

Développeuse Web FullStack Symfony / ReactJS

NOM Prénom :
FRANK Shir

Responsable Pédagogique :
NIGRO Jean-Marc

Branche :
Informatique et Systèmes
d'Information

Semestre :
Automne 2021

Résumé (150 mots)

Mon stage s'est déroulé au sein de l'association Gaea21 en tant que Lead développeuse, dans le département IT.

Ma mission a consisté au développement Full-Stack du site web du projet "Répertoire Vert", à l'aide des frameworks Symfony et ReactJS. En tant que chef de projet, je devais également gérer l'équipe et avoir une vision globale des tâches.

La gestion du projet se faisait selon la méthode Agile, avec différentes phases répétées cycliquement :

Recueil des besoins du responsable de l'association, Mise en place d'un planning et répartition des tâches dans l'équipe, Mise en oeuvre, Réunion hebdomadaire avec le responsable et l'équipe et Déploiement de la version terminée du site.

Le but est de livrer une version du site web fonctionnelle permettant le référencement et l'évaluation de tous les produits et/ou services professionnels verts proposés dans une région définie, destiné pour le moment aux entreprises.

Entreprise : Gaea21

Lieu : *Avenue des Morgines 9, 1213
Petit-Lancy, Suisse*

Responsable : Yvan CLAUDE

Mots clés (cf Thésaurus) :

- Mise en place, mise en oeuvre
- Associations
- Informatique
- Gestion - Logiciels

Remerciements

Je tiens à remercier mon clavier, sans qui rien de tout cela n'aurait pu arriver.

Une attention particulière pour le café, qui m'a permis de tenir tout du long.

Sans oublier mon écran, sans qui je n'aurais jamais vu le bout du tunnel.

Sommaire

Remerciements

Introduction	1
---------------------	----------

I Sujet et place dans le service	2
---	----------

A Sujet	2
B Fonction occupée dans le service	3

II Déroulement du travail	4
----------------------------------	----------

A Description de la mission	4
B Antécédents	8
C Objectifs visés et méthode choisie	10
D Planning prévisionnel du travail	13
E Application de la méthode	15
F Résultats par rapport à l'objectif et planning réel	43

Conclusion	44
-------------------	-----------

Bibliographie

Table des figures

Liste des tableaux

Annexes	I
----------------	----------

A Documents d'organisation et de planification	I
B Le site web	I

Introduction

Gaea21 est une association genevoise à but non lucratif et un centre de recherche appliquée en développement durable, créé en 2005. Sa raison d'être est la mise en oeuvre du développement durable et de l'agenda 21. Cela se traduit par plusieurs objectifs :

- Sensibiliser et développer les compétences en Développement Durable, favoriser le changement de comportements
- Stimuler la création d'emplois et d'entreprises vertes et durables
- Recenser, évaluer et soutenir les acteurs du Développement Durable

Gaea21 crée donc des outils et applique des méthodes stimulant le changement de comportement et la transition accélérée vers un modèle économique et social responsable. Elle cible les ONG, individus/familles, entreprises et administrations/villes.

La structure possède une antenne à Thonon-Les-Bains (Haute-savoie) depuis 2013, s'occupant principalement de la recherche et du développement en sciences physiques et naturelles. Globalement, l'association a une portée plutôt régionale (Bassin genevois), mais les membres viennent du monde entier comme tout se fait en télétravail.

Elle compte environ 200 membres et comprend divers départements :

- Observatoire
- Administration & Juridique
- Formations
- Marketing
- Culture

L'ensemble de ces départements travaille sur une quarantaine de projets.

Je fais partie du département Observatoire et du sous-département IT, qui s'occupe du design et de la réalisation des sites web ou applications mobiles pour les différents projets.

I Sujet et place dans le service

A Sujet

A.1 Sujet défini avant mon arrivée

Avant le début de mon stage, il était convenu que je sois dans un premier temps formée à Symfony et ReactJS.

Ensuite, il était prévu que je travaille majoritairement sur de l'intégration WordPress, à savoir :

- Intégration de design sur un site wordpress existant
- Intégration d'un système de paiement sur WordPress
- Intégration d'un système de quiz sur WordPress
- Intégration de multiples formulaires (communication, inscription) sur un site Word-Press
- Intégration des réseaux sociaux(API) sur un site WordPress

Enfin, j'étais également censée développer un outil de gestion des Ressources humaines :

- Analyse des besoins en système informatique
- Gestion des fichiers
- Gestion des employés

A.2 Sujet réel

Dès mon arrivée, j'ai bel et bien eu droit à une formation ReactJS et Symfony en ligne. Mais j'ai ensuite été assignée au développement d'un site web Symfony et ReactJS, mais pas Wordpress. Je n'ai pas non plus eu à développer un outil de gestion des ressources humaines.

Mon réel sujet de stage correspondait donc au développement Full-Stack d'un site web avec Symfony et ReactJS, ainsi qu'à la gestion du projet et de l'équipe, ce qui me convenait parfaitement.

B Fonction occupée dans le service

Ma fonction dans le département IT de l'association était double. J'étais avant tout développeuse web et devait donc me charger de l'implémentation de diverses fonctionnalités du site web, en front comme en back, le tout en ReactJS et Symfony.

Mais j'ai également occupé la fonction de coordinatrice de niveau 3, c'est-à-dire que je gérais un sous-projet avec l'aide d'un collègue, en l'occurrence le projet Répertoire Vert. Mes missions étaient les suivantes :

- Accueillir les nouveaux arrivants
- Avoir une vision globale des tâches à accomplir et évaluer leur difficulté
- Distribuer les tâches à chaque nouveau sprint en tenant compte des objectifs de formation et des capacités de chacun
- Faire un planning pour respecter les deadlines
- Vérifier le travail de l'équipe
- Mettre le site en production deux fois par semaine
- Présenter l'avancée au client

II Déroulement du travail

A Description de la mission

A.1 Le site web Répertoire Vert

Ma mission concernait le site web du projet Répertoire Vert.

Le Répertoire Vert est un outil de référencement et d'évaluation des produits et services verts proposés dans un rayon de 130 km autour d'un point de référence (ici Genève). Il permettra au «consomm'acteur» de trouver le meilleur produit/service «vert» en toute confiance et à un prix raisonnable, et à l'entreprise proposant un produit/service, d'avoir une meilleure visibilité, une reconnaissance active, et de vendre davantage.

Le répertoire vert propose une répartition de l'activité économique « verte » en 28 secteurs principaux, comme Alimentation et Boisson, Green Building ou encore Santé et Bien-être.

Il détermine 4 niveaux de référencement basés sur une triple évaluation :

- Celle de l'association gaea21
- Celle des professionnels de la branche d'activité concernée
- Celle du public (les consomm'acteurs)

Ainsi, les entreprises peuvent se démarquer des produits « pseudo-verts » mis en vente sur le marché de manière abondante, et par conséquent, donner confiance , orienter et informer le consom'acteur de manière éthique. Ce projet se décline en un site web et une application mobile.



FIGURE 1 – Logo du Répertoire Vert

Le site s'adresse donc aux entreprises proposant des produits et/ou services verts, aux particuliers souhaitant trouver ces produits et services, mais aussi aux villes/régions qui peuvent voir les entreprises vertes disponibles sur leur territoire. Il se compose donc de 3 parties distinctes, 1 pour chaque cible. Je me suis occupée principalement de la partie dédiée aux entreprises.



FIGURE 2 – Page d'accueil entreprise

Sur le site, une entreprise doit pouvoir se connecter, ajouter des produits et/ou services, visualiser son profil et ses statistiques. Plus largement, on pourra visualiser l'ensemble des entreprises inscrites dans la région, mais également en fonction des secteurs d'activité. Le site devra aussi être un site e-commerce permettant d'acheter des produits et services verts.

A.2 Fonctionnalités à réaliser

Les tâches à réaliser se définissaient petit à petit. En effet il est important dans l'association de savoir prendre des initiatives : on nous donne les instructions dans les grandes lignes, à nous de voir précisément les différentes tâches à réaliser pour atteindre l'objectif posé.

Au niveau des fonctionnalités donc, je devais dans un premier temps m'occuper de la représentation des entreprises sur des cartes.

La 1ère devait figurer sur le profil de l'entreprise avec un point la représentant sur cette carte.

La 2nde devait être plus générale en affichant toutes les entreprises inscrites sur le Répertoire vert. Elle devait faire l'objet d'un filtre comportant plusieurs critères pour les entreprises : catégorie, sous-catégorie, code postal, prix des produits/services vendus, etc... .

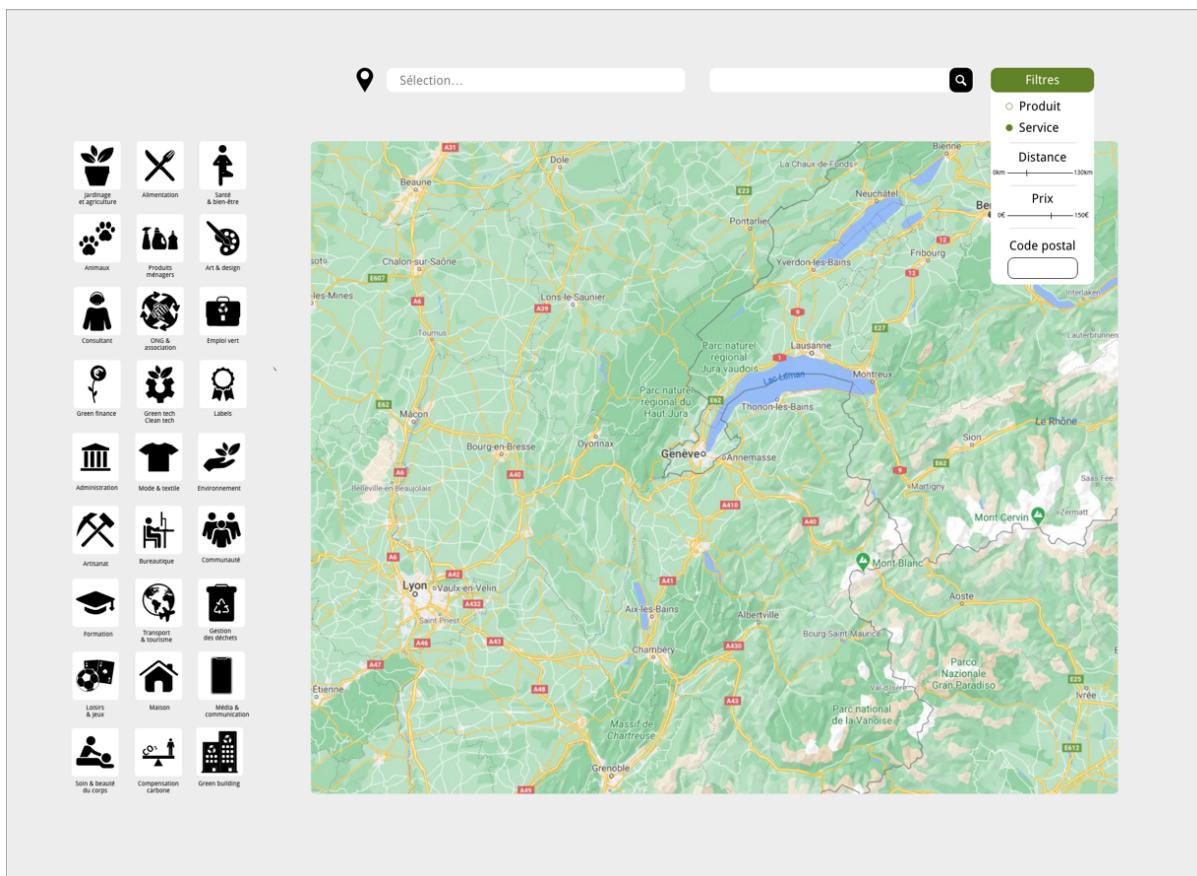


FIGURE 3 – Design de la page carte à implémenter

Mais d'autres tâches se sont peu après avérées beaucoup plus urgentes que la carte des entreprises. J'ai donc dû me charger de rendre l'inscription, la connexion et la réinitialisation de mot de passe fonctionnelles, mais aussi de faire en sorte que l'utilisateur puisse visualiser et gérer entièrement son profil et ses produits/services. J'ai aussi travaillé sur le responsive du site et sur la création de diverses nouvelles pages. Toutes mes tâches seront détaillées dans la partie E.

A.3 Gestion de projet

Outre le développement web, mon autre rôle majeur était celui de chef de projet du Répertoire Vert, se constituant d'une équipe de 3 à 4 autres développeurs. J'étais donc responsable du travail de mon équipe et devait m'assurer du respect des deadlines. Pour cela, il était parfois nécessaire de remplir des documents tels qu'un poker planning afin d'évaluer la difficulté des tâches et ainsi prévoir la durée de leur réalisation. Cette responsabilité impliquait aussi la vérification du travail de mes coéquipiers, pour qu'il soit présentable au client et puisse être mis en production.

