoute'),
        '3suisses'=>Product::getquery('3suisses'),
        'leprintemps'=>Product::getquery('leprintemps')
    ]);
}
```

Fig 29 : Extrait du fichier 'App/Controllers/Catalog.php' associant des clés de tableau à des requêtes déjà préparées.

Je choisis d'inclure mes requêtes au sein d'une structure switch afin de n'écrire du code SQL qu'au sein de la partie Model de l'application. De plus il me semblait que cette structure serait plus difficile à détourner par un utilisateur malveillant.

```php
public static function getquery($definedquery){

    switch($definedquery){
        case 'topsellers':
            $sqlQuery = "SELECT * FROM products ORDER BY sell_number DESC LIMIT 0, 3";
            break;
```

Fig 30 : Extrait du fichier 'App/Models/Product.php' définissant ces requêtes préparées.

Attentif aux thématiques de sécurité, je met en place des mesures de protection face aux attaques de type **injection SQL**. Pour cela j'utilise, pour faire des requêtes à la base de données, les fonctions prepare(), bindParam() puis execute() qui empêchent l'inclusion de mots-clés en place de variables et constituent les fonctions recommandées pour prévenir les injections SQL.

```php
public static function getOne($product_id){
    $db = static::getDB();       You, 2 weeks ago • recomit
    $stmt = $db->prepare("SELECT * FROM products WHERE id=:product_id");
    $stmt->bindParam(':product_id', $product_id);
    $stmt->execute();
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

Fig 31 : Extrait du fichier 'App/Models/Product.php' permettant d'effectuer une requête à la BDD de manière sécurisée.

4. Affichage

L'affichage de l'application se fait via des fichiers vues situées dans 'App/Views/'. Ces fichiers sont appelés par les contrôleurs correspondants qui leur envoient en paramètre des données provenant de la base de données. Ces données sont utilisées par le moteur de template Twig afin de générer un contenu dynamique dans lapage.

Malgré le temps très court, ce projet propose lui aussi un affichage *responsive*, bien que l'esthétique n'ait pas pu être aussi travaillée que pour le projet OSmOSE.

5. Exemple de fonctionnement : l'appel de la page d'accueil

Le fonctionnement général de l'application lors de l'appel de la page d'accueil est le suivant :

1. l'agent utilisateur demande l'URL de la page d'accueil du site ;
2. en faisant ceci (comme pour n'importe quel autre URL du site), il appelle le fichier 'index.html', situé dans 'Public' ;
3. via celui-ci, une instance de la classe 'Router' définie dans 'Core/Router.php' est créée ;
4. puis la méthode 'add()' de l'objet Router est utilisée pour créer un tableau de correspondances entre des chemins d'URL et des contrôleurs ;
5. la méthode 'dispatch()' de l'objet Router instancie la classe du contrôleur 'Home' situé dans 'App/Controllers/Home.php' qui correspond à la route demandée d'après le tableau des routes préalablement créé ;
6. cette méthode 'dispatch()' appelle ensuite la méthode nommée 'indexAction()' de l'objet contrôleur 'Home' ;
7. cette méthode 'indexAction()' appelle la fonction 'renderTemplate()' de la classe 'View' située dans 'Core/View.php' ;
8. cette fonction 'renderTemplate()' prend en argument plusieurs appels à la fonction 'getQuery()' de la classe Product située dans App/Models/Product.php ;

9. cette fonction ‘getQuery()’ fait correspondre un argument à une phrase de requête SQL puis utilise l’héritage pour faire appel à la fonction ‘getDB()’ de la classe ‘Model’ située dans le fichier ‘Core/Model.php’ ;
10. cette fonction ‘getDB()’ utilise l’extension PHP PDO ainsi que la classe ‘Config’ du fichier ‘App/config.php’ pour se connecter à la base de données et lui faire la requête SQL prévue ;
11. les retours de cette requête sont envoyés en tant qu’objets parmi les arguments de la fonction ‘renderTemplate()’ ;
12. la fonction ‘renderTemplate()’ prend aussi en argument le fichier *template* correspondant. En l’occurrence il s’agit du fichier ‘App/Views/Home/index.html’ ;
13. dans ce fichier *template,* du code twig permet d’appeler le fichier ‘App/Views/base.html’ servant de *layout* incluant le code HTML des éléments <head>, <header> et <footer> commun à tous les fichiers ‘*Views*’ ;
14. ce fichier *template* est aussi associé à du code HTML afin d’inclure des données provenant de la base de donnée ;
15. la fonction ‘renderTemplate()’ crée une instance de twig puis appelle la fonction ‘render()’ de l’objet twig en utilisant les arguments de cette fonction ‘renderTemplate()’ ;
16. le retour de cette fonction ‘render()’ est affiché dans le fichier contrôleur ‘App/Controllers/Home.php’, lequel apparaît sur le navigateur de l’internaute.

