

HAUTE ECOLE DE LA COMMUNAUTE FRANCAISE EN HAINAUT

Département Sciences et Technologies
8A Avenue Victor Maistriau – 7000 Mons

Développement d'une application web de type wiki : « Naturopath »

Travail de fin d'études réalisé en vue de l'obtention du titre de bachelier en Informatique et systèmes, orientation réseaux et télécommunications

Promoteur : Depreter Johan

Étudiant(s) : Hachard Victor
Vander Goten Maxime

HAUTE ECOLE DE LA COMMUNAUTE FRANCAISE EN HAINAUT

Département Sciences et Technologies
8A Avenue Victor Maistriau – 7000 Mons

Développement d'une application web de type wiki : « Naturopath »

Travail de fin d'études réalisé en vue de l'obtention du titre de bachelier en Informatique et systèmes, orientation réseaux et télécommunications

Promoteur : Depreter Johan

Étudiant(s) : Hachard Victor
Vander Goten Maxime

Dans un premier temps, nous voudrions adresser nos remerciements à notre promoteur M. Johan Depreter pour sa disponibilité ainsi que pour ses conseils fructueux.

Nous souhaitons également témoigner toute notre reconnaissance au corps enseignant de la HEH pour nous avoir fourni tous les outils nécessaires à la bonne réalisation de notre projet.

Table des matières

Table des matières	5
Tables des figures	8
Tables des codes.....	10
Tables des tableaux	11
Abstract	12
Introduction.....	15
Mise en contexte	16
Technologies.....	18
1. Back-end	18
2. Base de données.....	19
3. Front-end	20
4. Dépendances et logiciels	21
5. Méthodologie	22
Conception	23
1. Diagramme de contexte	23
2. Vision d'une page	24
2.1 Diagramme de création.....	25
Réalisation Back-end	26
1. Les Packages	26
1.1 Package « Controller ».....	26
1.1.1 Package « Rest »	26
1.1.2 Package « Services »	27
1.1.3 Package « Validators »	27
1.2 Package « Dto »	27
1.3 Package « Init ».....	27
1.4 Package « Mappers ».....	27
1.5 Package « Model ».....	28
1.5.1 Package « Entities »	28
1.5.2 Package « Facades »	28
1.5.3 Package « Repositories »	28
2. Dépendances	28
2.1 Lombok.....	28
2.2 Thendenda.....	28
2.3 Javax.mail.....	29
2.4 Junit	29
2.5 Jsonwebtoken.....	29
3. Le modèle métier.....	29
3.1 La classe abstraite « AbstractEntity ».....	29

3.2 La classe « User » et la classe « UserSecurity »	30
3.3 L'énumération « EnumState »	30
3.4 La classe abstraite « AbstractAutowire »	31
3.4 Les types et les catégories	31
3.5 Les pages et leurs contenus	32
3.6 Le modèle métier utilisateurs	33
3.7 Les repositories	34
3.8 L'interface « InnerRepositories »	34
3.9 Les facades	35
4. La sécurité	36
4.1 La configurations de la sécurité avec Spring Boot	36
4.2 Les filtres	36
4.3 L'authentification	37
4.4 Les validateurs	37
5. Les DTO	38
6. Les contrôleurs	38
7. Les services	39
8. Les mapeurs	40
9. Les mails	40
10. Le stockage de fichiers	41
11. La recherche	42
Réalisation du Front-end	43
1. Les services	43
2. Ergonomie	43
3. Les pages	44
3.1 Section catégories	45
3.2 Résultat de la recherche	46
3.3 Barre de navigation	46
3.4 Page de recherche	48
3.5 Pages des favoris	48
3.6 Page d'administration des pages	49
3.7 Page des tickets	50
3.8 Ajout de pages	52
3.9 ParagraphType	56
3.10 ParatagType	56
3.11 Parapage	57
3.11 Page d'ajout et d'édition des tags	58
3.11 Page d'ajout et d'édition d'images	59
3.12 Page d'ajout et d'édition des catégories	59
3.13 Page des settings	61
3.14 Page des conditions d'utilisations	64
4. Visualisation d'une page finie	64
5. Les mails	68

Points forts / faibles.....	69
1. Back-end	69
1.1 Générale	69
1.1.1 Organisation des package :.....	69
1.1.2 Génération du token :	69
1.1.3 La généralisation :.....	69
1.1.4 Les Dto :	70
1.2 Amélioration.....	70
1.2.1 La généralisation :.....	70
1.2.2 La recherche :	70
1.3 Erreur connue.....	70
2. Front-end	70
2.2 Amélioration.....	70
2.2.1 Un système d'erreur :.....	70
2.2.2 Plus de module :	70
Avancement sur le cahier des charges	71
1. Fonctionnalités faites	71
2. Fonctionnalités à faire	71
Conclusion	73
Lexique.....	75
Bibliographie.....	76
Syllabus.....	76
Source électronique	76
ANNEXES.....	77
Table des annexes	78

Tables des figures

Figure 1 : logo de Spring Boot	18
Figure 2 : logo de PostgreSQL.....	19
Figure 3 : logo de Angular.....	20
Figure 4 : logo de Postman	21
Figure 5 : logo de Swagger.....	21
Figure 6 : logo de Bootstrap	21
Figure 7 : logo de Material Angular.....	21
Figure 8 : diagramme de contexte.....	23
Figure 9 : diagramme de création	25
Figure 10 : diagramme représentatif des packages	26
Figure 11 : diagramme de la classe abstraite : AbstractEntity	30
Figure 12 : diagramme de la classe : User et UserSecurity	30
Figure 13 : diagramme de l'énumération « EnumState »	31
Figure 14 : diagramme de la classe Category et de la classe SortedType	31
Figure 15 : diagramme de la classe page et de leurs contenus.....	32
Figure 16 : diagramme du modèle métier utilisateur	33
Figure 17 : diagramme des repositories.....	34
Figure 18 : diagramme de l'interface « InnerRepositories »	34
Figure 19 : diagramme des façades	35
Figure 20 : diagramme de la classe sécurité.....	36
Figure 21 : diagramme des filtres	36
Figure 22 : diagramme des contrôleurs.....	38
Figure 23 : diagramme des services	39
Figure 24 : diagramme des mapeurs	40
Figure 25 : diagramme des mails.....	41
Figure 26 : visuel de la page home	44
Figure 27 : visuel des sections	45
Figure 28 : visuel de la recherche	46
Figure 29 : visuel de la barre de navigation.....	46
Figure 30 : visuel de la liste recherche	47
Figure 31 : visuel de la liste déroulante bis	47
Figure 32 : visuel d'une page de recherche	48
Figure 33 : visuel de la page des favoris	49
Figure 34 : visuel de la page d'administration.....	50
Figure 35 : visuel de la page tickets.....	51
Figure 36 : visuel de l'envoi du ticket	51
Figure 37 : visuel du suivi du ticket par l'administrateur	52
Figure 38 : visuel de l'ajout d'une page.....	52
Figure 39 : visuel de l'édition d'une page.....	53
Figure 40 : visuel de l'ajout de pages	54
Figure 41 : visuel de la validation d'un innerPage	55
Figure 42 : visuel de l'historique d'une page.....	55
Figure 43 : visuel de l'édition d'un paragraph.....	56

Figure 44 : visuel de l'édition d'un paratag	56
Figure 45 : visuel de l'édition d'un parapage	57
Figure 46 : visuel de publication d'une page	57
Figure 47 : visuel de la page d'ajout d'un tag	58
Figure 48 : visuel de la page d'édition d'un tag	58
Figure 49 : visuel de la page d'ajout d'une image	59
Figure 50 : visuel de la page d'ajout d'une catégorie	59
Figure 51 : visuel de la page d'édition de catégorie	60
Figure 52 : visuel de la page de définition d'une catégorie	61
Figure 53 : visuel de la page de setting	62
Figure 54 : visuel de la page setting - sécurité	62
Figure 55 : visuel de la page setting - profile	63
Figure 56 : visuel de la page setting - apparence	63
Figure 57 : visuel de la page de conditions d'utilisations	64
Figure 58 : visuel de l'alerte et de la catégorie d'une page fini	64
Figure 59 : visuel de la messagerie d'une page	65
Figure 60 : visuel du titre/image d'une page finie	65
Figure 61 : visuel des pages recommandées	66
Figure 62 : visuel des paragraphes sur une page fini	66
Figure 63 : visuel des paratags sur une page finie	67
Figure 64 : visuel des paratag sur une page finie	68
Figure 65 : visuel d'un mail reçu par l'application	68

Tables des codes

Code 1 : méthode permettant de crée une nouvelle instance	35
Code 2 : méthode dans la classe abstraite généraliser	35
Code 3 : méthode du front-end avec un header	37
Code 4 : méthode possédant une annotation « PreAuthorize »	39
Code 5 : méthode getAll() de la class abstraite « AbstractService »	40
Code 6 : attribut et méthode permettant de récupère les repositories	40
Code 7 : instanciation de la classe « SendEmail »	41
Code 8 : algorithme simplifier de la recherche exacte	42
Code 9 : algorithme simplifier de la recherche avec fautes	42
Code 10 : exemple de décomposition d'un mot avec la méthode « Utils.get3String() »	42
Code 11 : la classe abstraite « AbstractService »	43
Code 12 : méthode utilisant un service pour récupère des données	43
Code 13 : méthode permettant de génère un nouveau token	69

Tables des tableaux

Tableau 1 : tableau des avantages et des inconvénients de Spring Boot	19
Tableau 2 : tableau des avantages et des inconvénients de PostgreSQL.....	19
Tableau 3 : tableau des avantages et des inconvénients de Angular.....	20

Abstract

De nos jours, quel que soit le domaine d'activité, l'accès à l'information est devenu primordial. Elle se doit d'être accessible facilement mais surtout rapidement, être la plus complète possible et la plus juste. Pour certaines voies professionnelles, cette importance d'apport d'informations est même une nécessité. Prenons l'exemple de l'herboristerie, travaillant avec des plantes parfois méconnues, composées de divers ingrédients et pouvant être dangereuses dans certaines circonstances. Les herboristes doivent alors composer des recettes particulières ou des remèdes médicaux. Malgré leurs connaissances, on peut vite se rendre compte, qu'une simple erreur peut être dramatique. Un herboriste doit toujours contrôler, vérifier et analyser les différents produits qu'il emploie, leurs propriétés, leurs possibilités et leurs compatibilités.