II DÉROULEMENT DU TRAVAIL

2021-12-29

Features	Difficulté Moyenne pour l'équipe	Difficultés Estimée par le coordinateur	Difficulté Moyenne	Durée estimé	Durée réaliste	Durée réaliste (semi-confirmé)	Durée réaliste (junior)	Temporalité	Important
f1	21	21	21	31h 30 min	48h	64h	86h	UR	ID
f2	13	13	13	19h 30 min	30h	40h	54h	UR	ID
f3	7	7	7	10h 30 min	16h	21h	29h	UR	ID
f45	11	11	11	16h 30 min	25h	33h	46h	UR	ID
f87	13	13	13	19h 30 min	30h	40h	54h	UR	ID
f129	3	3	3	4h 30 min	7h 30 min	9h	13h	UR	ID
f171	3	3	3	4h 30 min	7h 30 min	9h	13h	UR	ID
f213	1	1	1	2h	3h	4h	6h	UR	GTD
f255	1	3	2	3h	4h 30 min	6h	9h	UR	GTD
f297	11	11	11	16h 30 min	25h	33h	46h	NM	GTD
f339	13	13	13	19h 30 min	30h	40h	54h	NM	GTD
f381	5	5	5	7h 30 min	12h	15h	21h	NM	GTD
f423	7	7	7	10h 30 min	16h	21h	29h	NM	GTD
f465	3	3	3	4h 30 min	7h 30 min	9h	13h	NM	GTD
f549	3	3	3	4h 30 min	7h 30 min	9h	13h	NM	GTD
f591	5	5	5	7h 30 min	12h	15h	21h	NM	GTD
f633	3	3	3	4h 30 min	7h 30 min	9h	13h	NM	GTD
f507	5	5	5	7h 30 min	12h	15h	21h	NM	GTD
Fonctionnalité type : Login	5			7h 30 min	12h	15h	21h		

FIGURE 4 – Planning Poker

Être chef de projet signifiait également accueillir les nouveaux arrivants, en leur appor-tant toute l'aide dont ils ont besoin pour prendre en main le projet, notamment expliquer où se trouvent les différents fichiers, la méthodologie à adopter, ou encore les commandes utiles en invite de commandes. J'ai eu l'occasion d'accueillir 5 nouvelles recrues. Mais même de manière plus générale, j'aidais souvent les membres de mon équipe dans leur tâche, ce qui m'a permis d'étudier des fonctionnalités sur lesquelles je n'étais normalement pas censée travailler.

Tout cela m'a permis d'acquérir une vision globale du projet, me permettant de cer-ner les différentes tâches restantes et comment elles devaient être réalisées.

Une autre responsabilité que j'avais était la mise en production du site web deux fois par semaine.

Cette étape de mise en production était très importante, car elle nous permettait d'ap-préhender d'éventuels bugs tout de suite plutôt que de les accumuler, ce qui assurait la qualité du rendu.

B Antécédents

Avant mon arrivée, le projet avait déjà été bien entamé, il a en effet débuté en Mars 2021, soit 5 mois auparavant.

- Une base de données était créée avec 59 tables, dont certaines remplies de données

<input type="checkbox"/> cart	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	1	InnoDB	utf8mb4_unicode_ci	32,0 kio
<input type="checkbox"/> cartline	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> category	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	27	InnoDB	utf8_unicode_ci	32,0 kio
<input type="checkbox"/> category_subcategory	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	373	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> commentreport	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> comment_product_fav	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	16,0 kio
<input type="checkbox"/> community	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio
<input type="checkbox"/> company	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	121	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> company_category	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	132	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> company_fav	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> company_favoris	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	11	InnoDB	utf8mb4_unicode_ci	16,0 kio
<input type="checkbox"/> coup_de_projecteur	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	4	InnoDB	utf8mb4_unicode_ci	16,0 kio
<input type="checkbox"/> cssstyle	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	48,0 kio
<input type="checkbox"/> depense	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	32,0 kio
<input type="checkbox"/> discussion	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	48,0 kio
<input type="checkbox"/> discussionquote	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8mb4_unicode_ci	64,0 kio
<input type="checkbox"/> doctrine_migration_versions	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	31	InnoDB	utf8_unicode_ci	16,0 kio
<input type="checkbox"/> event	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	32,0 kio
<input type="checkbox"/> event_fav	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	48,0 kio
<input type="checkbox"/> favoritecompany	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	48,0 kio
<input type="checkbox"/> favoriteproduct	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	48,0 kio
<input type="checkbox"/> fav_company	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	48,0 kio
<input type="checkbox"/> forumads	★	Parcourir	Structure	Rechercher	Insérer	Vider	Supprimer	0	InnoDB	utf8_unicode_ci	16,0 kio

FIGURE 5 – Extrait de la base de données du site Répertoire Vert

- De nombreuses pages avaient été intégrées conformément aux designs produits et validés

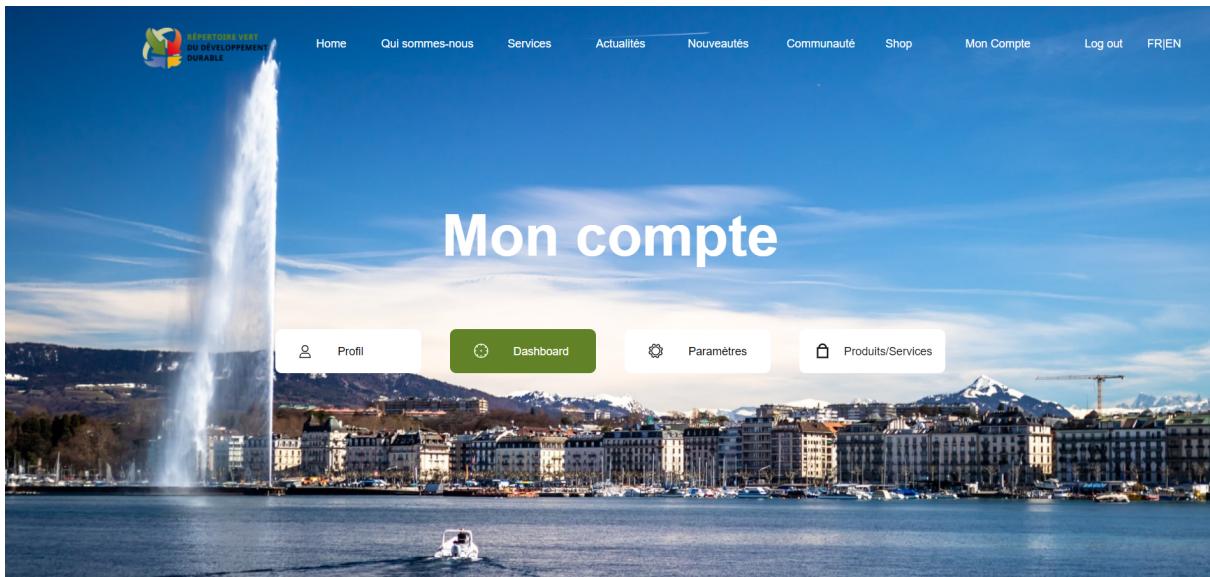


FIGURE 6 – Exemple d’entête de page intégrée

- Certaines fonctionnalités avaient été implémentées partiellement ou complètement. Par exemple, l’inscription d’une entreprise était fonctionnelle à mon arrivée, mais certains champs ne l’étaient pas, comme l’upload d’un logo pour l’entreprise. La plupart des pages intégrées n’étaient pas terminées, car les informations étaient entrées en dur, ou les fonctionnalités n’étaient pas abouties.

- Une API avait été créée par l’un des précédents membres de l’équipe et était utilisée pour tout ce qui concernait le compte d’un utilisateur, à savoir l’inscription, connexion, changement de mot de passe, etc...

Pourquoi une API ?

Un compte sur le site du Répertoire Vert est transversal, c'est-à-dire qu'il peut être utilisé sur les autres sites créés par Gaea21. Par conséquent, les informations de connexion des utilisateurs sont stockées dans une autre base de données. Il fallait donc envoyer les données nécessaires à l'API via une requête Ajax ou PHP pour qu'un traitement soit fait dans la base de données distante. Cette API, dont voici la [documentation](#), était donc déjà complètement fonctionnelle.

Etant donné le turn-over important dans l’association, l’un des principaux défi était de se familiariser avec le code existant et les nombreux fichiers. Certaines fonctionnalités étaient déjà créées et pouvaient donc être réutilisées. Mais en parallèle, beaucoup de fichiers ou morceaux de code étaient inutilisés, et certains se répétaient. Il fallait donc savoir distinguer ce qui était utile de ce qui était à ignorer.

C Objectifs visés et méthode choisie

C.1 Les objectifs

Comme mentionné en partie II)A.2, les objectifs se définissaient au fil du stage. En effet, le grand objectif était l'avancement du site, mais les étapes n'étaient pas clairement définies dès le début du stage. J'ai listé l'ensemble des objectifs qui m'ont été donnés au fil des mois, il s'agit donc d'objectifs réalistes.

Avant toute chose, mon 1er objectif était l'accomplissement des formations STAFF 1 et STAFF 2 en 1 semaine, puis la formation ReactJS et Symfony en 3 semaines. La formation Staff 1 concernait le développement durable et les projets de Gaea21, et la Staff 2 portait sur les outils Google très utilisés durant tout le stage.

A l'échelle de Gaea21, le 1er objectif était de livrer la version 0 du site. Cette version devait permettre à une entreprise de :

- S'inscrire et se connecter
- Visualiser son profil et modifier ses informations et son mot de passe
- Désactiver son compte
- Ajouter des produits et/ou services
- Visualiser ses produits/services, les supprimer et les modifier
- Visualiser ses statistiques, comme le nombre de clics sur ses produits
- Parcourir les différentes catégories et sous-catégories, et voir la liste des entreprises appartenant à chaque sous-catégorie.
- Visualiser les profils des autres entreprises ainsi que leurs produits/services

Cependant, je n'ai été au courant de cet objectif que 2 semaines après mon intégration au projet. Le 1er objectif que l'on m'avait donné pour le projet était celui des cartes mentionnées en partie A, qui devaient être prêtes pour la version 1.

En effet, la version 0 était déjà censée être quasiment terminée à ce moment-là, c'est pourquoi on m'a attribué une tâche moins urgente. 2 semaines plus tard, on m'a donc demandé d'aider un collègue à mettre en commun toutes les fonctionnalités pour mettre la V0 en production (autrement dit, merge toutes les branches Git sur la la branche principale). C'est alors que nous nous sommes rendu compte que la V0 n'était pas du tout aboutie, c'est-à dire que presque aucune fonctionnalité de la liste ci-dessus n'était terminée. Or nous n'étions plus que 2 sur le projet. Il fallait donc basculer sur ces tâches en priorité.

Une fois la V0 terminée en version ordinateur, L'objectif était de faire le responsive de toutes les pages intégrées.

En parallèle, pour le lancement du site, il fallait aussi ajouter à la base de données une soixantaine d'entreprises qui avaient déjà été inscrites, mais qui ont été supprimées durant le développement. Elles devaient alors recevoir un mail afin de réinitialiser leur mot de passe.

Pour finir, nous devions également entamer la page Actualités ainsi que l'inscription à la newsletter de Gaea21, à implémenter normalement pour la V1, mais qui ont été jugées finalement plus urgentes que le responsive de la V0.

C.2 La méthodologie

Durant toute la durée du stage, il y avait un suivi régulier, permettant une gestion de projet selon la méthode agile.

Ainsi, cela fonctionnait par cycles répétés :

Tous les jeudis, une réunion avec l'ensemble du département IT et le client (Yvan, le responsable de l'association) avait lieu afin de recueillir ses besoins pour le site.

Si nécessaire, un poker planning était réalisé pour répartir les tâches efficacement selon les difficultés et le temps prévu.

Tous les jours, une réunion de suivi avec le tuteur et l'équipe permettait de partager notre avancée et obtenir de l'aide sur des points de blocage.

Tous les mardis, une réunion avec le département Design permettait de mettre au clair certains points ou de faire des commandes, car le développement d'un site est étroitement lié au design.

Une fois toutes les 2 semaines, une réunion transversale pour le projet Répertoire Vert avait lieu.

Pour finir, tous les lundi et jeudi, une mise en ligne du site était nécessaire pour garder la version en ligne toujours à jour et présentable au client.

D'un point de vue technique, il fallait mettre à jour le site sur un serveur Linux distant à l'aide de Git, un logiciel de gestion de versions que j'ai particulièrement utilisé durant ce stage.

Justement, il était important de faire des **commits et push** au moins 1 fois par jour, pour garder un contrôle des versions du site. Chaque membre de l'équipe possédait une branche à son nom. Plusieurs fois par semaine, un merge des toutes les branches sur la branche de développement était réalisé, avant de faire un merge sur master de la branche de Développement. En effet, la branche master est celle utilisée pour la mise en production, elle doit donc être propre, fonctionnelle et régulièrement mise à jour.

D'autre part, il ne fallait pas négliger le remplissage les outils de suivi, à savoir :

- **le journal de bord** pour les heures et les tâches réalisées par jour/semaine/mois
- **le tableau des tâches** pour organiser les tâches à faire/en cours/faites, et répertorier le temps mis pour chacune d'entre elles.

Des points début, mi-stage et fin de stage étaient réalisés avec le tuteur et la coach RH pour vérifier le remplissage des outils et voir si tout allait bien.

D Planning prévisionnel du travail

D.1 1er planning prévisionnel

Les tâches étant ajoutées et révélées au fur et à mesure, il n'y avait pas de vision globale réaliste du projet en amont.

Le planning ci-dessous présente donc les prévisions ou souhaits d'Yvan et/ou de mes tuteurs au début de mon stage.