6. Lancement de l’application

Pour lancer l’application, il est nécessaire :

1. d’avoir une installation de PHP (7.4) et Composer (2.0)
2. d’installer les dépendances listées dans le fichier ‘composer.json’ à l’aide de la commande suivante dans un terminal :

```
λ composer install
Installing dependencies from lock file (including require-dev)
```

Fig 32 : Commande d’installation des dépendances du projet.

3. de placer la racine du serveur sur le dossier ‘Public’ du projet.
4. de lancer le serveur

D. Exemple de fonctionnalité : inscription utilisateur

1. Description et sécurisation

L'inscription utilisateur est une fonctionnalité extrêmement commune sur tout type de site internet, et à fortiori sur un site de e-commerce. J'ai donc décidé de développer cette fonctionnalité.

Pour ce faire, j'ai décidé de construire une page d'inscription accessible depuis la page de connexion et permettant via l'utilisation d'un formulaire, d'entrer ses données utilisateur qui seront ensuite, après nettoyage et vérifications de sécurité, stockées dans la base de données.

Bien qu'anciens, les formulaires sont toujours des entrées privilégiées pour plusieurs types d'attaques malveillantes. J'ai donc mis en place, en parallèle des mesures de prévention des injections SQL décrites plus haut, plusieurs procédures pour réduire l'exposition aux attaques de type XSS.

Je commence par créer un formulaire HTML dans la vue 'App/Views/Signin/index.php'. Ce formulaire envoie les informations en POST dans la fonction 'register()' du contrôleur 'App/Controllers/Signin.php'. Cette fonction effectue ensuite plusieurs tests pour vérifier la conformité des données. Entre autres :

- elle vérifie que tous les champs sont remplis avec des données acceptables, c'est à dire des caractères sûrs à rentrer dans cette partie *backend*. Pour se prémunir des attaques malveillantes, nombre de caractères spéciaux sont bannis, notamment les chevrons.
- elle vérifie en utilisant des expressions régulières que la valeur envoyée par le champ *email* est bien une adresse *email*.
- elle vérifie que le mot de passe envoyé comporte au moins 8 caractères.
- elle vérifie que la valeur du mot de passe est bien identique à celle de la confirmation du mot de passe.

En cas d'échec d'un de ces tests, les données ne sont pas envoyés à la base de données et un message d'erreurs adéquat est envoyé à l'utilisateur.

Si tous les tests sont validés, alors la fonction 'register()' soumet le mot de passe à la fonction PHP 'password_hash()' puis appelle la classe 'User' de la couche modèle située dans le fichier 'App/Models/User.php' afin d'envoyer ces données utilisateur dans la base de données.

```php
if($_SERVER['REQUEST_METHOD'] == 'POST'){
// Process form
// Sanitize POST data
$_POST = filter_input_array(INPUT_POST, FILTER_SANITIZE_STRING);

    $data = [
        'username' => trim($_POST['username']),
        'email' => trim($_POST['email']),
        'password' => trim($_POST['password']),
        'confirmPassword' => trim($_POST['confirmPassword'])
    ];

    $nameValidation = "/^[a-zA-Z0-9]*$/";
    $passwordValidation = "/^(.{0,7}|[^a-z]*|[^\d]*)$/i";

    //Validate username      You, seconds ago • Uncommitted changes
    if (empty($data['username'])) {
        $data['usernameError'] = 'Please enter username.';
    } elseif (!preg_match($nameValidation, $data['username'])) {
        $data['usernameError'] = 'Name can only contain letters and numbers.';
    }
```

Fig 34 : Extrait de la méthode 'register()' du contrôleur Signin.php permettant de s'inscrire sur le site.

Lors de la création d'une nouvelle entrée à la table Users, un certain nombre de champs de la table vont automatiquement se peupler avec des valeurs non envoyées. Par exemple le champs 'id' va générer une valeur numérique valant 1 de plus que le l''id' du précédent utilisateur de la table.

Les recommandations de sécurité publiées par l'ANSSI ont constitué pour moi un guide de qualité dans ma démarche de sécurisation de mes projets web. Des posts de blogs de développeurs ont aussi pu m'apporter éclairage ponctuel et exemple de sécurisation concret.

2. Jeu d'essais

Pour cette fonctionnalité sensible, je décide de réaliser un jeu de tests fonctionnels et de sécurité.