Le numérique a pris le pas, pourtant la population souhaite de plus en plus de méthodes naturelles pour les soins esthétiques ou médicaux. Les plantes sont très à la mode et recherchées. D'un simple clic, on retrouve des centaines de recettes en tout genre, garantissant toujours le meilleur. « Les petites recettes de grand-mère ». Mais peut-on toujours leur faire confiance ? comment peut-on s'y retrouver ? Sommes-nous certaines qu'elles soient sans danger ?

Pour répondre à de tels besoins, l'idée d'une application Web est la meilleure des options. Permettant ainsi la centralisation des informations sur le sujet, regroupant tant l'expertise de professionnel et le savoir d'amateur aguerri. L'idée est certes intéressante et innovante dans le domaine de l'herboristerie, reste à en définir ses possibilités, son fonctionnement, ces valeurs, mais surtout le contrôle de véracité des informations diffusées.

Trois lignes directrices ont été mises en place : vérification des données par un panel d'experts, personnalisation de l'interface en fonction du contenu et communication entre intervenants.

Le principe d'une application web apporte son lot d'avantages comme : ouverte à tout public, la facilité de recherche, la gratuité et ainsi que la possibilité de différents partenariats.

Sur papier, l'idée est là, concrète. Pourtant, en tant que développeur, le travail ne fait que commencer. Avant même la réalisation d'un code source, plusieurs étapes sont indispensables pour mener ce projet à bien. Il est impératif de se poser les bonnes questions et de trouver ainsi les meilleures options en termes de technologies, d'architectures et de méthodologies. Ainsi des recherches ont été entreprises pour y répondre.

Coté back-end, le choix c'est finalement porté sur le framework de développement Java : Spring Boot, offrant toutes les fonctionnalités et la sécurité nécessaire pour le développement de l'API.

Pour la gestion de la base de données de l'application, l'option de PostgreSQL semblait intéressante dans le cadre du développement du projet, permettant de valoriser l'intégrité de toutes les données et répondant ainsi aux besoins de protection et de sécurité nécessaire pour les utilisateurs.

Le framework Angular fut choisi pour le développement du front-end. Répondant en tout point aux besoins de réalisation pour cette partie du projet. Entre autres pour la fluidité qu'offre la création d'application on-page, pour l'emploi du HTML dans les interfaces utilisateurs, mais également pour les outils qu'il propose en termes de tests unitaires ou encore son architecture de types MVC.

D'autres dépendances ont été utilisée afin de faciliter le développement : Postman, qui a permis de tester directement l'API. Swagger pour la documentation de celle-ci. Bootstrap et Angular Material pour améliorer le style, les rendant plus attractives, cohérentes et fonctionnelles.

Enfin étant un travail collaboratif, le système de versioning Git coupler avec GitHub aura permis de se répartir correctement le travail tout en le sauvegardant à un seul et même endroit.

Une fois ces choix déterminés la réalisation pouvait commencer. Dans un premier temps, la conception a fait l'objet d'une réflexion proposant ainsi la vision souhaitée pour l'application.

Celle-ci propose donc plusieurs modèles de pages, en fonction d'une catégorie, d'une section, d'une correspondance ou du genre de contenu. Chaque type de page disponible sont au départ composé de la même manière, dans leur version la plus simpliste. Une image, un titre et une description. Lors de la création d'une nouvelle page, plusieurs actions entre en compte avant même de pouvoir la valider.

Ensuite est venu le temps de procéder à la réalisation du back-end proprement dit. Pour ce faire plusieurs packages ont été mis en place, comme le package « controllers », « dto » ou encore le package « model » ... dans la même optique, plusieurs dépendances ont été installées permettant de profiter de toutes leurs fonctionnalités additionnelles et faciliter ainsi le développement. Comme la dépendance « Lombok », « thedenda » ou « jsonwebtoken » ...

La suite de la construction du back-end, consiste en la définition des différentes classes, les décisions sur les interfaces, ainsi que la sécurité mise en place, entre autres pour l'authentification, tous les éléments dont l'application a besoin. Ceux-ci sont développés, détaillés et représentés aux chapitres correspondants.

En parallèle, le front-end a été étudié et mis en place pour correspondre aux mieux aux différents besoins de l'application. Celui-ci est divisé en deux parties distinctes permettant la communication, les services et les vues.

Les services créent et leurs méthodes permettent la communication entre les différents composants du back-end vers le front-end.

La page principale propose entre autres deux volets : Le premier (gauche) met en évidence les différentes options permettant la recherche. Celle-ci est possible par tag, par mots clés, ou simplement en sélectionnant une catégorie ou sous-catégorie. Le deuxième volet, qui représente la partie centrale de la page, mettant à disposition les visuels des choix de la recherche.

Les différentes vues permettent également l'ajout et la modification de nouveaux articles, ainsi que la validation par vote des experts avant publication.

Une sécurité a été également mise en place, tant pour l'authentification des utilisateurs, mais surtout pour la pertinence et la véracité des données ajoutées.

Une analyse a été réalisée pour comprendre aux mieux les points forts mais également les points faibles qu'apporte la vision et la réalisation de cette application. Tant côté back-end avec par exemple : l'organisation et les notions de split du code, point fort de la construction.

Enfin, un bilan a été posé, un regard critique sur la réalisation et les objectifs à atteindre. Permettant de voir ce qu'il a été possible de mettre en place et les modifications qui pourraient être ajoutées par la suite pour accroître et optimiser l'application, comme un forum, le multi-langues, statistiques, ou une référence des produits détaillés.

Introduction

« Nous voyons l'abeille se poser sur toutes les plantes et tirer de chacune le meilleur. »

Isocrate

De nos jours, quel que soit le domaine d'activité, l'accès à l'information est devenu primordiale. Elle se doit d'être accessible facilement mais surtout rapidement, être la plus complète possible et la plus juste. Pour certaines voies professionnelles, cette importance d'apport d'informations est même une nécessité. Aujourd'hui, cela est possible grâce au numérique et à internet. Cependant, lorsqu'on regarde certains métiers cela peut s'avérer compliqué, comme l'herboristerie. Connaissances ancestrales, revenu très fortement au goût du jour, cependant très mal référencées. En effet, il est très difficile à l'heure actuelle, de trouver la bonne information sur un tel sujet.

L'idée d'une application web, semble alors la solution. Permettant de centraliser l'information par le partage d'experts et d'amateurs aguerris. Un wiki des plantes, Proposant des recettes de grand-mère ou médicinales, des conseils, mais surtout des descriptions complètes des différents produits et plantes existantes. Telle une abeille, tirer le meilleur des vertus des plantes.

Plusieurs interrogations sont à prendre en compte : peut-on faire confiance a de telles recettes ? comment s'y retrouver sans danger ? peut-on réellement tout faire dans ce domaine ? comment peut-on trouver la « vrai » information ? La problématique était posée, l'idée était née.

Pour réaliser un projet de cette envergure, en tant que développeur, le choix des technologies fut l'objet d'une étude approfondie, afin de répondre aux mieux aux besoins d'une application de ce type. Un descriptif de celles-ci sera donc proposé.

Ensuite, l'ensemble du projet, de conception au résultat final, seront vus, détaillés, expliqués et analysés, permettant une compréhension plus aisée de ce qui est proposé. En effet, plusieurs étapes ont été nécessaire pour construire le projet, tant niveau back-end que front-end. Tout ceci dans le but de répondre aux plus près de la problématique.

Bien sûr, un tel projet amène son lot de complication. Tant pour le développement du code source que pour la mise en place de la sécurité. Effectivement, ce genre d'application demande à la fois un contrôle sur les informations renseignées, mais également, elle doit proposer à l'utilisateur une expérience agréable, tant dans sa recherche que pour l'ajout d'informations.

Enfin, la conclusion apportera le bilan de cette réalisation, de cette expérience. Il sera alors possible de poser un regard constructif sur ce qui été mis en place et envisager les améliorations futures. Petite rétrospective du projet, du travail en équipes et des connaissances acquises durant toute cette période.

Mise en contexte

« Etancher sa soif du savoir a toujours demandé du temps, du travail, de la passion, des rencontres. Rien de tout cela n'a changé, bien sûr. Mais notre époque place l'assoiffé de connaissance devant de nouveaux périls qui pourrait lui faire avaler du poison s'il n'y prend garde »

Normand Baillargeon

Rien n'est plus vrai, l'avènement d'internet offre de nombreux avantages dans l'apport d'informations. Cependant, la vigilance reste de mise. On y trouve tout mais aussi n'importe quoi. Et le domaine de l'herboristerie n'en fait pas exception. Plusieurs constats sont à mettre en évidence : le sujet est très mal documenté sur internet, mais surtout aucune centralisation de l'information n'est réellement disponible.