Mais comme toutes ces personnes travaillaient sur d'autres projets, elles n'étaient pas vraiment au courant de l'état d'avancement précis du projet, ce qui explique la grande différence entre ce planning et les objectifs listés précédemment.

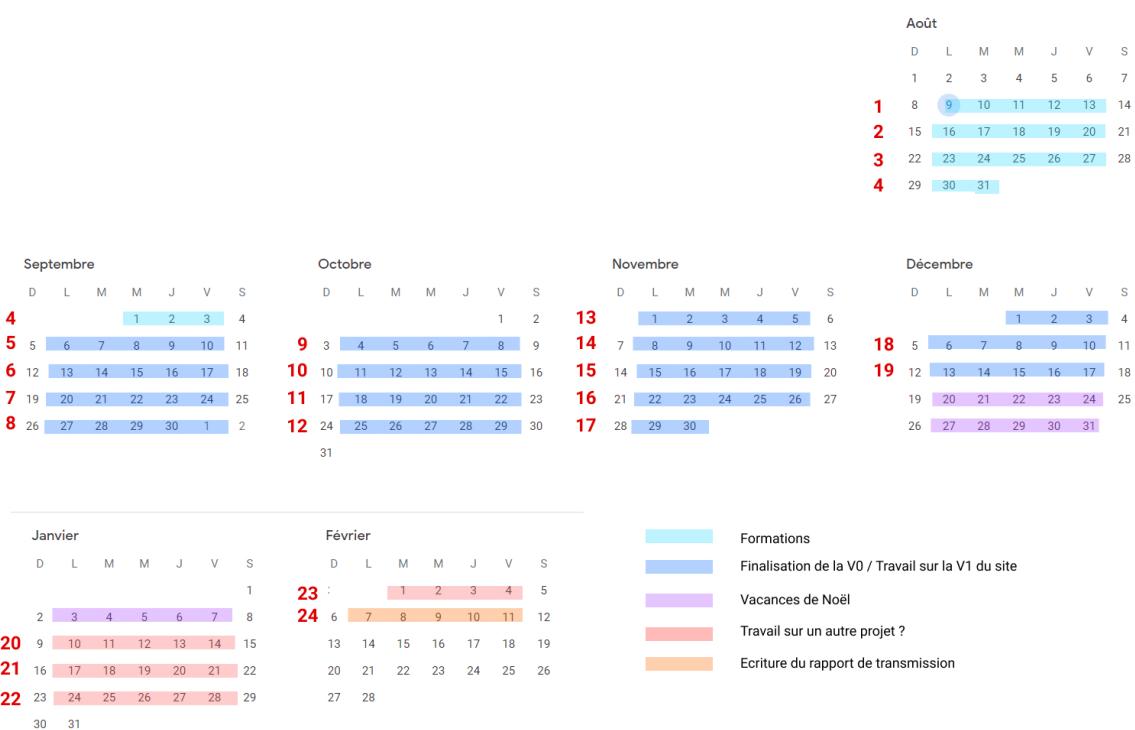


FIGURE 7 – Planning prévisionnel

Il était donc prévu qu'après ma formation, je travaille sur la version 1 du site (en commençant par les cartes), pendant que le reste de l'équipe finalisait la V0. Je devais les aider également si besoin. Cette phase devait durer jusqu'aux vacances de Noël.

Pour information, la V1 devait contenir :

- Les pages "Qui sommes-nous", "Nouveautés", "Actualités" et "Communauté"
- La possibilité de noter des produits et entreprises, ainsi que de les mettre en favoris.

Au retour des vacances, on ne m'a pas vraiment précisé ce que je ferais, mais les versions urgentes du Répertoire Vert devant être prêtes, on m'aurait sûrement mise sur un autre projet.

La dernière semaine de stage était consacrée à la rédaction d'un rapport de transmission, permettant aux suivants de reprendre mon code.

D.2 2nd planning prévisionnel

Mi-septembre, le planning a pu évoluer en une version un peu plus réaliste.

Lorsque tout le monde a compris que la V0 n'était en réalité pas du tout terminée, Yvan nous avait fixé la deadline pour cette version au 7 octobre.

Après cela, il fallait rendre les pages fonctionnelles responsive. Et une fois cela terminé, nous pouvions nous atteler à la V1.

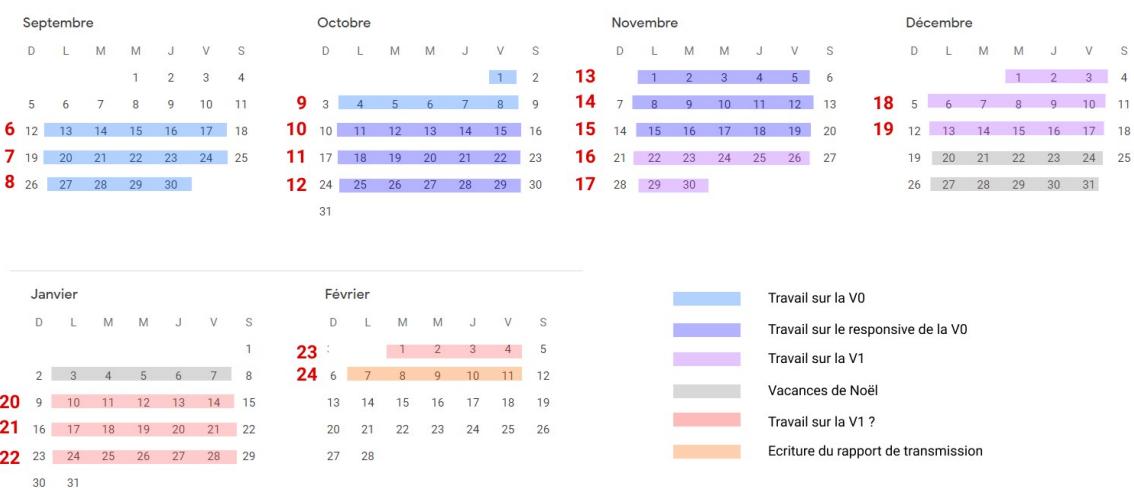


FIGURE 8 – Planning prévisionnel de septembre

E Application de la méthode

E.1 Formation React et Symfony

Le 1er mois de stage était donc consacré à la formation ReactJS et Symfony 5.

J'ai commencé par ReactJS, en suivant un cours du site Graphikart sur ce framework. J'y ai appris les bases, ainsi que des notions plus avancées, comme les hooks, contextes, etc... Des petits exercices faits par l'association m'ont permis de consolider mes acquis.

1 semaine après, j'ai débuté la formation Symfony, qui se faisait quant à elle sur le site SymfonyCasts, avec des vidéos également. J'ai donc pu apprendre les bases de ce framework, avec le principe de routes, services, bundles, contrôleur, mais aussi la partie bases de données avec Doctrine.

La finalité de ces deux formations était un exercice dont l'objectif était de réaliser un CRUD (Create, Read, Update, Delete) liant Symfony et React. Il était donc demandé :

- D'afficher dans un tableau les informations d'une table "utilisateur" de la base de données avec ReactJS, après l'avoir créée.
- D'ajouter les fonctionnalités supprimer et modifier pour chaque ligne, et aussi ajouter un nouvel utilisateur.
- D'ajouter les possessions de l'utilisateur, visibles sur une autre page en cliquant sur le nom d'un utilisateur, et de pouvoir en ajouter.
- De calculer l'âge de l'utilisateur grâce à un service.

La 1ere chose à faire était la **création de l'entité Utilisateur** avec tous ses attributs, en ligne de commande.

```
php bin/console make:entity
```

Puis, une fois la page créée grâce à un fichier **Twig** (front-end) et une fonction dans le **contrôleur** indiquant la route vers cette page (back-end), il fallait afficher les informations de la base de données dans un tableau html, en passant par **React**.

Pour ce faire, il fallait tout d'abord récupérer l'ensemble des informations de chaque utilisateur via le contrôleur. J'ai donc fait **une requête Doctrine** via le Repository de l'entité Utilisateur.

Puis, il fallait envoyer ces données au javascript sous forme de tableau json. Chaque page de l'application correspondait à un **composant React**. Le tableau affichant les utilisateurs était donc un composant récupérant les données dynamiquement par son état, via une **requête Ajax GET**.

Quant à la suppression d'un utilisateur, on appelle la fonction de suppression située dans le contrôleur Symfony via une **requête Ajax POST**.

Les fonctionnalités d'ajout et de modification reposent sur le même principe.

Pour calculer l'âge il suffisait d'importer le service en question dans le contrôleur afin d'utiliser la fonction.

Voici donc le résultat de ces 1ères fonctionnalités :

Utilisateur index

Id	Nom	Prenom	Email	Adresse	Telephone	Age		
1	Frank	Shir	shir.frank@utt.fr	12 rue Marie Curie	0938382929	21	Modifier	Supprimer
2	Raimbault	Gerard	test	125 Cours de Verdun	test	21	Modifier	Supprimer

[Ajouter un utilisateur](#)

FIGURE 9 – Accueil de l'appli

Edit Utilisateur

Nom : <input type="text" value="Frank"/>	Prénom : <input type="text" value="Shir"/>	Email : <input type="text" value="shir.frank@utt.fr"/>	Adresse : <input type="text" value="12 rue Marie Curie"/>	Téléphone : <input type="text" value="0938382929"/>
Date de naissance : <input type="text" value="29/04/2000"/>	<input type="checkbox"/>	Mettre à jour		
back to list				

FIGURE 10 – Modification d'un utilisateur

Pour ajouter les possessions de chaque utilisateur, une nouvelle table/entité était nécessaire, ayant une relation **ManyToOne** du côté de la possession. Cela signifie qu'un utilisateur peut avoir plusieurs possessions, mais qu'une possession ne peut appartenir qu'à un seul utilisateur.

Il était facile de récupérer les possessions d'un utilisateur selon son id, mais la difficulté était la sérialisation des données à envoyer au JS sous forme de tableau json. En effet, on récupérait un tableau de tableaux. Il fallait donc boucler dans le 1er tableau pour sérialiser chaque possession.

Une fois cela terminé, j'ai développé la fonctionnalité d'ajout d'une possession, comme pour les utilisateurs.

Utilisateur

Id	1
Nom	Frank
Prenom	Shir
Email	shir.frank@utt.fr
Adresse	12 rue Marie Curie
Telephone	0938382929
Âge	21

Possessions

Id	Nom	Valeur	Type
1	Chaise	4	Meuble

[back to list](#) [edit](#) [Ajouter une possession](#)

Delete

FIGURE 11 – Page de l'utilisateur avec ses possessions

Une fois cet exercice validé, j'ai pu être intégrée au projet Répertoire Vert.

E.2 Cartes des entreprises

Carte d'1 entreprise

Ma toute première tâche était donc l'intégration d'une carte permettant à l'entreprise de se localiser visuellement sur son profil.

Avant toute chose, j'ai étudié les différents fichiers existants, à la recherche d'un éventuel code que je pourrais réutiliser.

Il en existait effectivement un, une page avait été créée, contenant une carte avec toutes les entreprises de la base de données. **LeafletJS** était utilisée, une bibliothèque JavaScript libre de cartographie.

Il m'a simplement fallu modifier l'url de la requête ajax, afin de récupérer les coordonnées de non pas toutes les entreprises, mais d'une seule selon son id. J'ai également copié ce code dans un fichier javascript à part.

Fiche entreprise ▶ Lalalande
Ajouter aux favoris  0 favoris

Aucun logo sélectionné

Lalalande
9 rue Lac Saint-André
Le Bourget-du-Lac
32767
sfgtfd



Description de l'entreprise :

- ▶ Véhicules électriques

Vision du développement Durable

Comment résumeriez-vous votre philosophie d'entreprise au sujet du développement durable ?

- ▶ La planète est notre ressource la plus précieuse

FIGURE 12 – Résultat de la page profil avec la carte

Cependant, pour afficher l'emplacement de l'entreprise, il était nécessaire de récupérer les coordonnées GPS à partir de l'adresse, et ce dès l'inscription.

Après m'être familiarisé avec le code de l'inscription, j'y ai ajouté une requête Ajax vers l'API **OpenStreetMap** : "un projet collaboratif de cartographie en ligne qui vise à constituer une base de données géographiques libre du monde, en utilisant le système GPS et d'autres données libres". En mettant l'adresse entrée par l'utilisateur en paramètre, on peut ainsi récupérer les latitude et longitude associées. Ces données sont ensuite envoyées au contrôleur pour être enregistrées en base de données.