La session s'est déroulée comme suit :

Scénario (avant envoi)	Attendus	Observé
Oubli de renseignement d'un champ.	Aucune nouvelle entrée dans la base de données. Rechargement de la page d'inscription.	Fonctionnement attendu.
Valeurs des champs 'password' et 'confirm_password' différents.	Aucune nouvelle entrée dans la base de données. Rechargement de la page d'inscription.	Fonctionnement attendu.
Mot de passe de 5 caractères seulement	Aucune nouvelle entrée dans la base de données. Rechargement de la page d'inscription.	Fonctionnement attendu.
Adresse e-mail sans extension de domaine	Aucune nouvelle entrée dans la base de données. Rechargement de la page d'inscription.	Fonctionnement attendu.
Balise <script> dans le champ 'username'.	Aucune nouvelle entrée dans la base de données. Rechargement de la page d'inscription.	Fonctionnement attendu.
Champs renseignés correctement	L'inscription est validé. Une nouvelle entrée est créée dans la table 'user' avec les informations correspondantes et une id unique. L'internaute est redirigé sur la page Profil.	Fonctionnement attendu.

Les résultats des tests sont positifs, les composants se comportent comme prévus. Néanmoins, le choix de simplement recharger la page d'inscription sans fournir d'indications claire concernant l'échec et la raison de l'échec de l'inscription va vite s'avérer limitant pour le confort de l'utilisateur. Ce comportement, temporaire, a été dicté par des contraintes de temps de développement et est évidemment amené à être amélioré. Par exemple en envoyant des valeurs de retour vers le client en AJAX puis en implémentant des animations et couleurs très lisible. Bootstrap propose des animations de formulaire pour ce type de scénario.

E. Rendu visuel

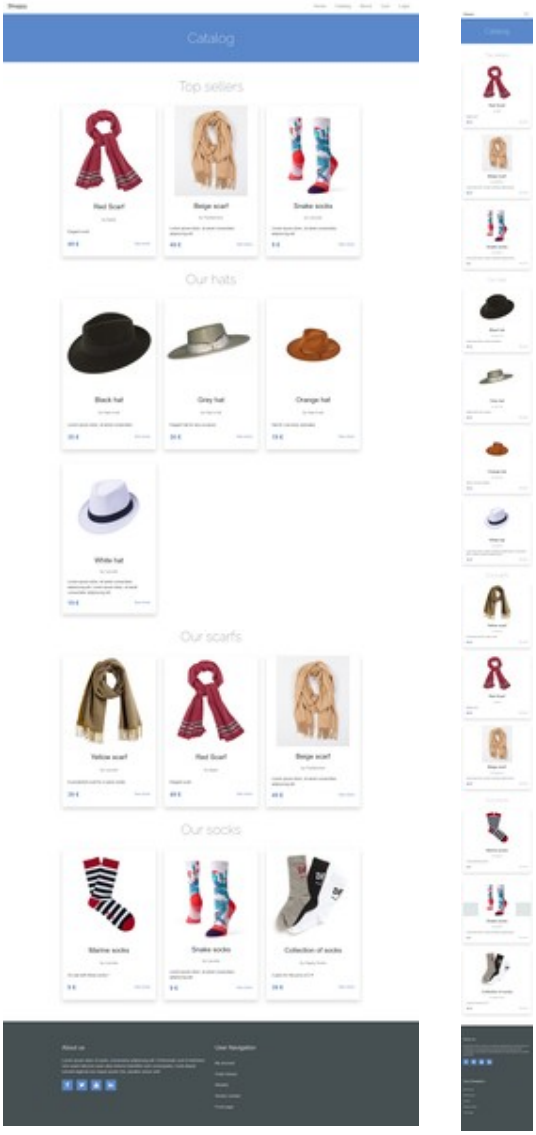
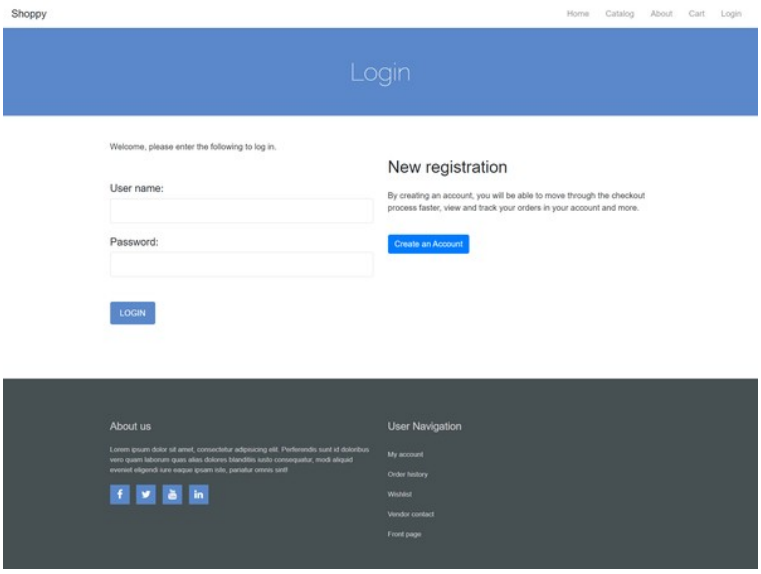
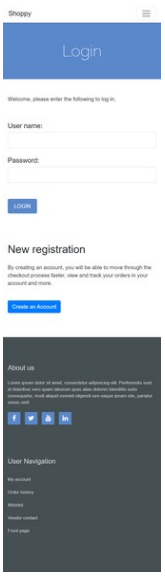