Pourtant, la génération d'aujourd'hui, se veut de plus en plus en adéquation avec la nature, avec une forte demande de produits naturels, locaux et fait maison. Cependant, on ne peut pas faire tout ce qu'on veut même avec les plantes. Sont-elles compatibles entre elles ? sont-elles comestibles ? est-ce qu'il existe des risques d'allergie ? qu'elles sont leurs réelles vertus et bienfaits ? ne sont là qu'un petit panel de question légitimes à se poser dans ce domaine.

Une application web est donc la meilleure option pour répondre à cette demande, en répondant aux mieux à la problématique posée.

Celle-ci permet donc avant tout de centraliser toutes informations sur le sujet, proposant des fiches techniques sur les différentes plantes, des recettes médicinales ou façon grand-mères. Mais également des conseils pratiques.

Elle permet également l'assurance de la véracité de l'information proposée. Pour ce faire, elle fait appel à un panel d'experts qui contrôle chaque nouvel ajout avant la publication. En procédant à un système de vote par ceux-ci. L'auteur a alors la possibilité de procéder à des modifications, après différents échanges avec les experts. Cela permet avant tout de comprendre l'erreur d'écriture mais également de partager son expérience.

Un procédé d'annotations est également proposé, permettant d'informer l'utilisateur en cas de danger sur l'emploi d'un ingrédient. Celui-ci, sera dès lors redirigé vers les fiches techniques disponibles pour être informé des risques d'allergies ou autres incompatibilités.

Une référence vers ces fiches est également mise à disposition de l'utilisateur lui permettant de prendre connaissance des propriétés de chaque ingrédient d'une recette. Cette redirection apporte plusieurs avantages : d'une part, cela facilite la recherche et la navigation sur le site, et permet également à l'utilisateur d'accroître ces connaissances.

Enfin, le lecteur peut laisser un commentaire sur une publication. Cette option apporte également plusieurs avantages tant pour l'utilisateur que pour les contributaires. L'utilisateur peut dès lors demander un conseil, des informations supplémentaires ou laisser son appréciation, et le contributaire peut quant à lui, connaître l'intérêt porté à sa publication et apporter des compléments d'informations.

Technologies

Pour la réalisation de ce projet, une fois la demande définie, reste le choix des technologies à déterminer. Une analyse des différentes options de développement a donc été réalisé. Cette étape est indispensable, permettant de définir les différents éléments techniques à prendre en compte. Tant pour le back-end, le front-end que pour la création de la base de données, plusieurs offres sont disponibles sur le marché. Il n'est pas toujours facile de prendre les bonnes décisions, en effet, celles-ci proposent toutes, à différents niveaux, nombre d'avantages et d'inconvénients. Voici un descriptif des différentes technologies, framework ou dépendances, ainsi que leurs fonctionnalités qui ont guidés ces choix pour le développement de ce projet

1. Back-end



spring boot

Figure 1 : logo de Spring Boot

Flexibilité, rapidité et efficacité sont des mots clés dans tout développement. Les Spring en général et Spring Boot plus particulièrement sont les technologies par excellence pour la production fonctionnelle du back-end. Allié au bon framework comme Angular, celui-ci peut vraiment devenir très puissant. Concrètement Spring Boot est un procédé d'exécution permettant les choix des différents chemins de code dans un système logiciel. Spring Boot est donc un Framework de développement Java pour la création d'API RESTful, composé de nombreuses fonctionnalités permettant une mise en service rapide, flexible et de haute qualité. ORM est l'une de ces fonctions intéressantes.

- ORM pour Mappage Objet-Relationnel. Ce mappage fonctionne comme un pont bidirectionnel entre une base de données relationnelle et les différentes classes et objets de l'application. Il convertit ceux-ci dans le but de les stocker et de les gérer dans la base de données. Ainsi capable de reproduire les modèles métiers qui représentent cette base de données.

En comparaison avec une architecture hexagonale, Spring Boot évite pour l'ajout de nouveaux composants la réécriture du code en proposant des features plug and play. Comme « Mapper », en tant que boîte à outils pour combiner à la fois l'ingénierie de ligne de produits logiciels (SPL) et le développement de logiciels piloté par modèle. Les « DTO » (Data Transfert Object) permettant la simplification des transferts de données entre les sous-systèmes d'une application. Mais également des notions de « Repository » permettant une centralisation et une organisation de stockage de données en vue d'une distribution réseau ou directement accessible aux utilisateurs. Spring Boot dispose également d'une version spécialement conçue pour la sécurité, « Spring Boot Sécurité ». Offre un cadre d'authentification et de contrôle d'accès afin de sécuriser les applications basées sur Spring. Celui peut être associé à JWTToken permettant l'échange sécurisé de jetons entre différentes parties. Celui-ci se traduit par la vérification de l'intégrité des données à l'aide de signatures numériques. Offrant une couche de sécurité supérieure en permettant par exemple de faire naviguer les informations de connexion utilisateur une seule fois, pour ensuite travailler avec des tokens cryptés à chaque demande.

<i>Avantages</i>	<i>Inconvénients</i>
Flexibilité et modulaire	Manque d'outils d'analyse de performance
Sécurité	Dépendance souvent inutile, augmentant considérablement la taille du fichier de déploiement
Communauté forte	
Open source	

Tableau 1 : tableau des avantages et des inconvénients de Spring Boot

2. Base de données



Figure 2 : logo de PostgreSQL

Résiliences, intégrité et exactitude de hauts niveaux sont les premiers atouts mis en avant pour ce système de gestion. Celui-ci est le résultat de 20 ans de développement communautaire. PostgreSQL est le concurrent direct des autres systèmes de gestion qu'il soit libre comme MariaDB ou propriétaire Comme Oracle ou MySQL. En effet, en comparaison, il donne la priorité à la conformité et à l'extensibilité SQL, que MySQL par exemple se concentre sur l'évolutivité et les performances. Au contraire des autres systèmes traditionnels, PostgreSQL à l'avantage de posséder une architecture orientée objet, le rendant extensible et donnant aux utilisateurs de nombreuses possibilités. Comme permettre l'ajout de nouveaux types de données, ou fonctions SQL afin de personnaliser la manière dont les données sont stockées et interagissent à la base de données. Proche d'Oracle, PostgreSQL est largement reconnu pour son comportement stable, mais également pour ses possibilités de programmations étendues. Il propose entre autres des méthodes économiques d'optimisations de requêtes basées sur des jointures, ainsi qu'une validation stricte des données avant tout insertion à la base de données.

<i>Avantages</i>	<i>Inconvénients</i>
Open-source	Requêtes de calculs assez lentes
Intégrité des données	
Moteur d'annulation de requêtes	
Analyseur de coût de requêtes	

Tableau 2 : tableau des avantages et des inconvénients de PostgreSQL

3. Front-end



Figure 3 : logo de Angular

Google a lancé ce Framework web open source côté client en 2009, voulant ainsi résoudre les multiples problèmes liés à la création d'applications sur une seule page. Ces pages web uniques permettent dès lors de fluidifier l'expérience utilisateurs et évitent ainsi des chargements de pages à chaque nouvelle action. Voulant voir le web comme des applications interrelées et non plus comme des sites web contenant des pages. Pour ce faire, Angular fournit nativement tout le nécessaire pour la production des « Progressive Web Apps », offrant l'illusion d'une application native, tout en restant résilient aux soucis de connexions. Contrairement à d'autres alternatives, Angular est intéressant car c'est un réel Framework à part entière et non une simple « Library », avec une approche particulière pour les « Batteries Included ». Outre ces quelques caractéristiques avantageuses, voici plusieurs raisons de choisir Angular comme Framework. L'interface utilisateur : pour définir son interface utilisateur, Angular utilise l'HTML qui est langage déclaratif et moins fragile. Ce qui permet de simplifier le processus de développement. Les tests : cela facilite l'implémentation des tests unitaires en fournissant tous les outils nécessaires. Flexibilité et maintenabilité : ce Framework est basé sur une architecture de types MVC (modèle vue contrôleur), ce qui lui permet de différencier les données, le visuel et les actions, garantissant ainsi une meilleure gestion des responsabilités. Ce modèle a largement fait ses preuves et apporte une forte maintenabilité et une amélioration du travail collaboratif.

Avantages	Inconvénients
Synchronisation automatique des données entre la vue et les composants	Les éléments du DOM ont des problèmes de performance
Cadre facilement testable	Les portées sont difficiles à déboguer
Routage simple	Routage simple mais limité
Solution de construction de modèles robustes	Intégration tierce complexe
	Courbe d'apprentissage abrupte

Tableau 3 : tableau des avantages et des inconvénients de Angular

4. Dépendances et logiciels



Figure 4 : logo de Postman

Postman, permet de tester directement les interfaces de programmation d'application. Dans un cadre d'intégration, il permet également de déterminer le bon fonctionnement en matière de fonctionnalité, de fiabilité, de performance et de sécurité. Se voit comme une plate-forme de collaboration pour le développement d'API. Ses nombreuses fonctionnalités simplifient chaque étape de la création.



Figure 5 : logo de Swagger

Swagger est framework open source constitué d'une boîte à outils permettant à tout développeur de documenter et de modéliser les API. Il permet également la synchronisation avec le code source. Au fil du temps, il a bien sûr évolué et propose de nouvelles commodités, toujours dans le but de concevoir, créer et utiliser et documenter les différents services.