Carte de toutes les entreprises

La prochaine tâche concernait la page affichant toutes les entreprises sur une carte, avec la possibilité de les filtrer, et d'obtenir plus d'informations sur chaque entreprise :

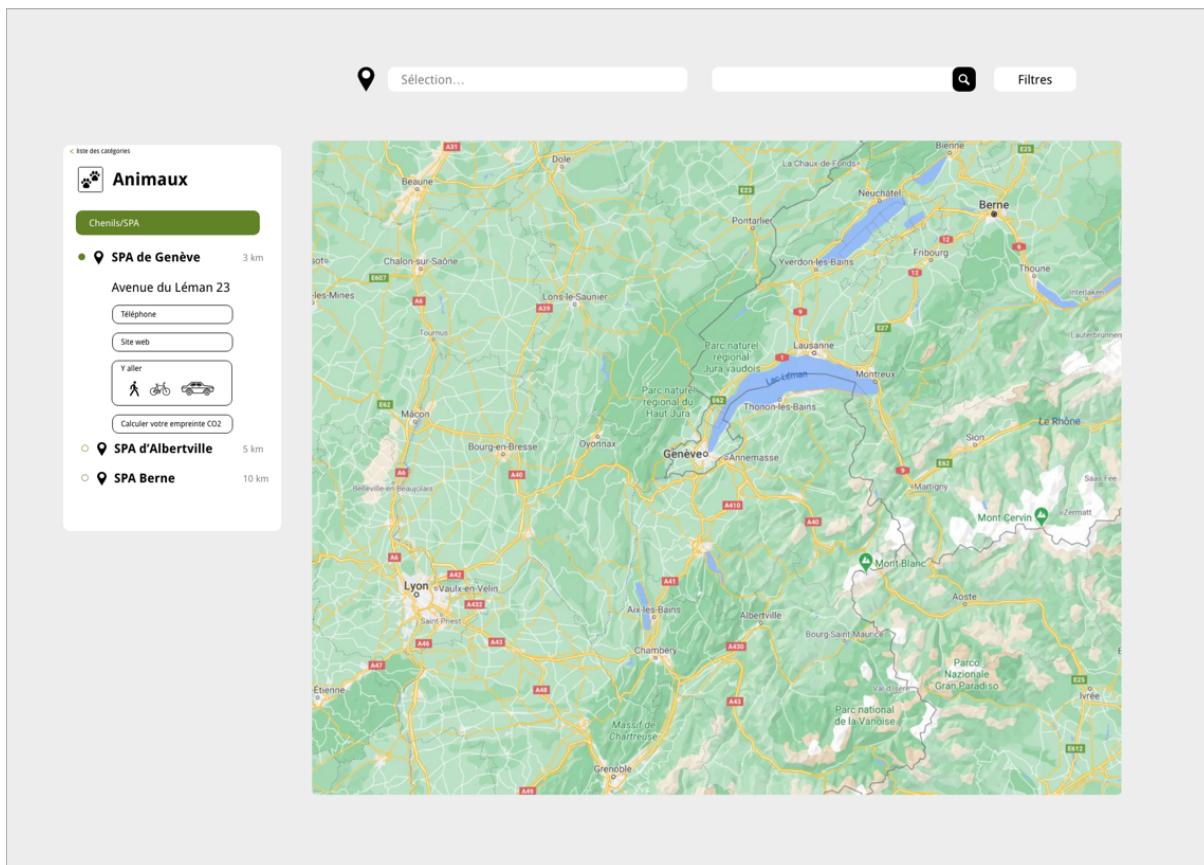


FIGURE 13 – Design page carte

J'ai décidé d'implémenter cette fonctionnalité en ReactJS, car l'affichage est considérablement modifié à chaque clic.

On a donc un composant principal **MapPage**, se chargeant d'afficher la carte, les bons points des entreprises et de les filtrer selon les critères reçus par les autres composants. L'autre composant appelé par le principal correspond au "Side Menu", qui s'affiche de 3 manières dans l'ordre ci-dessous, grâce aux clics sur les différents boutons.

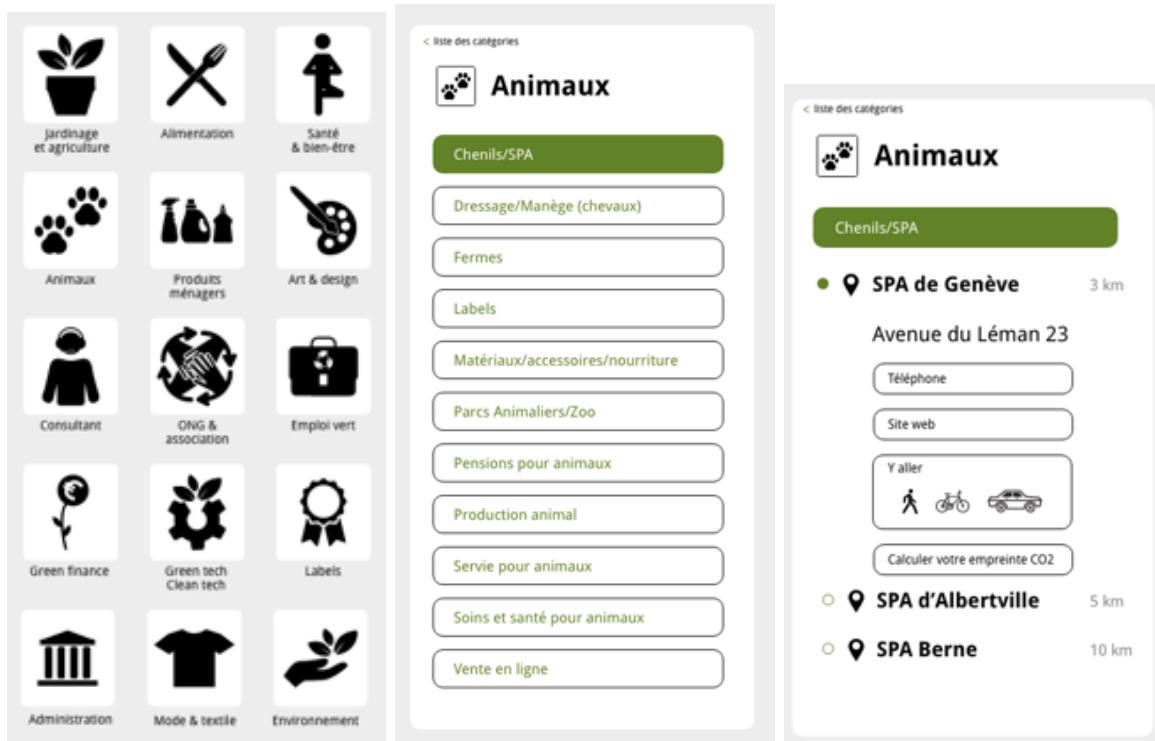


FIGURE 14 – Affichages successifs du side menu

On a d'abord les catégories, puis les sous-catégories, et enfin les entreprises correspondant à la sous-catégorie cliquée.

Le composant principal **SideMenu** appelle le sous-composant **SubCategoryMenu**, qui gère les 2nd affichage. Il appelle lui-même le sous-sous-composant **SubCategoryDetail**, qui appelle **CompanyDetail** pour le 3ème affichage.

Bien sûr, toutes les informations du menu sont affichées dynamiquement et donc récupérées par requête Ajax, et stockées dans les **états** des différents composants.

On commence par l'ensemble des catégories dès le "montage" du composant (méthode `componentDidMount`). Au clic de l'une des catégories, on récupère toutes ses sous-catégories, et au clic d'une sous-catégorie, les entreprises associées. Mais la relation Sous-catégories/Entreprises ne se fait pas directement. Il faut passer ici par les produits.

En effet lorsqu'une entreprise s'inscrit, elle choisit ses catégories, mais pas ses sous-catégories. Elles s'ajoutent progressivement au fur et à mesure que l'entreprise ajoutera des produits et services. Ces derniers sont bien associés directement à des sous-catégories.

Pour récupérer les entreprises appartenant à une sous-catégorie donc, j'ai dû ajouter dans la table "product" un attribut `company_id`, car l'attribut initial "owner" renvoyait à un individu de l'entité "user". Or un produit appartient forcément à une entreprise. Ce nouvel attribut lie les 2 tables par une relation `OneToMany` du côté de l'entreprise.

A savoir

L'entité Company, au même titre que Person, hérite de User.

C'est ici que je me suis heurtée à un problème : Lorsque l'on veut faire migrer de nouveaux changements de structure dans la BDD, un **fichier de migration** est créé. Ce fichier contient les règles SQL à exécuter lors de la nouvelle migration. Mais dans ce projet et pour une raison inconnue, de nombreuses autres règles déjà exécutées dans la base de données sont écrites. Cela empêche donc l'exécution du fichier, car certaines règles ne peuvent être exécutées 2 fois. Il fallait donc supprimer manuellement toutes les règles inutiles du fichier généré.

Je récupère les entreprises d'une sous-catégorie via le Repository de l'entité Company. Ici, on joint la table Company à la table Category pour garder les entreprises de la catégorie choisie. Puis on joint le tout à la table Produit et à la table Subcategory, pour ne garder que les entreprises qui ont des produits appartenant à la sous-catégorie choisie.

En plus des données des bonnes entreprises affichées, il fallait noter la distance entre la position de l'utilisateur et celle de l'entreprise. Dans le composant CompanyDetail, une fonction se charge donc de calculer la distance à vol d'oiseau à partir des latitudes et longitudes.

On obtient les coordonnées GPS de l'utilisateur grâce à la librairie Leaflet, depuis le composant principal **MapPage**, et on les envoie au SideMenu par des propriétés lors de son appel.

```
mymap.locate();
mymap.on('locationfound', this.getUserLocation);

getUserLocation(e) {
    this.setState({
        userLocation: e.latlng
    })
}
```

FIGURE 15 – Code de récupération de la localisation de l'utilisateur

Le filtre

En cliquant sur une catégorie dans le Side-Menu, les entreprises sur la carte devaient être filtrées pour ne montrer que celles appartenant à cette catégorie.

J'ai commencé par créer une **fonction de filtrage** assez générale dans le composant MapPage.

Il s'agit d'un filtrage exclusif, c'est-à-dire que l'élément doit respecter **tous** les critères et non au moins 1. Il tient compte des **critères** et des **paramètres**. Le critère utilisé ici

est "categories", car on filtre selon les categories. Le paramètre est quant à lui la valeur de la catégorie choisie.

On passe par plusieurs étapes imbriquées :

- On parcourt chaque entreprise à filtrer
- On parcourt chaque critère (même si ici il n'y en a qu'un) souhaité
- On parcourt chaque paramètre souhaité de ce critère
- On parcourt les valeurs de catégories que l'entreprise courante possède
- Si elle possède le paramètre souhaité, le paramètre est respecté
- Si tous les paramètres sont respectés, le critère est respecté
- Si tous les critères sont respectés, on peut ajouter l'entreprise au tableau filtré

L'affichage des marqueurs filtrés

Une fois le tableau des entreprises filtrées généré, il faut supprimer tous les marqueurs inutiles de la carte.

Pour cela :

- On parcourt chaque marqueur actuel de la carte
- On parcourt chaque entreprise filtrée
- Si le marqueur et l'entreprise filtrée ont les memes coordonnées GPS et le même nom, on garde le marqueur
- Sinon on le supprime de la carte

Comme la fonction qui filtre et affiche les bons marqueurs doit être appelée depuis le composant SideMenu et que les paramètres doivent y être définis également, on envoie toutes les informations nécessaires comme ci-dessous :

```
<SideMenu data={this.state.dataCompanies}
          criterias={this.state.criterias}
          setParams={(newParam, callback) => this.setState(state => {
            param: newParam
          }, callback)}
          params={this.state.param}
          filter={this.filterShowCompanies}
          init={this.reinitMap}
          userLocation={this.state.userLocation}/>
```

FIGURE 16 – Code d'envoi des propriétés nécessaires au filtrage au SideMenu

La propriété setParams permet au SideMenu d'ajouter la valeur de la catégorie cliquée à l'état du composant père MapPage, et d'appeler la fonction de filtre une fois l'état bien mis à jour : la fonction setState est une **fonction asynchrone**, c'est pourquoi elle accepte un paramètre de type callback.

Les boutons "Y aller" et "Calculer votre empreinte carbone" du 3ème SideMenu ont été intégrés, mais je n'ai pas eu le temps de les rendre fonctionnels, comme je devais finalement m'occuper de la V0 du site.

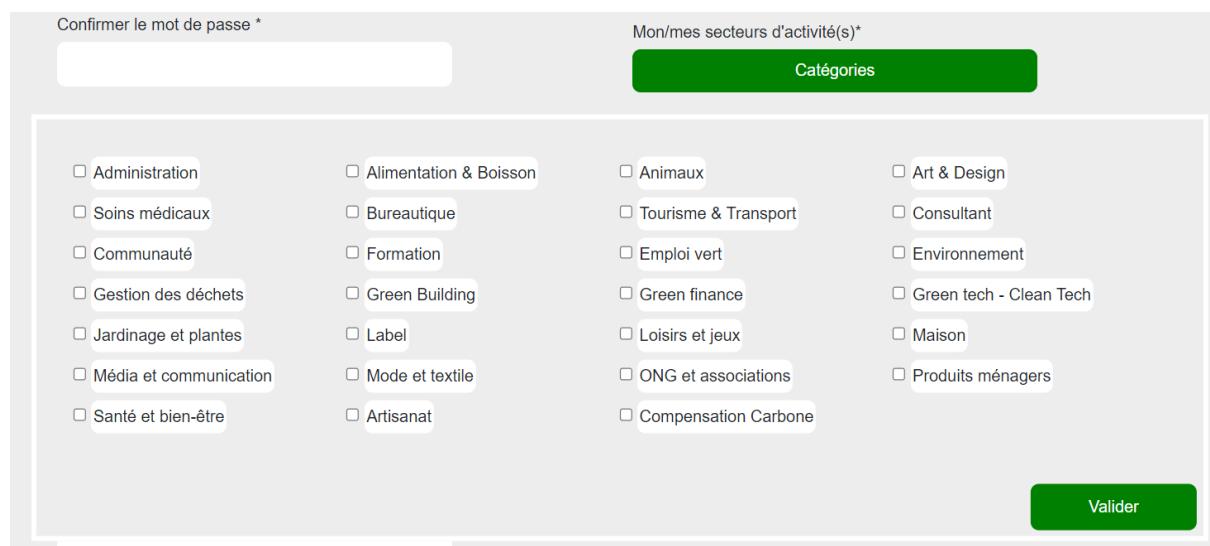
E.3 Rendre inscription et connexion fonctionnelles

La principale fonctionnalité à terminer était l'inscription et la connexion d'une entreprise.