Fig 35a et 35b (ci-contre à droite) : captures d'écrans de la page 'Catalog' sur ordinateur de bureau et sur smartphone

Fig 36a et 36b (ci-dessous) : captures d'écrans de la page 'Login' sur ordinateur de bureau et sur smartphone





F. Bilan

1. Difficultés

Le gros problème de ce projet a été le manque de temps. Après avoir passé plusieurs semaines à développer le site OSmOSE, j'ai tardé à me lancer sur ce second projet en raison d'incertitudes sur mes possibilités d'intervention sur la partie *backend* du projet OSmOSE ainsi que des scrupules à mettre en partie de coté le travail que me demandait l'équipe qui m'avait chaleureusement accueillie. Je n'ai au final passé qu'environ 2 semaines à développer ce projet de site de e-commerce.

C'est pourquoi plusieurs parties de l'application méritent encore un certain nombre de développements : le CSS et l'esthétique sont assez sommaires (bien qu'entièrement *responsive*), le site ne propose qu'un nombre limité de fonctionnalités (bien qu'il soit dynamique) et la base de données mérite encore plusieurs approfondissements (bien qu'elle fournissent les données nécessaires au bon fonctionnement de l'application)

De surcroît j'ai dû développer le site de zéro et en utilisant la technologie Twig que je ne connaissait pas et en PHP orienté-objet, ce que je n'avais pas fait pour l'entièreté d'une application jusque là.

2. Suite

Je projette, dans les semaines à venir, de retravailler le projet afin d'ajouter des fonctionnalités habituelles aux site de commerce et finaliser la BDD afin de considérer avoir achever ce projet de site e-commerce en PHP orienté-objet sans *framework backend.*

Néanmoins ce projet n'aura, je pense qu'une valeur éducative. Cette application, qui ne compte environ qu'une trentaine de fichiers, ne pourrait pas constituer un choix sérieux dans le cadre d'un site de e-commerce professionnel tant sur le plan esthétique que sur celui des fonctionnalités. De même, il ne m'apparaît pas intelligent de se passer des nombreuses dépendances de qualités développé par des développeurs d'expériences disponible via ou directement intégrés dans les *frameworks* populaires et réputés comme Symfony ou Laravel. J'ajoute que ce sont de surcroît parmis les technologies les plus demandées en développement web.

De la même manière, même si au cours du développement et à plusieurs occasions j'ai mis en place des mesures pour sécuriser l'application et la base de données, il est évident que je n'ai pas les moyens de proposer une sécurisation au niveau des *frameworks* cités précédemment. La sécurité étant aujourd'hui, et encore plus demain, une partie absolument primordiale du développement web, il convient d'adopter les meilleures pratiques qui soient.

Malgré cela, ce projet m'aura permis d'en apprendre beaucoup sur la programmation orientée-objet et cette connaissance aura certainement de forts échos quand je débuterai de nouveau un projet Symfony ou Laravel.

Ce projet m'a aussi mis en position de largement augmenter ma capacité de travail afin d'être particulièrement efficace dans la période de temps très réduite dont je disposais pour finaliser une version fonctionnelle de l'application.

III. Conclusion

J’ai eu l’occasion, pendant ces 6 semaines, non seulement de mettre à profit mes connaissances mais d’apprendre énormément de choses tant sur le plan technique que sur le plan de la méthode. L’échange avec le client, les discussions techniques avec les développeurs senior, et le dynamisme de l’équipe ont sans doute été pour quelque chose à mes efforts continus pour arriver à de beaux résultats pour un stage à la durée si réduite, et ce, malgré la période de confinement qui n’était pas pour améliorer mon investissement a priori.

J’ai pu, avec ces deux projets, couvrir l’ensemble des compétences que j’avais pour objectif de travailler et identifier plusieurs sujets et technologies qui ont grandement captés mon intérêt. Nul doute que je passerai du temps à l’avenir pour revenir sur React, Twig, la programmation orientée-objet ou la conception de diagrammes tant dans une perspective professionnelle qu’en raison d’un attrait personnel.

Enfin je remercie pour leur accueil Dorian CAZAU et toute l’équipe OSE dont la sympathie et la disponibilité a été grandement appréciée tout au long de ce stage.

Florent BOUTONNET

26 mai 2021

à Brest