Figure 6 : logo de Bootstrap

Bootstrap, reprend dans une feuille de style CSS toutes les définitions de base pour les différents composants HTML. Présentée comme une librairie HTML, CSS, JS avec l'ajout de plugins JQuery, permettant avant tout de rendre le visuel plus attractif et responsive. Bootstrap permet donc d'un rendu uniforme et dispose de multiples éléments graphiques dans un format standardisé. Offrant un gain de temps de la réalisation et une interface fluide pour l'utilisateur.



Figure 7 : logo de Material Angular

Angular Material, dans une même vision de Bootstrap, propose de nombreuses facilités pour les interfaces utilisateur. Permettant de construire des pages et/ou des applications web attrayantes, cohérentes mais surtout fonctionnelles. Cette bibliothèque respecte tous les principes de conception, comme les notions de portabilité des navigateurs, l'indépendance des appareils ou encore la dégradation progressive. Permettant la création de pages plus rapides, mais également plus belles et plus réactives. Celles-ci sont composées d'une suite de composants préconstruits pouvant être directement introduits à l'application. Elles possèdent une particularité intéressante, le comportement des composants peuvent être contrôlés et réutilisables.

5. Méthodologie

Ce projet a été développé en collaboration. Pour faciliter le travail de chaque développeur, sans que ceux-ci interfèrent sur le code de l'autre, le système de versioning Git, couplé avec GitHub Desktop ont été préconisé. En effet, ils sont la référence en termes de système de contrôle. Conçus pour tout gérer, du projet le plus simple au plus complexe. Cela a permis d'apporter de nombreux avantages au développement de cette application. D'une part, du fait que Git soit distribué permet un clonage du référentiel, laissant à chaque intervenant une sauvegarde complète du serveur principal. De plus grâce au principe d'effectuer les opérations en local, apporte un énorme avantage de vitesse.

Conception

1. Diagramme de contexte

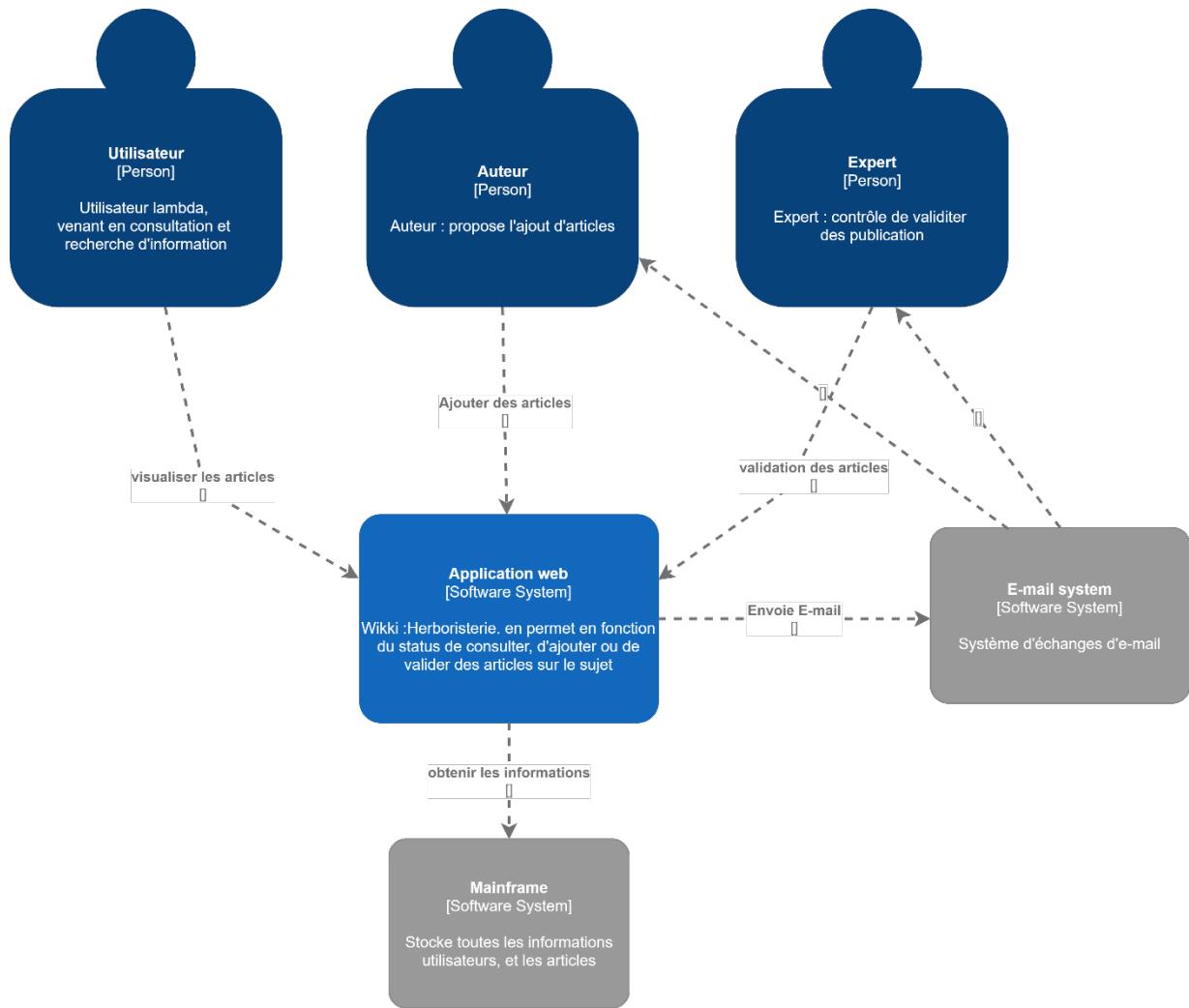


Figure 8 : diagramme de contexte

2. Vision d'une page

L'application propose plusieurs modèles de page, en fonction d'une catégorie, d'une section, d'une correspondance, ou du genre de contenu. Chaque type de page disponible sont au départ composé de la même manière, dans leur version la plus simpliste. Une image, un titre et une description. Lors de la création d'une nouvelle page, plusieurs actions entrent en compte avant même de pouvoir la valider.

- Récupérer l'auteur ainsi que la date de création correspondante, afin de permettre son affichage ;
- Etablir l'association à une catégorie distincte, permettant à la nouvelle page d'être répertoriée à la place représentative de son contenu ;
- Etoffer la nouvelle page en employant les différentes sections disponibles :
 - Paragraph : élément de base dans sa forme la plus classique. Proposant simplement une section avec un titre et un contenu ;
 - Parapage : permet la redirection vers l'information d'une autre page. Proposant ainsi de référencer des éléments déjà cités dans d'autres publications de l'application ;
 - Paratag : version similaire mais améliorée du paragraph. Permet de faire un condensé de ses propres tags afin de les ajouter à un contenu et au titre.
 - Tags : ajoute des informations de nature distinctive en démarquant celles-ci d'une autre, en lui donnant de l'importance.

Le graphique ci-dessous représente les différentes interactions possibles lors de la création d'un article. On peut constater que plusieurs étapes sont à prendre en considération. Dans un premier temps, l'auteur une fois la création terminée choisit l'option soit de sauvegarder ou de lancer la procédure de validation. Si c'est cette deuxième qui est sélectionnée, la procédure de contrôle sera lancée, elle seule, permettra la publication de l'article. Pour ce faire elle devra recevoir l'approbation de cinq experts, par système de vote. Dans le cas d'un refus, même d'un seul expert, se suivra un échange de messages entre les deux partis, permettant de comprendre d'une part le refus et de l'autre de comprendre l'erreur commise lors de la rédaction de l'information. Lorsque les pages sont disponibles à la consultation sur le site, une particularité est proposée sur certaines d'entre elles en plus de la possibilité de laisser des commentaires. Il s'agit ici, d'un indicateur permettant d'alerter les utilisateurs. Il existe de deux types :

- En cas de danger : Indicateur de danger, qui permet d'alerter l'utilisateur d'un risque. Cet élément est mis en évidence, pour lui donner la priorité lors de la consultation.
 - Exemple : si la recherche porte sur une plante ou une recette dont un ingrédient peut provoquer des effets contraires, le lecteur en sera averti à l'aide de cette indicatrice. Il pourra alors avant toutes autres actions consulter cette liste des indésirables ou des dangers.
- Recommandée : permet des redirections vers les différentes pages descriptives.
 - Exemple : si le lecteur recherche une recette, et que celle-ci référence une plante particulière, un lien de référence sera disponible pour rediriger la consultation vers le descriptif correspondant.