L'inscription

Bien que le principal ait déjà été codé, l'inscription n'était pas encore fonctionnelle, et la gestion des erreurs n'était pas terminée. Il s'agit d'un formulaire fonctionnant "manuellement" avec Javascript.

Je me suis tout d'abord occupée du champ permettant d'**ajouter des catégories**. Il avait déjà été intégré : il s'agit d'une pop-up avec une checkbox pour chaque catégorie.



The screenshot shows a modal window with a light gray background. At the top left is a field labeled "Confirmer le mot de passe *". At the top right is a field labeled "Mon/mes secteurs d'activité(s)*". Below these fields is a green button labeled "Catégories". The main area contains a list of categories, each with a checkbox next to it. The categories are arranged in four rows:

- Administration, Alimentation & Boisson, Animaux, Art & Design
- Soins médicaux, Bureau, Tourisme & Transport, Consultant
- Communauté, Formation, Emploi vert, Environnement
- Gestion des déchets, Green Building, Green finance, Green tech - Clean Tech
- Jardinage et plantes, Label, Loisirs et jeux, Maison
- Média et communication, Mode et textile, ONG et associations, Produits ménagers
- Santé et bien-être, Artisanat, Compensation Carbone

At the bottom right of the modal is a green "Valider" button.

FIGURE 17 – Pop-up de choix des catégories

Pour conserver les choix de l'utilisateur, j'ai fait une requête ajax pour récupérer toutes les catégories de la base de données, parmi lesquelles j'ai bouclé afin de récupérer l'état de chaque checkbox. Si la checkbox est cochée, alors on ajoute le nom de la catégorie associée à un tableau.

```
let categories = [];
$.get('/rest/category', function(dataCategories) {
    dataCategories.forEach(category => {
        let checkbox = document.getElementById(category.name)
        if(checkbox.checked){
            categories.push(category.name)
        }
    })
})
```

FIGURE 18 – Code d'enregistrement des catégories

Ce tableau sera envoyé à la validation du formulaire avec toutes les autres données au contrôleur. Là, on récupère les catégories avec chaque nom récupéré, et on l'ajoute à la

liste de catégories de l'entreprise, dans la base de données.

J'ai ensuite fait fonctionner les boutons radio de demande d'évaluation.

Je souhaite être évalué par gaea21 : *

En savoir plus >

Oui Non

FIGURE 19 – Champ de demande d'évaluation

Il suffisait d'associer un booléen (vrai/faux) à l'état des boutons radio, puis d'assigner un niveau à l'entreprise selon cette valeur, et d'envoyer ces 2 variables au contrôleur.

Le menu déroulant de zone d'influence était à l'origine vide, j'ai donc communiqué avec le responsable du projet pour déterminer les options de ce champ.

Zone d'influence géographique *

--Merci de sélectionner une option--



FIGURE 20 – Champ de demande d'évaluation

Le lien "En savoir plus" sur la figure 19 devait renvoyer à une page statique présentant les tarifs et niveaux de référencement auxquels une entreprise peut appartenir. J'ai entièrement intégré cette page avec Sass notamment.

Niveaux	Tarifs entrepreneurs (entreprises en création et/ou start-up de moins d'un an)	Tarifs associations	Tarifs collectivité locales publiques	Tarifs entreprises
N.0	GRATUIT	GRATUIT	GRATUIT	GRATUIT
N.1	CHF 150.-	CHF 100.-	Subventions ou dons en nature	CHF 250.-
N.2	CHF 450.-	CHF 350.-	Subventions ou dons en nature	De CHF 700.- à CHF 1000.- selon le chiffre d'affaire

Niveaux de référencement du Répertoire

N.0 Acteur externe <ul style="list-style-type: none"> • Inscription gratuite. • Pas d'évaluation. Pas de recommandation gaea21 Référence mais pas d'évaluation 	N.1 Membre de l'association gaea21 <ul style="list-style-type: none"> • Évaluation : - questionnaire - feedback client. • Peer - reviewing : - évaluation par des entreprises proposant des services similaires. Recommandation gaea21 niveau 1 En cours d'évaluation 	N.2 Membre du réseau gaea21 <ul style="list-style-type: none"> • Signature de la charte qualité de gaea21 • Évaluation : benchmark = détermination d'un prix par produit/service. • Audit environnemental tous les 3 ans Recommandation gaea21 niveau 2 Recommandation gaea21 
---	--	---

FIGURE 21 – Page Tarifs/Niveaux

Je me suis ensuite assurée que tous les champs marqués obligatoires le soient bien dans le code Javascript.

Ainsi, chaque champ marqué avec une * doit être rempli, sinon la requête exécutant l'inscription ne sera pas appelée.

De même, l'adresse mail entrée dans le champ "Confirmation d'adresse mail" doit être identique à celle du précédent, donnée qui sera stockée dans un booléen.

Pour le contrôle du format de l'adresse mail, j'ai revu l'expression regex existante, car toutes les adresses mail n'étaient pas acceptées :

/ ^([\s]+)@([\s]+\.\w{2,4})/ i

\S renvoie à tous les caractères autres que les espaces.

A chaque validation, j'ai ajouté l'affichage d'un loader pour plus de visuel. J'ai aussi vérifié que l'affichage des différents messages d'erreurs était cohérent, et j'en ai ajouté certains, notamment si l'adresse entrée n'existe pas, ou que les 2 adresses mail ne sont pas identiques.

Une fois l'inscription validée, un mail doit être envoyé afin que l'utilisateur puisse confirmer son nouveau compte. Le code d'envoi de mail avait déjà été préparé.

Dans le contrôleur gérant les Users (UserController), j'ai créé une fonction appelée au clic sur le lien du mail. Cette fonction fait une requête HTTPS vers la fonction de l'API GaeaUser permettant de confirmer un compte. Le lien contient un token unique permettant de confirmer le bon compte de manière sécurisée. Une fois le compte confirmé, l'utilisateur est redirigé sur une page de succès, que j'ai intégrée.



FIGURE 22 – Page de succès de confirmation de compte

La connexion

La connexion, se faisant par un formulaire traité en javascript comme l'inscription, était déjà fonctionnel à mon arrivée.

Seulement, elle se faisait par le biais du menu, qui a été refait en ReactJS par un membre. Mais le lien avec la fonction initiale n'avait pas été mis en place. Je me suis donc occupée de la faire fonctionner simplement en important la fonction et en l'appelant dans la navbar.

J'ai aussi fait en sorte que la connexion ne soit possible que si l'utilisateur a confirmé son compte, afin d'éviter les robots.

Ce contrôle devait se faire dans le projet GaeaUser, gérant la connexion à l'échelle de tous les utilisateurs Gaea21. Une condition vérifiant l'attribut "enabled" permet donc de

restreindre la connexion, et si elle n'est pas vérifiée, un message d'erreur est renvoyé et affiché sur la page.

Lors d'une mise à jour faisant passer Symfony de 5.2 à 5.3, une exception s'affichait à la connexion, car la table User n'a pas d'attribut username. En effet, le username se trouve dans la table **gaea_user**, qui est générale pour tous les comptes des sites de Gaea21. Or il est obligatoire à partir de la version 5.3. J'ai donc résolu ce bug simplement en ajoutant l'attribut, même s'il restait vide.

E.4 Uploader une image

Cette fonctionnalité était utilisée à la fois pour l'inscription, afin d'ajouter un logo pour son entreprise, et l'ajout ou la modification d'un produit/service, pour ajouter une photo du produit/service. La plupart des étapes étaient communes aux 2 situations.

Dès qu'un nouveau fichier est chargé dans l'input, une vérification du fichier est effectuée :

- sur l'extension : seuls les fichiers image sont autorisés
- sur le poids : le fichier ne doit pas dépasser 8Mo

Si le fichier correspond aux critères, il est envoyé au contrôleur. Là, un **service** commun va être utilisé : FileUploader. Ce service comporte une fonction upload qui va renommer le fichier et l'enregistrer dans un dossier spécifique. Mais surtout, l'image va être redimensionnée, pour que toutes les images respectent une taille maximale. Le plus grand côté (largeur ou hauteur) est redimensionné à 350px et l'autre est redimensionné proportionnellement. Une nouvelle image est ensuite recréée aux bonnes dimensions à partir du fichier original.

Pour finir, le nom du fichier est sauvegardé dans la base de données pour pouvoir le retrouver plus tard.

Au niveau des différences entre inscription et ajout/modification de produits, elles proviennent notamment du fait que l'inscription se fait entièrement en js, tandis que le second se fait via un formulaire Symfony.

A l'inscription



FIGURE 23 – Upload d'une image à l'inscription

Comme les données devaient être envoyées par l'intermédiaire de Javascript, il fallait trouver un moyen d'envoyer le fichier au serveur de manière à ce qu'il puisse toujours être lu en tant qu'image. A l'origine, toutes les données de ce formulaire étaient envoyées sous forme de tableau JSON "stringifié". Mais pour ajouter le fichier, il a fallu envelopper le tout d'un **FormData**, qui permet d'encoder un fichier.

A l'ajout/modification d'un produit/service

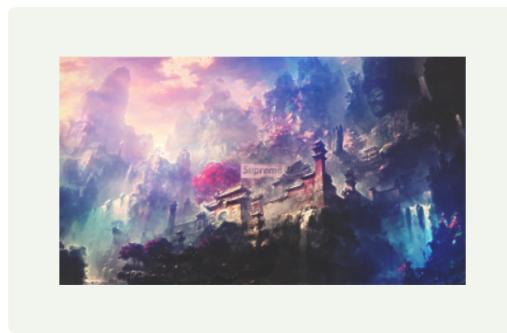


FIGURE 24 – Upload d'une image à l'ajout d'un produit

Sur ces formulaires, le champ image comporte une preview. Pour cela, on crée de nouvelles instances des classes **FileReader** et **Image** et on les associe à notre fichier. Une fois l'image lue, on l'affiche dans la preview.

Il s'agit là de formulaires Symfony, générés automatiquement à partir de l'entité Produit et Service sous forme de fichiers php. Chaque champ, ou **form_widget** est intégré dans la vue (fichier twig) comme une fonction appelant cette partie du formulaire PHP :

```
<div class="erreur">
    {{ form_errors(form.image) }}
</div>
{{ form_widget(form.image, {'attr': {'class': 'fileUpload'}}) }}
```

On ne peut à priori donc pas personnaliser les champs. Sauf avec un template. C'est donc ce que j'ai utilisé pour que le bouton d'upload soit remplacé par une image. Il suffit de placer le contenu de notre nouveau template dans un bloc respectant une nomenclature spécifique pour que le widget original soit bien remplacé.

Pour finir, j'ai ajouté des contraintes au champ pour que, malgré les avertissements, l'utilisateur ne puisse pas mettre de fichiers non autorisés.

Dans le cas d'une modification de produit ou service, même si l'attribut image contient seulement son nom, on doit envoyer au formulaire le fichier en tant que tel, comme le champ est de type Fichier. On procède donc à la conversion avant le rendu du formulaire.

A la validation, même si l'utilisateur ne modifie pas l'image, on doit manuellement réassigner l'attribut à l'ancienne image, car le formulaire Symfony la met automatiquement à null si aucun fichier n'a été réuploadé.

E.5 Page produits

Un autre groupe de fonctionnalités à développer était la visualisation et la gestion des ses produits et services.

Généralités

Cette page se décomposait en plusieurs parties, à commencer par le récapitulatif des informations de l'entreprise, et de ses catégories et sous-catégories.

Logo entreprise


Carte


Haute-savoie Genève Lausanne

Voir la carte sur google map

Description de l'entreprise :

▶ Lorem ipsum Paucorum Simonides beate vitiis nati levitate patriam ac splendor alia sed laeditur sunt levitate perfecta. Paucorum Simonides beate vitiis nati levitate patriam ac splendor alia sed laeditur sunt levitate perfecta.

Vision du développement Durable

Comment résumeriez-vous votre philosophie d'entreprise au sujet du

▶ Lorem ipsum Paucorum Simonides beate vitiis nati levitate patriam ac splendor alia sed laeditur sunt levitate perfecta. Paucorum Simonides beate vitiis nati levitate patriam ac splendor alia sed laeditur sunt levitate perfecta.

Secteurs d'activités	Sous-secteurs
Alimentation & boissons	Apiculture Horticulture
Santé & bien-être	Herboristerie
Jardins & plantes	Permaculture Agriculture d'intérieur

Raison sociale
Association

Niveau : N.2

Début de l'activité
Mars 2012

Recommendation gaea21
Non

Certification(s)
ISO14800

FIGURE 25 – Design de la 1ère partie de la page Produits/Services

Ce design avait déjà été intégré, j'ai simplement affiché les données récupérées de la base de données dynamiquement, grâce au contrôleur de la page. J'ai aussi intégré la carte comme sur le profil.

Pour le tableau des secteurs, j'ai d'abord récupéré les catégories de l'entreprise directement dans le fichier twig.