2.1 Diagramme de création

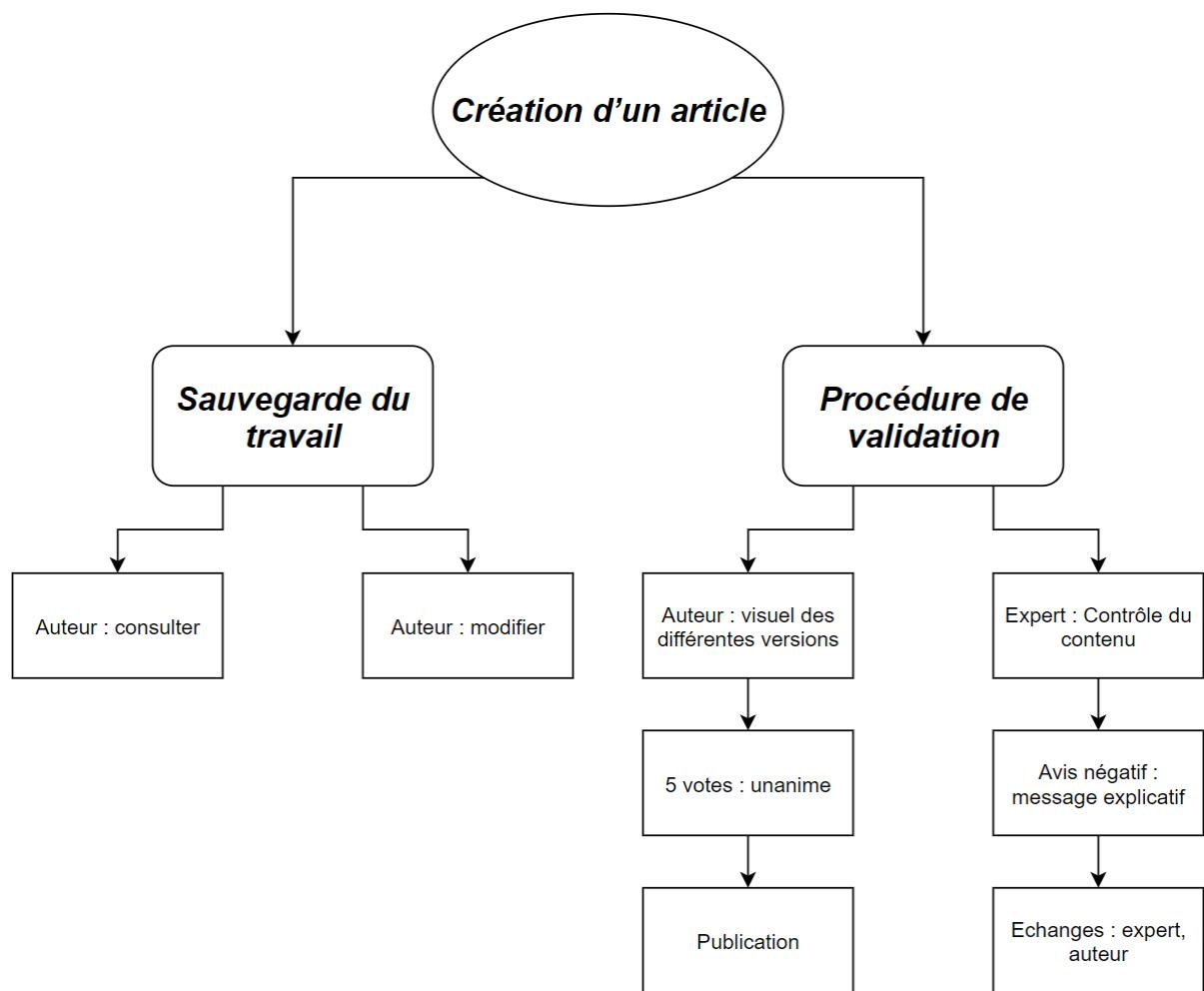


Figure 9 : diagramme de création

Réalisation Back-end

Pour la réalisation de ce projet, un back-end a été mis en place. Celui-ci a fait l'objet de plusieurs étapes de construction : la création des différents packages, l'installation de dépendances, la création des différentes classes nécessaires ...

1. Les Packages

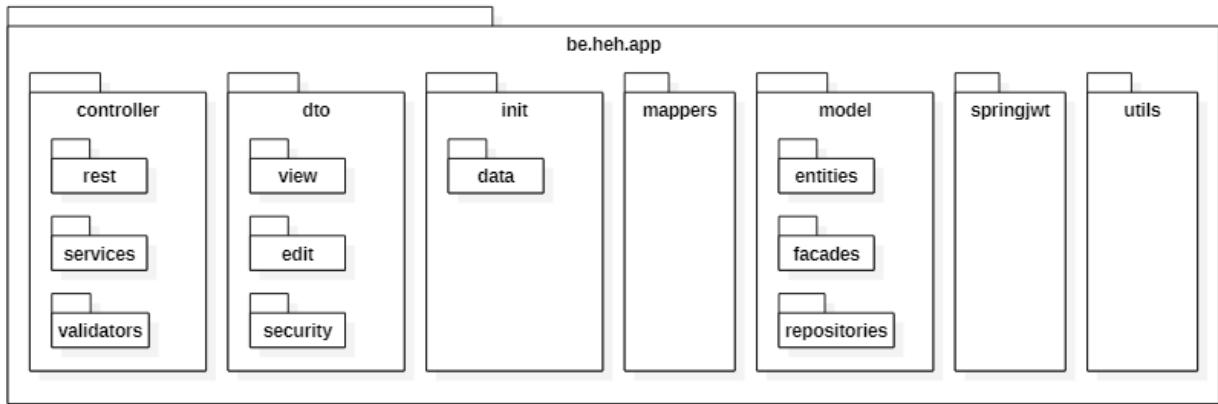


Figure 10 : diagramme représentatif des packages

Comme le propose le tableau ci-dessus, plusieurs packages sont partie intégrante de la mise en place du back-end. Dont voici la liste :

- Controller, composé lui-même de trois sous-packages
 - Rest
 - Services
 - Validator
- Dto
- Init
- Mappers
- Model, composé lui-même de trois sous packages
 - Entities
 - Facades
 - Repositories

1.1 Package « Controller »

1.1.1 Package « Rest »

Le package « Rest » rassemble les différents contrôleurs REST de l'API. L'API REST utilise des requêtes http pour les méthodes GET, PUT, POST et DELETE. Devenue rapidement la norme en termes de communication entre les différents services d'une application. « *RestController* » de Spring Boot permet de créer ces API en exposant les méthodes citées précédemment.

Concrètement « *RestController* » est une annotation Spring Boot permettant de créer des API REST de manière déclarative. Celle-ci est appliquée à une classe pour la définir comme gestionnaire de requêtes. Spring Boot effectue alors la construction et fournit le service au moment même de l'exécution. Elle permet également de vérifier le rôle utilisateur exécutant une requête. En effet, trois statuts sont définis, dont celui d'administrateur donnant l'accès à des pages réservées.

1.1.2 Package « Services »

Le package « *Services* » a accès à une multitude d'autres packages, le service procède à différentes opérations telles que : la vérification de l'existence des données, la récupération et sauvegarde de celles-ci via les *Repositories*. Les composants du package « *Services* » sont des classes contenant l'annotation correspondante « *@Service* ». Ces classes sont uniquement utilisées pour écrire la logique, cela permet de séparer la logique des contrôleurs REST. Le package « *Services* » peut également appeler des mappers permettant la création ou la modification de ces données.

1.1.3 Package « Validators »

Les classes du package « *Validators* » sont construites avec différents attributs. Dès lors, il est indispensable d'ajouter des contrôles sur ces attributs, comme par exemple un regex ou un vérificateur de non nullité. La validation ne doit pas être dans le front-end, mais dans le back-end.

1.2 Package « Dto »

Le package « *DTO* » contient des classes « *DTO* » ou objets de transferts de données permettant de redéfinir les données. Lorsque les données sont envoyées vers le front-end, elles se voient épurées de toutes données inutiles. Cela a pour but de permettre un envoi de données moins conséquentes, évitant que des informations confidentielles ne circulent inutilement et permettant d'encapsuler ces données et de les envoyer dans le bon format.

1.3 Package « Init »

Le package « *init* » contient des classes permettant de fabriquer de fausses données pour l'application. De plus il possède des classes « *données* » contenant des répertoires d'instances de repositories, de facades, de services et de mappers.

1.4 Package « Mappers »

Le package « *Mappers* » est utilisé par les services, permettant dans un premier temps la création et la modification des entités en utilisant les facades. Le mappage d'objets offre aussi une aisance

pour la conversion d'un modèle à un autre. L'objectif de « Mapper » est donc de simplifier le mappage.

1.5 Package « Model »

1.5.1 Package « Entities »

Le package « Entities » est composé de plusieurs classes qui sont la représentation des différentes tables d'une base de données relationnelles dont chaque instance correspond à une ligne de cette même table. Au sein de ces classes, les cardinalités sont renseignées sous forme d'annotation.

1.5.2 Package « Facades »

Le package « Facades » sert à séparer le constructeur des entités et permet de créer ou de modifier les instances via les informations renseignées en paramètres. La façade est un patron de conception structurel. La façade ajoute cette interface au système existant pour masquer les complexités générales d'une application. Elle permet également de regrouper les différentes dépendances indésirables au même endroit.

1.5.3 Package « Repositories »

Les interfaces « Repositories » héritent de l'interface « AbstractRepository ». Leur but est de rendre la création de la couche d'accès aux données plus rapide. Cela permet également de minimiser la dispersion et la duplication du code des requêtes. Cela permet aussi l'emploi des méthodes liées au CRUD sans avoir besoin de les créer, ainsi que la création de requêtes complexes avec l'annotation « @query » permettant des recherches spécifiques dans la base de données.

2. Dépendances

2.1 Lombok

Java est un langage puissant, mais parfois considéré comme trop verbeux pour les tâches simples. L'outil de développement Lombok permet d'accroître la productivité en fournissant des annotations et en éliminant une grande quantité de codes standard. La résultante de son emploi est que les classes deviennent alors plus propres, plus concises et faciles à gérer. Pour ce faire Lombok met à disposition un générateur pour l'automatisation des variables et méthodes.

2.2 Thendenda

Lors de la création d'application de démonstration, la nécessité de présenter du texte représentatif est souvent nécessaire. Pour ce faire, la dépendance « Thedenda » permet de

générer automatiquement des textes en latin de la taille souhaitée et permet la création de données fictives dans le but d'obtenir un visuel le plus proches du résultat final souhaité.