Pour les sous-catégories de l'entreprise, je les récupère via le Repository de la classe Subcategory, grâce à une requête Doctrine, qui se transforme automatiquement en requête SQL à l'exécution.

Concrètement, cette requête joint la table Subcategory à Product, Company et Category, en ne gardant que l'entreprise qui nous intéresse.

Une fois la fonction appelée dans le Controleur et la liste envoyée à la vue, je n'avais plus qu'à vérifier dans une double boucle que chaque sous-catégorie appartenait à la catégorie courante, pour l'afficher dans la bonne ligne du tableau.

Supprimer - Ajouter - modifier un produit ou service

La 2ème partie de la page concerne la visualisation et gestion des produits.



FIGURE 26 – Affichage d'un produit sur le site

Quelqu'un d'autre s'est chargé de cet affichage, mais j'ai tout de même réglé quelques bugs de css.

J'ai aussi ajouté un lien sur le nom du produit vers la fiche détaillée du produit, sur laquelle j'ai fait des modifications mineures également.

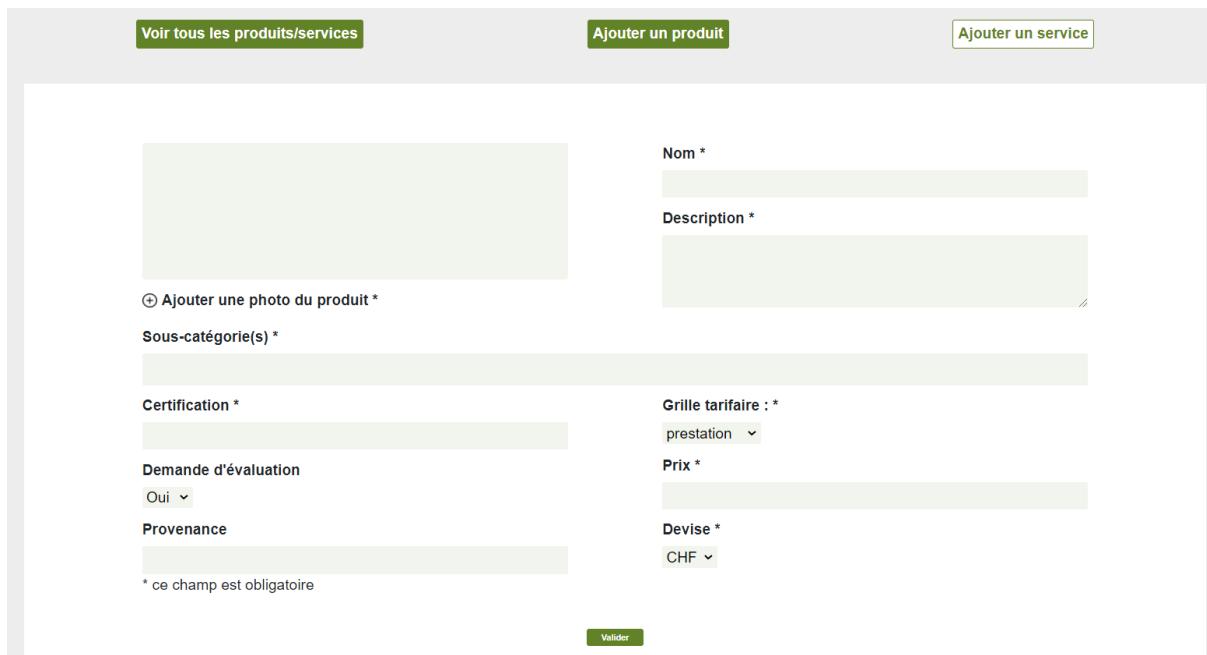
Supprimer

La suppression se fait tout simplement en passant l'id du produit à supprimer, et en appelant les fonctions remove(\$element) et flush() (enregistrement de la modification) du DoctrineManager.

Ajouter et modifier

Ces 2 fonctionnalités étaient très liées car elles reposaient sur le même modèle de formulaire, généré en php automatiquement.

Lorsque j'ai repris cette mission, les vues étaient déjà créées avec leur formulaire et le contrôleur, mais il manquait la mise en forme et notamment le champ "Sous-catégorie(s)".



The screenshot shows a web-based form for adding a service. At the top, there are three buttons: "Voir tous les produits/services" (View all products/services), "Ajouter un produit" (Add a product), and "Ajouter un service" (Add a service). The main form area contains the following fields:

- Photo du produit:** A placeholder image with a "Ajouter une photo" button.
- Nom ***: A required text input field.
- Description ***: A required text input field.
- Sous-catégorie(s) ***: A dropdown menu showing "Art et culture" selected, with other options like "Agriculture", "Chômage et Emploi", and "Cohésion sociale".
- Certification ***: A dropdown menu showing "Oui" selected.
- Grille tarifaire :** A dropdown menu showing "prestation" selected.
- Demande d'évaluation**: A dropdown menu showing "Oui" selected.
- Prix ***: A required text input field.
- Provenance**: A dropdown menu showing "CHF" selected.
- Devise ***: A dropdown menu showing "CHF" selected.
- Note:** A note stating "* ce champ est obligatoire" (This field is mandatory).
- Valider**: A green "Validate" button at the bottom right.

FIGURE 27 – Résultat du formulaire d'ajout d'un service mis en forme

Ces formulaires sont accessibles sur la page Produits par un bouton que j'ai ajouté.

Les 3 boutons que l'on peut voir en haut sont encapsulés dans un composant pour plus de modulation. Ils permettent de naviguer plus facilement d'un formulaire à l'autre et de pouvoir retourner à la page Produits.

Comme leur affichage change en fonction de la page sur laquelle on se trouve, je lui passe des variables lors de ses appels, que je définis à "selected" ou "unselected", grâce au mot-clé **with**.

Le menu déroulant à choix multiples devait satisfaire un design assez spécifique, avec des icônes intégrées au menu.



FIGURE 28 – Résultat du champ sous-catégories

Pour cela, j'ai donc dû utiliser la librairie jQuery Select2, permettant de personnaliser des champs de sélection.

J'ai d'abord ajouté un champ à choix multiple de type Entité au formulaire php. Pour que seules les sous-catégories appartenant aux catégories de l'entreprise soient disponibles, j'ai fait une requête doctrine intégrée au champ grâce à l'option 'query_builder'. On passe

en option l'entreprise connectée lors de la création du formulaire, pour l'utiliser dans cette requête.

La librairie Select2 en js permet ensuite d'afficher un template spécifique pour chaque option du menu. Pour cela, une boucle invisible est exécutée et un état spécifique (state) est attribué au menu à chaque itération (donc à chaque option). A l'intérieur, on doit boucler à travers un tableau associatif des catégories/sous-catégories afin de pouvoir récupérer les bonnes icônes, et on affiche ce nouveau template composé d'une image et d'un titre.

Pour que la nouvelle combinaison de sous-catégories soit bien ajoutée au produit, il ne fallait pas oublier d'ajouter le produit à chaque sous-catégorie. En effet, Subcategory et Product sont 2 entités liées. A la validation du formulaire tous les champs remplis de l'entité Product sont automatiquement mis à jour dans la base de données. Or il faut mettre à jour les sous catégories à la main dans l'autre entité liée (Subcategory), sinon la modification n'aura pas lieu.

Pour finir, j'ai ajouté le code permettant d'associer le bon niveau de produit selon la réponse au champ "Demande d'évaluation".

Carousel filtrant les produits

La dernière fonctionnalité de cette page était un carousel des catégories de l'entreprise permettant de filtrer les produits selon leur(s) catégorie(s). Elle ne m'était pas assignée, mais j'ai beaucoup aidé mon collègue à l'implémenter. En effet, c'est moi qui ai réfléchi à la manière dont il fallait le faire.

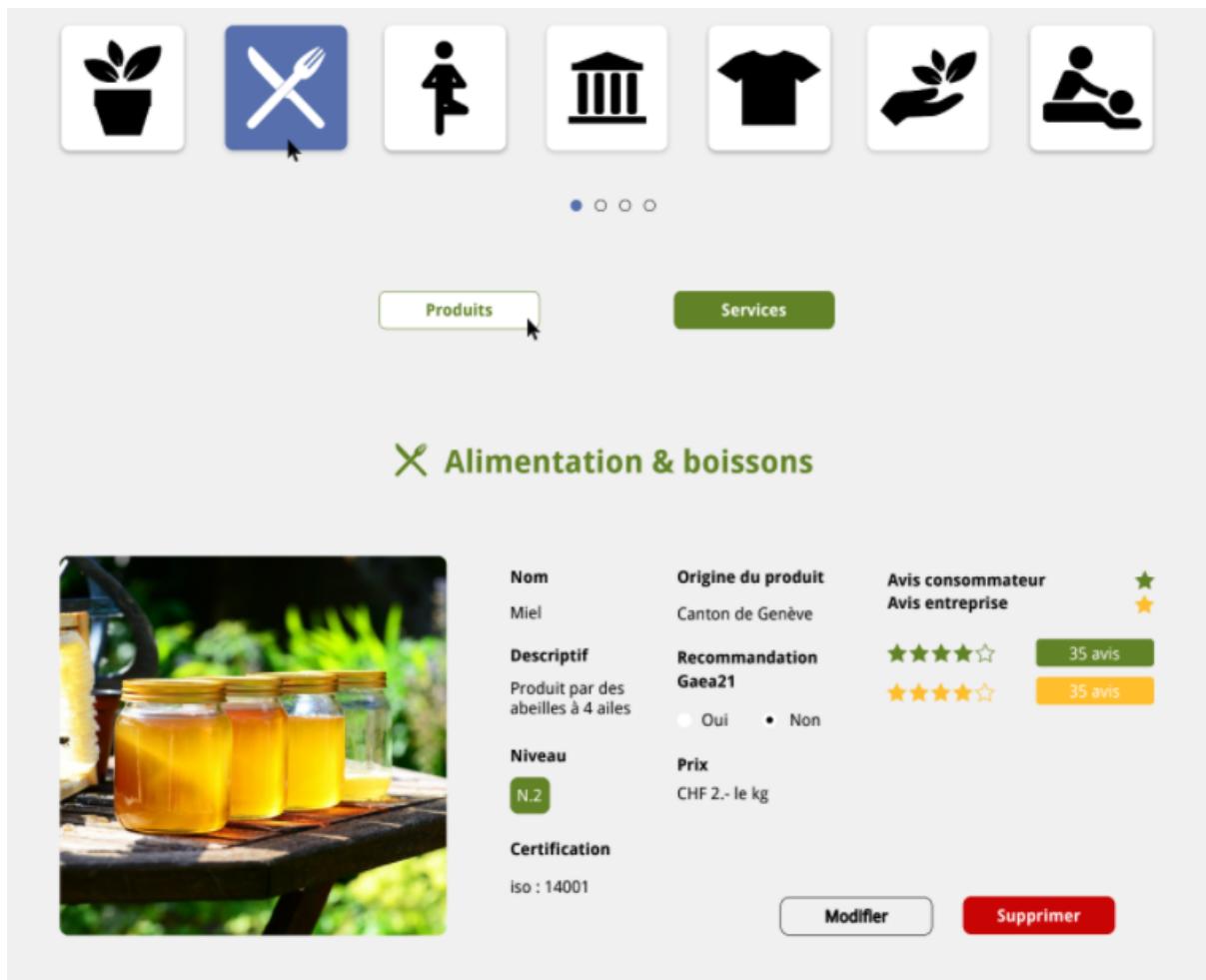


FIGURE 29 – Design du carrousel avec l'affichage des produits

Pour faire le carrousel, j'ai utilisé la librairie Flickity.js. Je l'avais complètement intégrée en twig, mais je me suis rendu compte après que pour filtrer les éléments sans recharger la page, il serait plus judicieux de tout coder en React. Il fallait donc adapter le code du carrousel pour qu'il soit fonctionnel avec React.

Pour cela, nous avons utilisé le package react-flickity-component. Il suffit alors d'appeler le composant `<Flickity/>`, puis de boucler à l'intérieur parmi les catégories de l'entreprise récupérées par requête Ajax dans l'état du composant.

La fonctionnalité est composée de 2 composants : l'un pour le carrousel, et l'autre pour la liste de produits.

Au clic sur une catégorie, on l'ajoute à l'état du composant dans la liste des catégories actives, mais si on reclique sur une catégorie déjà active, on l'enlève du tableau, donc on la désactive. On envoie cet état dès qu'il y a un changement à l'autre composant, pour qu'un filtrage puisse être fait de son côté.

Si aucune catégorie n'est sélectionnée, le filtre est inactif et on affiche donc tous les produits de toutes les catégories.

Pour récupérer les produits selon leur catégorie, on a une triple boucle : Pour chaque

produit, on parcourt chaque sous-catégorie, et pour chaque sous-catégorie, on parcourt chaque catégorie (car une sous-catégorie peut appartenir à plusieurs catégories). On doit alors remplir un tableau d'objets JSON contenant chacun une clé pour le nom de la catégorie, et ayant comme valeur le tableau de produits correspondants. Ce tableau d'objets se construisait à la volée.

Si au moins une catégorie est activée, on filtre les produits. Dans ce cas, un fonction très similaire à la précédente est appelée. La seule différence est qu'elle n'ajoute le produit au tableau que si le produit possède l'une des catégories activées.

Pour finir, on affiche les titres et leurs produits selon le tableau d'objets retourné.