2.3 Javax.mail

Proposer la fonctionnalité d'envoi de mails par le biais d'une boîte de messagerie est assez commun dans une application. JavaMail sert à composer, écrire ou lire des messages électroniques, en fournissant l'ensemble de classes pour la modélisation d'un système de messagerie. Le package Javax.mail, quant à lui, contient les classes de base de JavaMail. Celles-ci sont alors définies pour tous les systèmes de messagerie.

2.4 Junit

C'est un framework qui permet le développement et l'exécution de tests unitaires automatisables. L'intérêt d'un tel outil réside dans l'assurance que le code répond toujours correctement aux besoins, et ce même après modifications. Junit propose :

- Un framework dédié au développement des tests unitaires reposant sur le principe des assertions qui testent les résultats attendus ;
- Des applications pour permettre l'exécution des tests et afficher leur résultat.

Le but premier est donc l'automatisation des tests. Ceux-ci sont représentés dans des classes sous la forme de cas tests avec les résultats souhaités.

2.5 Jsonwebtoken

« Json web token » propose un accès par token permettant un échange sécurisé de donnée entre deux parties. Cet outil est surtout intéressant pour les opérations d'authentification. Les tokens sont générés par un algorithme puissant (HS256) et offre l'avantage d'être sécurisés et signés. Ce qui permet d'augmenter considérablement la sécurité de l'API.

3. Le modèle métier

3.1 La classe abstraite « AbstractEntity »

Toutes les classes du model métier héritent de la classe abstraite « AbstractEntity », en effet toutes les classes du model métier qui sont une représentation de nos tables en base de données ont besoin des deux mêmes attributs :

- Un id unique
- Une date de création

La date de création permet de faire des recherches plus précises ou de faire des tris par date de création.

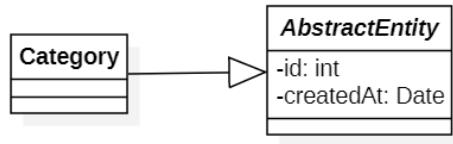


Figure 11 : diagramme de la classe abstraite : AbstractEntity

La classe « Category » est juste un exemple.

3.2 La classe « User » et la classe « UserSecurity »

Il y a deux classes pour les utilisateurs. La classe principale « UserSecurity » qui reprend les informations personnelles des utilisateurs, celle-ci est séparée du modèle métier pour éviter que celui-ci connaisse les informations personnelles des utilisateurs.

Ensuite, la classe User qui peut être vue comme un auteur. La classe User contient :

- Un string : username
- Un UserSecurity

Dans le cas où l'utilisateur supprime son compte, la classe « UserSecurity » sera supprimée mais la classe User restera intacte, cela permet de garder un lien entre les utilisateurs et la page qu'il aurait pu créer ou les messages envoyés.

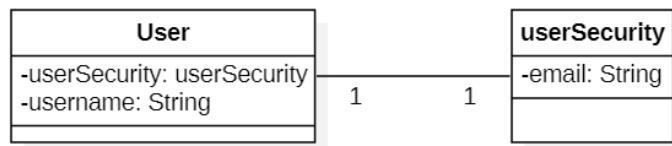


Figure 12 : diagramme de la classe : User et UserSecurity

3.3 L'énumération « EnumState »

L'énumération « EnumState » est composée de 4 entités permettant de définir l'état d'avancement d'autres objets.

- « **DRAFT** » est l'état signalant que l'objet est en cours de réalisation.
- « **VALIDATING** » est l'état signalant que l'objet est en cours de validation par les administrateurs.
- « **VALIDATED** » et « **NOT_VALIDATED** » sont les états résultant du vote des administrateurs.

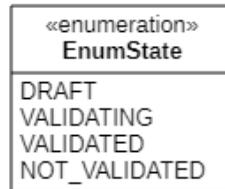


Figure 13 : diagramme de l'énumération « EnumState »

3.4 La classe abstraite « AbstractAutowire »

La notation Autowire permet de faire de l'injection automatique de dépendances. Cela va permettre d'avoir accès aux Mappers, aux Façades, aux Services et aux Repositories dans n'importe quelle classe qui hérite de la classe abstraite « AbstractAutowire ». Ceci a été fait pour pouvoir accéder à toutes les dépendances avec plus de facilité.

3.4 Les types et les catégories

Une Catégorie est composée d'une liste de types qui ensemble formeront une convention. Les types sont des sections qui formeront une page. L'avantage de cette convention réside dans le principe que toutes les pages appartenant à une même catégorie et auront la même structure. Un type va être une section de la catégorie, il existe plusieurs types :

- paragraphType
- paratagType
- parapageType

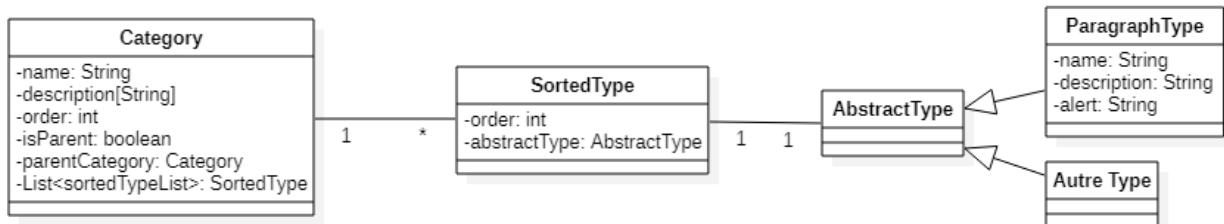


Figure 14 : diagramme de la classe Category et de la classe SortedType

Pour la suite des explications le type « paragraphType » sera utilisé. En effet, celui-ci étant le plus simple d'entre tous, il sera plus facile d'expliquer sa structure. Cependant ceci est aussi valable pour les autres types.

Une catégorie contient :

- Une liste SortedType,
 - Un SortedType est défini par un « AbstractType » et un « order ». Tous les types héritent de la classe abstraite « AbstractType » qui ne contient rien. L'utilité d'avoir cette classe est qu'il est possible d'ajouter tous les types dans une seule liste car ils héritent tous de « AbstractType ».
 - Par exemple la liste « SortedType » peut contenir un « paragraphType » ou « paratagType ».

- Une « Category parentCategory » stocke un parent, cela permet d'avoir une arborescence de catégorie.
- Un « boolean isParent » :
 - S'il est vrai la catégorie est un parent, en d'autres termes il possède des enfants. Et donc la création de pages avec cette catégorie n'est pas possible.
 - S'il est faux la catégorie ne peut pas avoir d'enfant mais il est possible de créer des pages depuis cette catégorie.
- Deux strings : name et description.

3.5 Les pages et leurs contenus

La classe page contient :

- Une classe « Category »
- Une classe « User »
- Une énumération « EnumState »
- Une liste innerPageList qui contient des « InnerPage »
- Une liste paragraphList qui contient des « Paragraph ».

La classe « *page* » contient également deux autres listes qui représentent les autres types :

- « *paratag* »
- « *parapage* »

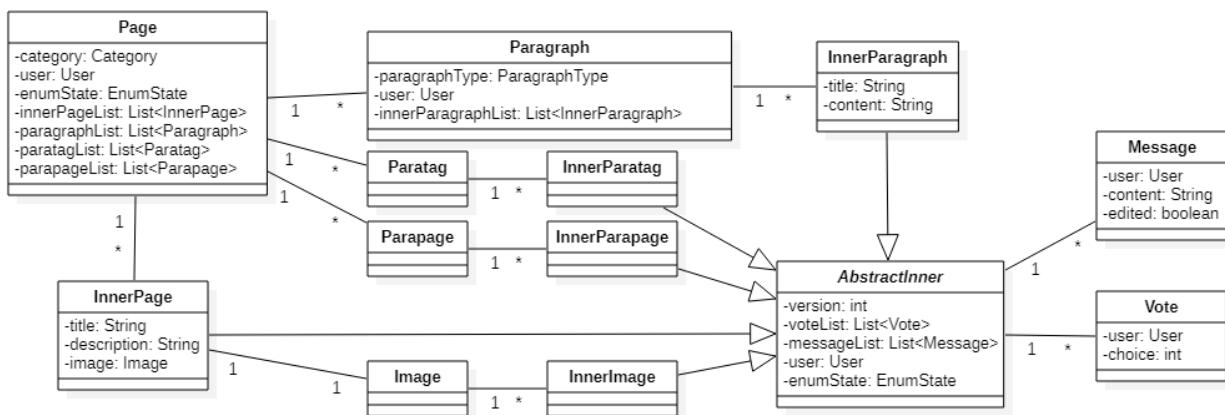


Figure 15 : diagramme de la classe page et de leurs contenus

Dans la classe « *Page* », il y a une liste de « *Paragraph* ». Ces « *Paragraph* » sont composés d'un « *ParagraphType* », de l'auteur et d'une liste d'« *InnerParagraph* ». La classe « *InnerParagraph* » contient, quant à elle, un titre et un contenu.

La classe « *InnerParagraph* » permet de créer un système de version, effectivement, chaque « *InnerParagraph* » est une nouvelle version du « *Paragraph* ». La liste « *InnerParagraph* » va donc stocker chaque version du paragraphe. La liste va être composée d'absolument toutes les versions, qu'elles soient en cours de validation, validées, sauvegardées ou non validées.