E.6 Profil et Modification du profil

Outre la carte de l'entreprise, le profil devait aussi présenter les catégories et sous-catégories correspondantes de l'entreprise.

J'ai donc utilisé la même méthode que dans le tableau sur la page Produits : voir partie E.5, paragraphe Généralités. En voici le résultat :

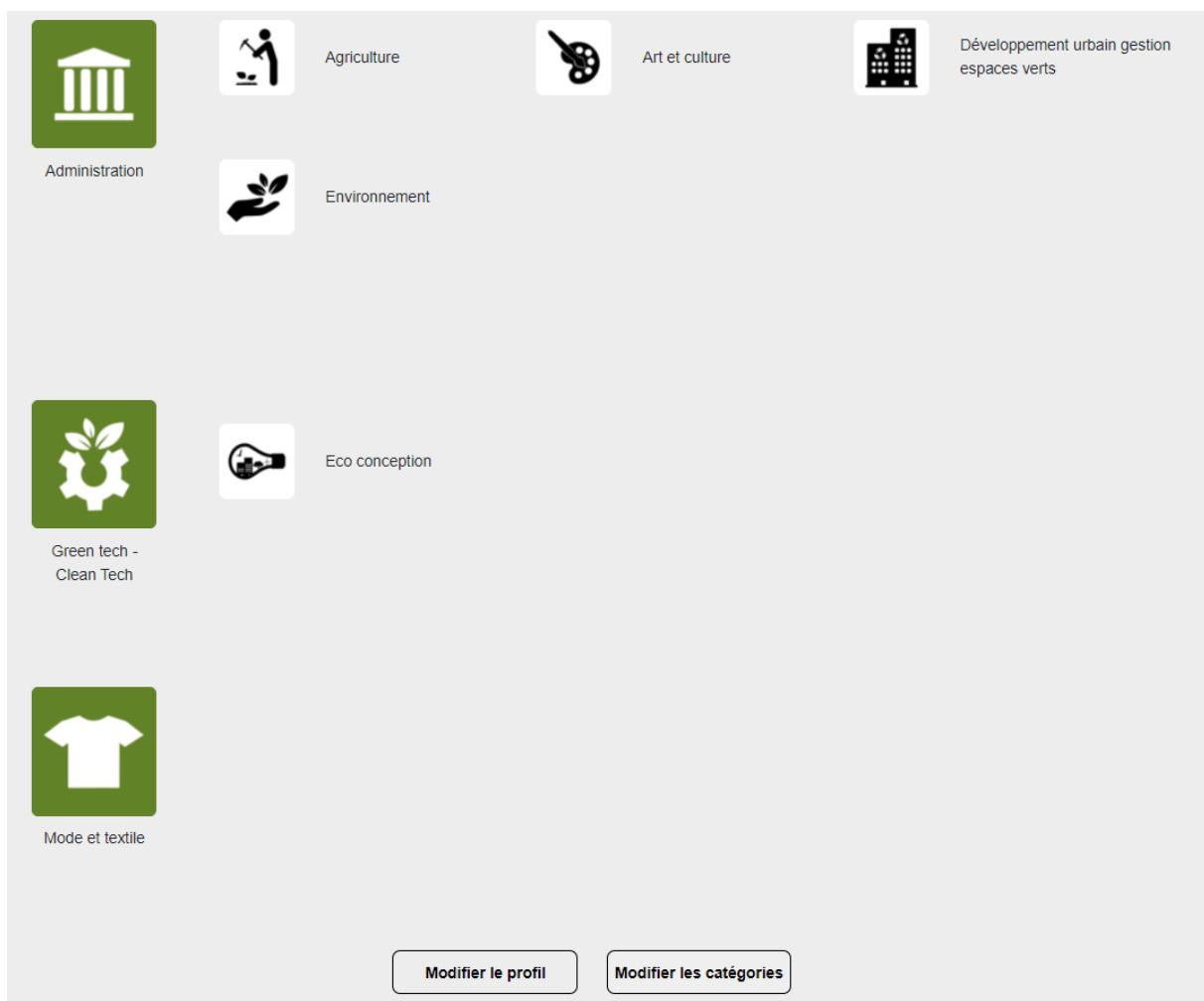


FIGURE 30 – Résultat du profil de l'entreprise

Le bouton en bas "Modifier les catégories", ouvre la pop-up des catégories à cocher, comme à l'inscription : Voir partie E.3, figure 17. Une fois le formulaire validé, le même processus de récupération et d'enregistrement a lieu (Voir partie E.3, figure 18), sauf que les catégories précédentes sont supprimées avant d'ajouter la nouvelle combinaison. Une fois la requête effectuée, la page est rechargée pour montrer un état du profil toujours à jour.

Pour finir, je devais rendre la modification de profil entièrement fonctionnelle. Le formulaire était déjà intégré et tous les champs basiques fonctionnaient.

Je modifie mes données

Nom de l'entreprise	Canton, département	Je souhaite être évalué par gaea21 :
Lalalande	duo	En savoir plus > <input type="radio"/> Oui <input checked="" type="radio"/> Non
Raison sociale	Pays	Description de votre entreprise :
cdfd	France	dfs vdbfdfffffff
Adresse	Téléphone	Votre vision du Développement Durable :
9 rue Lac Saint-André	sfgtfdf	Comment résumeriez-vous votre philosophie d'entreprise au sujet du Développement Durable ?
Ville	Site web	gdso
Le Bourget-du-Lac	https://lalaland.com	
Code postal	Lien Linkedin	
32767	https://lalaland.com	
	Lien Facebook	
	https://lalaland.com	
	Lien Twitter	
	https://lalaland.com	
	Début de l'activité	Téléchargez votre logo (jpeg/png)
	01 <input type="button" value="▼"/> Jan <input type="button" value="▼"/> 1800 <input type="button" value="▼"/>	<input type="button" value="Parcourir"/> 61b5ff417d165.jpg
	Certification(s)	<input type="button" value="Supprimer l'image"/>
	vdsjioji	
	Zone d'influence géographique *	
	50km <input type="button" value="▼"/>	* ce champ est obligatoire
<input type="button" value="Modifier"/>		

FIGURE 31 – Résultat du formulaire de modification du profil

J'ai donc, comme pour l'inscription, rendu fonctionnels la demande d'évaluation, l'upload d'image, et la vérification de l'adresse avec des messages d'erreur.

J'ai repris exactement le même code qu'à l'inscription (E.3), y compris pour l'upload d'image (E.4).

E.7 Activation manuelle des comptes

En attendant que le mail d'activation de compte soit fonctionnel sur le serveur, il nous a été demandé créer une page permettant d'activer chaque compte manuellement. Comme pour le carousel filtrant, j'ai aidé un collègue à réaliser cette tâche.

Liste administrateur des comptes utilisateurs inactifs		
Id	Prénom	Email
47737	Le Thièc ASD	alexandre@lethiec.ch
		Activer le compte
47738	Les Art'Isanes	info@art-tisanes.ch
		Activer le compte
47739	Les potions d'adèle	lespotionsdadele@gmail.com
		Activer le compte
47740	Les vélos du marché	thierry.briquet@lvdm.ch
		Activer le compte

FIGURE 32 – Résultat de la page d'activation manuelle des comptes

La page se présente sous forme de liste, avec un bouton **Activer**, confirmant le compte au niveau de Gaea21.

Dans un premier temps, dans le projet **GaeaUser** nous avons récupéré tous les utilisateurs non activés dans le contrôleur, en appelant le Repository de la classe GaeaUser. Dans ce dernier, on fait une requête Doctrine. Une fois les données récupérées dans le contrôleur, on les sérialise sous forme d'objet JSON, que l'on retourne.

Du côté du Répertoire vert, on peut alors récupérer ces données via une **requête HTTPS**, que l'on va ensuite décoder. On doit après filtrer les utilisateurs Gaea pour ne garder que ceux inscrits sur le Répertoire Vert. On récupère donc les **GaeaUserIds** de tous les utilisateurs Répertoire vert, on les compare 1 à 1 avec les utilisateurs Gaea dans une double boucle, et on stocke dans un tableau les utilisateurs dont l'id est commun.

Ensuite, on peut afficher la liste dans la vue, avec pour chaque ligne un bouton appelant une fonction **enableAccount**. Grâce au token unique associé à chaque ligne que l'on envoie au javascript par un **dataset**, on peut faire une requête Ajax confirmant le compte associé à ce token.

E.8 Page sous-catégorie

Les pages sous-catégories permettent de visualiser la liste des entreprises appartenant à une sous-catégorie. On y accède par la page d'accueil, en choisissant une catégorie, et sur la page catégorie, on peut ensuite choisir une sous-catégorie.

Cette vue était déjà intégrée et un système de filtre avait été implémenté par un collègue, mais je devais revoir la récupération des entreprises qui n'était pas fonctionnelle, corriger certains bugs de style, mais aussi implémenter un nouveau design (ci-dessous) :

			Bureautique		Accessoire de bureau	
			Filtre			
Nom entreprise	Origine du produit	Date d'inscription au Répertoire Vert	Niveau	N.2	Voir la fiche produit	Voir la fiche entreprise
Ikea	Canton de Genève	13/05/2021				
Ikea	Canton de Genève	13/05/2021				
Ikea	Canton de Genève	13/05/2021				

FIGURE 33 – Design de la page sous-catégorie

Il doit présenter les entreprises sous forme de liste, alors que la version actuelle les présente de manière plus détaillée. Il fallait aussi ajouter une fonctionnalité de tri, par ordre alphabétique ou par proximité. Et enfin, un système de pagination devait être mis en place, permettant de répartir l'affichage des entreprises sur plusieurs pages et de naviguer entre elles.

J'ai récupéré les bonnes entreprises via le CompanyRepository selon la catégorie et la sous-catégorie, en réutilisant la fonction utilisée pour le SideMenu de la carte de toutes les entreprises (voir partie E.2).

Comme on a besoin des ID catégorie et sous-catégorie, j'ai ajouté ces 2 paramètres à l'URL pour pouvoir les récupérer au fil de la navigation.

Une fois cette requête (et non le résultat de la requête) récupérée dans le contrôleur, il faut l'ajouter à la pagination.

La pagination fonctionne grâce au bundle **KnpPaginatorBundle** de KnpLabs

```
$companies = $paginator->paginate(
    $companiesQuery,
    $request->query->getInt('page', default: 1),
    10);
```

FIGURE 34 – Code pour la pagination

Les 3 paramètres correspondent respectivement à la requête que l'on veut paginer (ici les entreprises), le numero de la page à afficher (ici on récupère dans l'url la valeur de

l'option "page", et par défaut on affiche la 1ère page), et le nombre d'éléments à afficher par page.

Pour finir, j'ai refait le css de la rubrique "Consultez aussi", dont les titres ne s'affichaient pas conformément au design et dont les images étaient déformées.

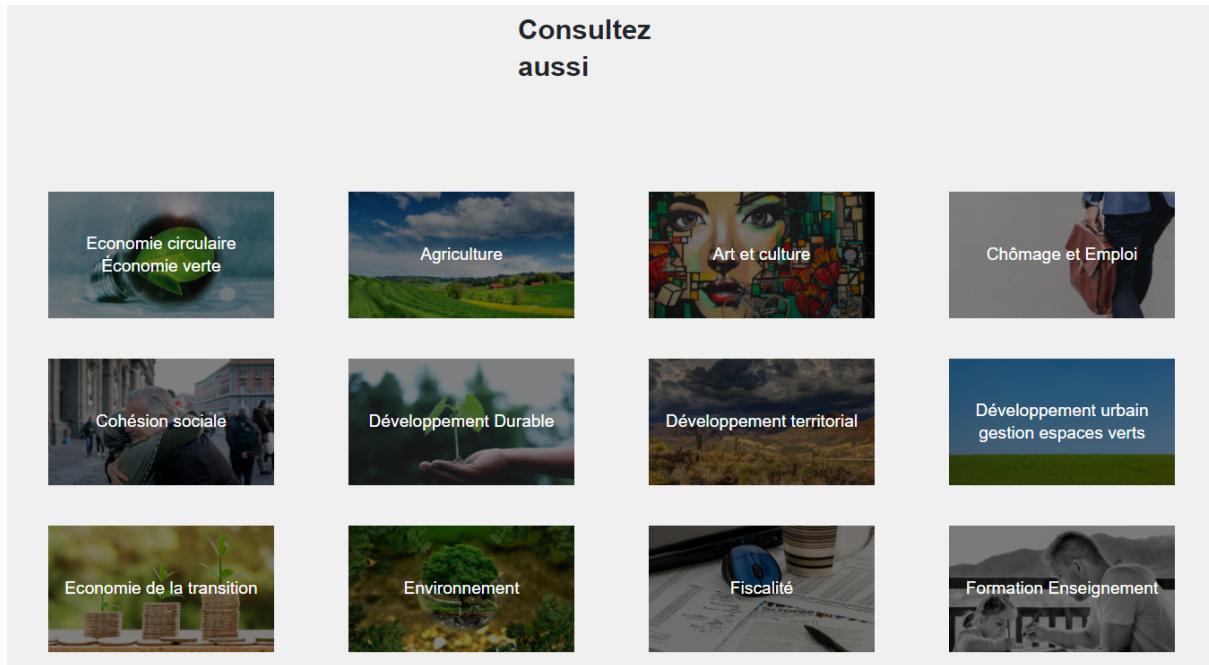


FIGURE 35 – Résultat de la rubrique Consultez aussi

Je n'ai pas eu le temps de m'occuper de l'affichage sous forme de liste, ni des différents tris, le nouveau design ayant été livré peu avant la fin de mon stage.

E.9 Ajout des entreprises manquantes

Comme expliqué dans les objectifs, 62 entreprises étaient déjà inscrites chez Gaea21, mais leurs comptes avaient été supprimés.

Ma mission était d'écrire un script pour lire un fichier Excel contenant leurs informations et ajouter les entreprises à la base de données. J'utilise le bundle **phpspreadsheet** de phpooffice pour lire le fichier excel, et convertir toutes les informations en un tableau de tableaux.

Après cela, on boucle dans ce tableau : chaque information de l'entreprise est assignée à une variable, et j'ai repris là le code utilisé à l'inscription : J'ai une requête HTTPS POST qui ajoute un nouveau GaeaUser, avec un mot de passe généré aléatoirement, puis je crée une nouvelle instance de Company et j'assigne tous les attributs à des valeurs par défaut, sauf celles renseignées dans le fichier Excel.

E.10 Réinitialisation de mot de passe

Les entreprises ajoutées par l'Excel devaient recevoir un mail leur permettant de réinitialiser leur mot de passe.

La fonctionnalité de Réinitialisation de mot de passe a donc été faite pour ce motif uniquement, il n'y a pas bouton "mot de passe oublié" disponible à la connexion.

Etape 1 - Préparer le code d'envoi du mail de réinitialisation :

Dans la même fonction que celle qui ajoute chaque utilisateur, on fait une requête vers la fonction **requestPassword** de l'Api GaeaUser, et on lui passe l'email de la ligne courante et l'url du site vers lequel le lien du mail doit renvoyer.

Cette fonction va récupérer le bon utilisateur, avec son token unique, qu'elle pourra introduire dans le lien du mail.

Etape 2 - Créeer la page de réinitialisation et son contrôleur :

Dans le contrôleur gérant les Users, j'ai créé une fonction affichant une vue, en lui passant le token via l'url de la page. Puis j'ai créé la la vue :



FIGURE 36 – Page de réinitialisation de Mot de passe

Etape 3 - Faire fonctionner la réinitialisation en JavaScript :

Dans un fichier js, j'ai tout d'abord mis en place la vérification du format du nouveau mot de passe avec les messages d'erreurs (nombre de caractères, chiffres, etc...), que j'ai reprise du code de l'inscriptiton.

Lorsqu'on clique sur valider, le contenu des 2 champs de mot de passe sont **encodés en md5**, puis ils sont envoyés à la fonction qui réinitialise le mot de passe dans l'API, toujours par requête Ajax.

Les messages d'erreurs sont gérés également dans l'API, et sont affichés sur la page si la réponse renvoie un code d'erreur. Sinon, un message de succès apparaît.

E.11 Rendre les graphes de statistiques du Dashboard fonctionnels

Sur le Dahboard de l'utilisateur, celui-ci peut voir ses statistiques sur des graphiques. Ils avaient déjà été implémentés à mon arrivée.

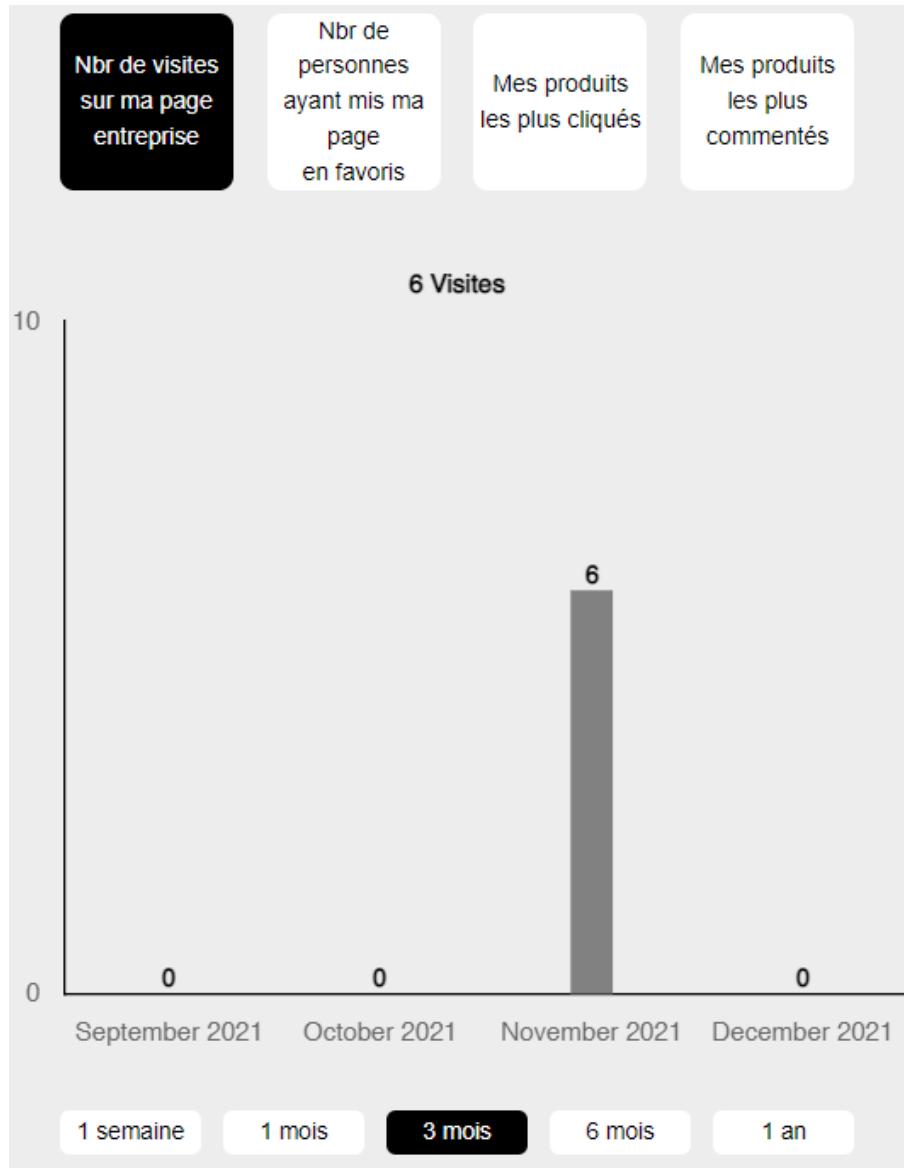


FIGURE 37 – Graphe du nombre de visites sur la page entreprise pour les 3 derniers mois

Mais je me suis rendu compte de certains bugs, que j'ai corrigés.

1er bug :

Il concernait l'affichage du nombre de visites au mauvais jour. En effet, le graphe présentant les statistiques sur les 7 derniers jours ne comptait pas le jour courant. Donc si une visite avait eu lieu le jour courant, elle serait affichée au 1er jour du graphique. J'ai donc

simplement fait en sorte que le jour courant fasse partie des dates affichées sur le graphique.

2nd bug :

Lorsqu'on affichait les statistiques pour les 3, 6 ou 12 derniers mois et que l'on se trouvait un 29, 30 ou 31, certains mois ne s'affichaient pas, car ces jours n'existent pas, notamment pour le mois de février. Il fallait donc soustraire 3 jours à la date courante si le jour où on se trouvait excédait le 29 du mois. Ainsi, on pouvait ensuite soustraire le nombre de mois que nous souhaitions à la date courante, sans qu'il en manque.

E.12 Désactivation du compte Répertoire Vert

Cette fonctionnalité se trouve dans les paramètres du profil et a pour but de désactiver le compte du côté du Répertoire Vert (et non de GaeaUser), et était développée par un collègue. Mais je suis intervenue pour ajouter et terminer certains détails oubliés.

J'ai tout d'abord fait le CSS du formulaire, qui rend comme ceci sur le site :



Désactivation du profil

Adresse mail *

Mot de passe *

Confirmer mot de passe *

Valider

* ce champ est obligatoire

FIGURE 38 – Rendu du formulaire de désactivation de compte

Ensuite, j'ai ajouté une Pop-up de demande de confirmation lors de la validation :

```
{{form_widget(desactivateForm.submit, {'attr': 
    {'onclick': 'return confirm("En validant, vous recevrez un code de réactivation sur votre boîte mail. 
    Conservez ce code, il permet de réactiver votre compte à tout moment. Êtes-vous sûr(e) de vouloir désactiver votre compte ?")'
    , 'class': 'label-btn-send-disable'}})}}
```

FIGURE 39 – Code pour la boîte de dialogue de confirmation

Comme il s'agit d'un formulaire Symfony, on appelle chaque composant/widget du formulaire comme ci-dessus, avec la possibilité d'ajouter des attributs html, comme une fonction **onClick**.

Ensuite, je me suis occupée du mail de réactivation du compte. Le code d'envoi du mail à la désactivation du compte, ainsi que la réactivation du compte, avaient déjà été préparés. J'ai juste modifié l'url d'activation dans l'email, et décommenté l'envoi.

Une fois le compte réactivé, l'utilisateur est redirigé vers la page de succès d'activation du compte, comme sur la figure 22.

Enfin, je me suis assurée que les entreprises désactivées n'apparaissaient plus sur le site, notamment lorsqu'on affiche la liste des entreprises d'une sous-catégorie spécifique. J'ai fait cela en ajoutant une clause **where** dans la requête Doctrine.

E.13 Responsive

Pré-homepage
Homepage entreprise
Menu
(refs React) Dashboard
Inscription

E.14 Plannings Poker

+ résolution bugs
Communication avec autres départements

F Résultats par rapport à l'objectif et planning réel

Parler du retard V0

De plus, ce projet était marqué par un fort turnover. Il fallait donc un temps d'adaptation et de familiarisation pour chaque nouvelle recrue, notamment pour l'installation du projet, savoir quel fichier correspondait à quoi, etc... assigner des tâches qui correspondent aux objectifs de formation de chacun Compétences acquises

Page sous catégorie (liste) ?

Conclusion

Bibliographie

- [1] ARISTOTE. *La Politique*. 300 AEC.
- [2] Paul Adrien Maurice DIRAC. *The Principles of Quantum Mechanics*. International series of monographs on physics. Clarendon Press, 1981. ISBN : 9780198520115.
- [3] Pauline GUIRAGOSSIAN. « Former le citoyen-soldat sous la République jacobine ». In : *L'éducation des citoyens, l'éducation des gouvernants*. Aix-en-Provence, France, sept. 2019. URL : <https://hal-amu.archives-ouvertes.fr/hal-02115427>.
- [4] Tancrede RAMONET. *Ni Dieu ni maître, une histoire de l'anarchisme*. 1 :20 :48 ([short.url](#)) - Editorial Moscou. ARTE. 2019.
- [5] Pablo SERVIGNE et Gauthier CHAPELLE. *L'entraide, l'autre loi de la jungle*. Les Liens qui Libèrent, 2019.
- [6] James W. KUROSE et Keith W. Ross. *Computer Networking : A Top-Down Approach*. 8^e éd. url : ([short.url](#)). Pearson, 2021.
- [7] WIKIPÉDIA. *Portail de Cryptologie*. [En ligne ; page disponible]. URL : <https://fr.wikipedia.org/wiki/Portail:Cryptologie>.

Table des figures

1	Logo du Répertoire Vert	4
2	Page d'accueil entreprise	5
3	Design de la page carte à implémenter	6
4	Planning Poker	7
5	Extrait de la base de données du site Répertoire Vert	8
6	Exemple d'entête de page intégrée	9
7	Planning prévisionnel	13
8	Planning prévisionnel de septembre	14
9	Accueil de l'appli	16
10	Modification d'un utilisateur	16
11	Page de l'utilisateur avec ses possessions	17
12	Résultat de la page profil avec la carte	18
13	Design page carte	19
14	Affichages successifs du side menu	20
15	Code de récupération de la localisation de l'utilisateur	21
16	Code d'envoi des propriétés nécessaires au filtrage au SideMenu	22
17	Pop-up de choix des catégories	23
18	Code d'enregistrement des catégories	23
19	Champ de demande d'évaluation	24
20	Champ de demande d'évaluation	24
21	Page Tarifs/Niveaux	25
22	Page de succès de confirmation de compte	26
23	Upload d'une image à l'inscription	27
24	Upload d'une image à l'ajout d'un produit	28
25	Design de la 1ère partie de la page Produits/Services	29
26	Affichage d'un produit sur le site	30
27	Résultat du formulaire d'ajout d'un service mis en forme	31
28	Résultat du champ sous-catégories	31
29	Design du carousel avec l'affichage des produits	33
30	Résultat du profil de l'entreprise	34
31	Résultat du formulaire de modification du profil	35
32	Résultat de la page d'activation manuelle des comptes	36
33	Design de la page sous-catégorie	37
34	Code pour la pagination	37
35	Résultat de la rubrique Consultez aussi	38
36	Page de réinitialisation de Mot de passe	39
37	Graphe du nombre de visites sur la page entreprise pour les 3 derniers mois	40
38	Rendu du formulaire de désactivation de compte	41
39	Code pour la boîte de dialogue de confirmation	41

Liste des tableaux

Annexes

Table des matières

A	Documents d'organisation et de planification	I
B	Le site web	I

A Documents d'organisation et de planification

B Le site web