La classe « *InnerParagraph* » hérite de la classe abstraite « *AbstractInner* ». Cette classe abstraite contient :

- Un numéro de version
- Un auteur (« *User* »)
- Une énumération « *EnumState* »
- Une liste de messages permettant de communiquer
- Une liste de votes qui va permettre de valider ou de refuser la classe « *InnerParagraph* »
 - Si le vote est accepté à l'unanimité, l'état de l'énumération « *EnumState* » passe à « **VALIDATED** »
 - Si le vote est refusé, l'état de l'énumération « *EnumState* » passe à « **NOT_VALIDATED** » et le votant est dans l'obligation de transmettre un message

L'explication ci-dessus est vraie pour « *Parapage* », « *Paratag* », « *Image* », « *Tag* » et « *Page* ». Néanmoins certaines différences persistent mais la structure des points suivants est la même que la classe « *InnerParagraph* »

- La classe « *InnerPage* » est composée en plus d'une image.
- La classe « *InnerImage* » est composée en plus d'une url.
- La classe « *InnerParatag* » est composée en plus d'une liste de tags.
- La classe « *InnerParapage* » est composée en plus d'une liste de pages.

3.6 Le modèle métier utilisateurs

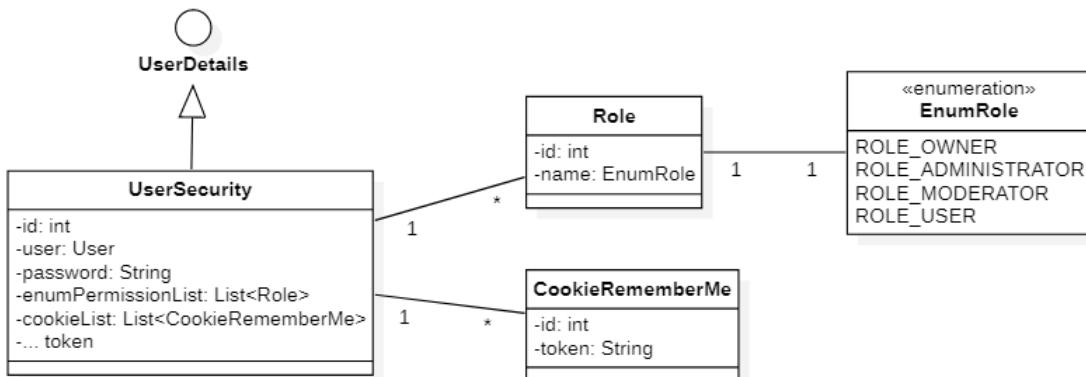


Figure 16 : diagramme du modèle métier utilisateur

La classe « *UserSecurity* » contient :

- Une liaison avec la classe « *User* »
- Une liste « *enumPermissionList* » qui permet de définir les différents rôles à associer à un utilisateur. Cette liste est composée de différents rôles. Un rôle est défini par une énumération « *EnumRole* » qui est composée de 4 entités permettant de définir les rôles.

- « **ROLE_OWNER** »
- « **ROLE_ADMINISTRATOR** »
- « **ROLE_MODERATOR** »
- « **ROLE_USER** »
- Une liste « *cookieToken* » contenant des classes « *CookieRememberMe* », cette classe contient un token qui est également sauvegardé dans les cookies du navigateur web. Cela permet une connexion instantanée sur un appareil déjà connecté.
- Un string *password* qui représente le mot de passe de l'utilisateur. Celui-ci est haché grâce à la fonctionnalité de Bcrypt.
- Possède également plusieurs autres token permettant la validation de l'adresse mail, la réinitialisation du mot de passe, la suppression du compte, ...
- Ainsi que plusieurs boolean permettant de définir si le profil est privé, si l'interface de l'utilisateur est en mode sombre...

« *UserSecurity* » hérite de l'interface « *UserDetails* », cela permet l'intégration du système d'authentification et de gestion de rôle via Spring Security.

3.7 Les repositories



Figure 17 : diagramme des repositories

Le schéma ci-dessus montre trois interfaces. Tout d'abord la première interface, celle-ci représente le repository de category qui va implémenter l'interface « *AbstractRepository* ». Cette interface est vide, elle a été créée dans le cas où il y aurait besoin de factoriser du code. Cependant, elle va implémenter « *JpaRepository* » qui est l'interface permettant l'accès aux méthodes CRUD.

Le générique « *T* » dans le schéma représente l'entité passée en paramètre. Celle-ci peut être par exemple une « *Category* ». Enfin le générique « *I* » représente un « *Integer* » qui est l'*Id* de l'entité qui est fournie en paramètre.

3.8 L'interface « InnerRepositories »

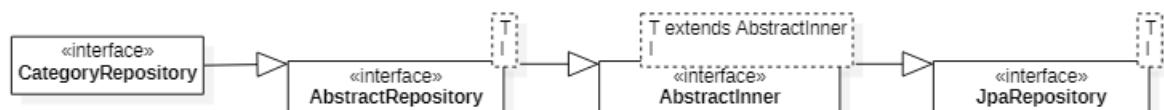


Figure 18 : diagramme de l'interface « *InnerRepositories* »

Dans ce schéma il existe des similitudes avec le schéma vu précédemment, en effet, la différence majeure entre ces deux schémas réside dans l'ajout de l'interface « *AbstractInnner* ». Cette interface a été créée dans le but de contenir une méthode « *hasVoted()* » permettant d'exécuter une query. Celle-ci permet de vérifier si l'utilisateur a déjà voté sur l'entité courante. Il faut savoir que toutes les entités « *Inner* » contiennent une liste de votes d'où la nécessité d'avoir créé cette interface « *AbstractInnner* ».

Il y a le même système avec les génériques « *T* » et « *I* » que dans le schéma précédent, voir figure 17.

3.9 Les façades

L'idée principale de la Façade est que les constructeurs devant se situer normalement dans les entités ont été déplacés dans les Façades. Ce qui permet d'alléger les classes entités. La Façade permet donc de créer des entités mais aussi de les modifier.

Dans le schéma ci-dessous toutes les façades vont hériter de la classe abstraite « *AbstractFacade* ». La décision d'un tel procédé réside dans fait que toutes les façades vont posséder une date et que celle-ci doit être définie. Le « *T* » dans le schéma représente l'entité « *Category* ».

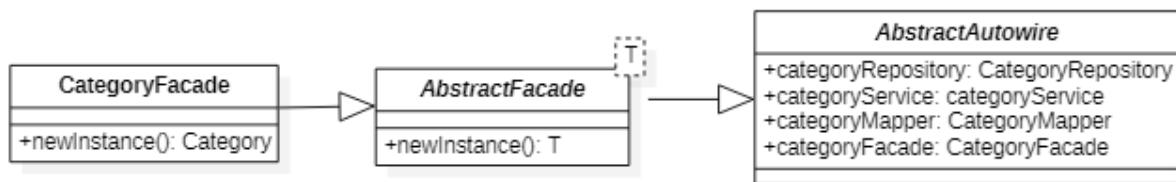


Figure 19 : diagramme des façades

La classe « *CategoryFacade* » est juste un exemple.

Dans le premier code proposé, l'objet « *Category* » est instancié en appelant la classe abstraite « *AbstractFacade* ». Dans celle-ci la méthode « *newInstance* » crée un objet « *abstractEntity* », elle lui définit la date actuelle et transforme l'objet « *abstractEntity* » en type « *Category* » et le renvoie.

```

public Category newInstance(String name{
    Category res = super.newInstance();
    res.setName(name);
    return res;
}
  
```

Code 1 : méthode permettant de créer une nouvelle instance

```

public T newInstance() {
    AbstractEntity obj = null;
    Object instance = ((Class) ((ParameterizedType)
        this.getClass().getGenericSuperclass()).getActualTypeArguments()[0]).newInstance();
    obj = (AbstractEntity) instance;
    obj.setCreatedAt(new Timestamp(System.currentTimeMillis()));
    return (T) obj;
}
  
```

Code 2 : méthode dans la classe abstraite généraliser

4. La sécurité

4.1 La configurations de la sécurité avec Spring Boot

Dans le schéma ci-dessous est représentée la classe « `SecurityConfig` ». Celle-ci est composée d'une liste de `String` qui va contenir toutes les URLs d'entrée accessibles dans l'API. Ce qui a pour effet de bloquer tout autre demande faite à l'API avec une URL qui n'est pas contenue dans cette liste.

Ensuite il existe la classe abstraite « `WebSecurityConfigurerAdapter` », celle-ci est le cœur de l'implémentation de la sécurité. Elle fournit des configurations « `HttpSecurity` » pour configurer les cors, les csrf, la gestion des sessions, les règles pour les ressources protégées. La classe « `SecurityConfig` » hérite donc de cette classe pour ensuite redéfinir les méthodes de sécurité par défaut.



Figure 20 : diagramme de la classe sécurité

4.2 Les filtres

Pour ajouter une couche de sécurité supplémentaire l'ajout des filtres a été réalisé. En effet, le schéma ci-dessous, représente la classe « `RequestFilter` » qui permet d'implémenter l'interface « `Filter` » et ainsi de redéfinir la méthode « `doFilter()` ». Cette méthode est redéfinie pour permettre l'implémentation de nouveaux filtres.

Ces filtres possèdent plusieurs paramètres d'acceptation :

- Si l'origine de la requête est <http://localhost:4200>
- Si la méthode de la requête fait partie de l'une de ces méthodes :
 - POST
 - PUT
 - OPTIONS
 - DELETE
- Si le header est présent dans la requête
- Si le header présent possède l'un des noms suivants :
 - « Autorisation »
 - « Content-type »

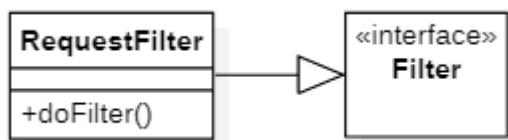


Figure 21 : diagramme des filtres

Ces filtres possèdent également un compteur de temps qui permet de déterminer la durée pendant laquelle la requête peut rester en cache. Celui-ci est déterminé à 3600 secondes (Max-Age). Ce compteur permet également de déterminer le temps d'autorisation d'identification d'un utilisateur connecté sur cette requête. Pour ce cas, cela se passe grâce au header d'autorisation (Allow-Credentials).

4.3 L'authentification

Dans la classe « *SecurityConfig* » il y a les méthodes permettant dans un premier temps de récupérer le token, ensuite grâce à ce token il est alors possible de retrouver l'utilisateur qui a émis la demande.

Par la suite grâce au package « *springjwt* » plusieurs vérifications seront exécutées. Tout d'abord, la vérification du format du token afin de déterminer si celui-ci a bien été créé par l'API de cette application, ensuite il vérifie si le token est bien en lien avec un utilisateur, après il vérifie si le token n'a pas été bloqué et enfin il vérifie si le token n'a pas atteint la limite de temps qui lui a été donnée.

Lors de la connexion d'un utilisateur, celui-ci envoi le couple email et mot de passe. L'API récupère ce couple pour ensuite vérifier si l'email existe mais aussi si le mot de passe correspond bien à celui en base de données. Une fois que cette opération est terminée un « *tokenjwt* » va être généré. Ce token va permettre de remplacer l'id dans les prochains échanges. Cette décision a été prise dans un souci de sécurité, en effet si les échanges se faisaient avec l'id, celui-ci pourrait être récupéré afin de faire des échanges non voulus par des personnes mal intentionnées. Un token a une durée de vie de 24h car il existe une connexion par cookie, celui-ci ayant une durée de vie de 1 mois.

Dans l'appel du front-end vers le back-end il faut créer un headers http qui a pour nom « *Authorization* » avec comme contenu le token de l'utilisateur comme le montre le code ci-dessous.

```
public call(body: any): Observable<any> {
  return this.http.post<any>(callUrl, body,
    {headers : new HttpHeaders().set('Authorization', JwtToken)});
}
```

Code 3 : méthode du front-end avec un header

4.4 Les validateurs

La décision de créer des validateurs pour plusieurs fonctionnalités a été prise. En effet un validateur a été mis en place pour les opérations d'ajout, de modifications et de validation. Les attributs contenus dans les validateurs possèdent des règles. Voici les principales règles :

- NotNull qui permet de définir que l'attribut n'est pas nul.

- NotEmpty qui permet de définir que l'objet n'est pas vide. Cet attribut va s'appliquer aux String.
- Size permet de définir combien de caractères minimum et maximum doit contenir l'attribut.
- Min qui permet de définir la valeur minimale autorisée pour un attribut.
- Max qui permet de définir la valeur maximale autorisée pour un attribut.
- Email va vérifier que la valeur de l'attribut est au format email.

5. Les DTO

Un Dto, comme vu précédemment, va permettre de retravailler les données, ce qui va procurer plusieurs effets bénéfiques.

Il y a 3 types de Dto qui sont :

- Le Dto de vue, celui-ci va être utilisé pour envoyer des données vers le front-end afin de lui permettre de créer la vue des pages de l'application web.
- Le Dto d'édition, il permet d'envoyer beaucoup plus de données que le Dto de vue. Effectivement, prenons l'exemple du versioning qui va demander un envoi de toutes les versions des différentes sections de contenus, ce qui fait beaucoup de données, alors que pour l'affichage il faut envoyer uniquement la dernière version.
- Le Dto de sécurité, il est utilisé pour envoyer les informations sensibles vers le front-end.

6. Les contrôleurs

Les contrôleurs permettent de définir l'URL d'entrée. Le format de l'URL est « api/v1/le_nom_du_contrôleur ». Il est également possible d'ajouter un id en paramètre directement dans l'URL : le format est alors « api/v1/le_nom_du_contrôleur/{id} ».

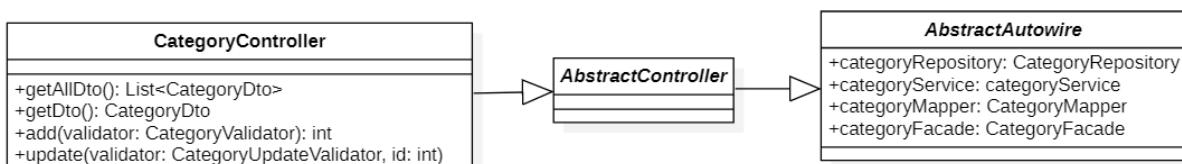


Figure 22 : diagramme des contrôleurs

Dans le schéma ci-dessus tous les contrôleurs héritent de la classe abstraite « *AbstractController* » qui est une classe abstraite et donc vide. Elle a été prévue dans le cas où il faudrait factoriser du code. Celle-ci va aussi hériter de la classe « *AbstractAutowire* » qui, comme vu précédemment, permet d'avoir accès aux Mappers, aux Facades, aux Services et aux Repositories.

L'annotation « *PreAuthorize* » permet de filtrer uniquement les utilisateurs authentifiés. De plus il existe un paramètre nommé « *hasRole* » qui filtre l'accès à la ressource en fonction du rôle.

```
@PutMapping("update/{id}")
@PreAuthorize("hasRole('OWNER') or hasRole('USER')")
public void update(@Valid @RequestBody CValidator validator, @PathVariable("id") int id) {
    categoryService.update(validator, id);
}
```

Code 4 : méthode possédant une annotation « *PreAuthorize* »

7. Les services

Les Services sont le cœur de l'API, ceux-ci appellent les Repositories, les Mappers, les Façades, les Validators afin d'exécuter diverses opérations.

Dans le schéma ci-dessous il y a la classe « *CategoryService* » qui va hériter de la classe abstraite « *AbstractService* ». Ensuite le générique « *T* » dans le schéma représente l'entité qui est passée en paramètre. Celle-ci peut être par exemple la classe « *Category* ».

Cette classe abstraite est composée de 4 méthodes :

- La méthode *get()* qui permet de récupérer l'entités ciblée via un id.
- La méthode *getAll()* qui permet de récupérer toute les entités demandées.
- La méthode *update()* qui permet de vérifier si les données à modifier existent.
- La méthode *delete()* qui permet de supprimer une entité ciblée grâce à un id.

Cette même classe va hériter de la classe abstraite « *AbstractAutowire* »



Figure 23 : diagramme des services

La classe « *CategoryService* » est juste un exemple.

Dans le code ci-dessous la méthode « *getAll()* » se situe dans la classe abstraite « *AbstractService* ». Cette méthode récupère le repository grâce au « *this.getClass()* », cette instruction permet de récupérer l'enfant qui est dans cet exemple « *CategoryService* ». Ensuite « *InitRepository.get()* » permet de faire appel à la méthode se situant dans la figure ci-dessus. Cette méthode permet de récupérer le repository en rapport avec « *CategoryService* ». Par la suite il est donc possible de faire les opérations vues précédemment.

Chaque Repository est retrouvable grâce au nom du service. C'est pour cela que la méthode « *init()* » existe, celle-ci permet de définir la Map qui reprend le nom des Services avec le Repository qui lui est associé.

```

public List<I> getAll() {
    AbstractRepository repository = InitRepository.get(this.getClass());
    if (repository.findAll().isEmpty()) {
        throw new ResponseStatusException(HttpStatus.BAD_REQUEST, "Error");
    } else {
        return (List<I>) repository.findAll();
    }
}

```

Code 5 : méthode getAll() de la classe abstraite « AbstractService »

```

public static Map<String, AbstractRepository> repositoryMap = new HashMap<>();

@PostConstruct
public void init() {
    repositoryMap.put("Category", categoryRepository);
}

public static <T>T get(Class c) {
    return (T) repositoryMap.get(c.getSimpleName().replace("Service", ""));
}

```

Code 6 : attribut et méthode permettant de récupérer les repositories

8. Les mapeurs

Les Mappers permettent de créer et modifier les entités en utilisant les façades. De plus ils permettent de convertir des entités en Dto.

Dans le schéma ci-dessous il y a la classe « CategoryMapper » qui hérite de la classe abstraite « AbstractMapper ». La classe abstraite a été créée dans le cas où il faudrait factoriser du code.

Cette classe abstraite est composée de 4 méthodes :

- La méthode getView() qui permet de récupérer le Dto de vue pour la catégorie.
- La méthode getAllView() qui permet de récupérer toute les Dto de vue pour la catégorie.
- La méthode update() qui permet de modifier les entités via l'appel d'une façade.
- La méthode set() qui permet de créer des entités via la Façade catégorie qui prend en paramètre les différentes valeurs récupérées depuis le Validator.

Cette même classe va hériter de la classe abstraite « AbstractAutowire » qui permet l'accès aux Repositories, aux Facades et aux Validators.

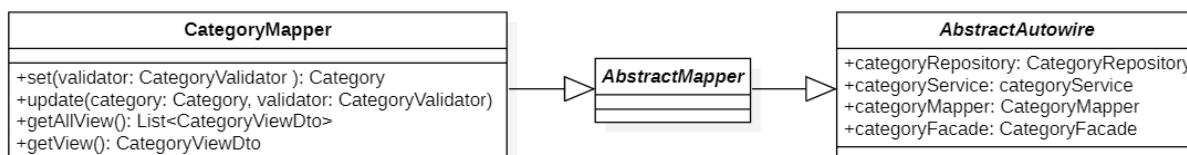


Figure 24 : diagramme des mapeurs

9. Les mails

Les mails sont envoyés en HTML. Ceux-ci sont envoyés dans le cadre de quatre opérations effectuées par l'utilisateur. Celles-ci sont citées ci-dessous :

- La confirmation de son compte lors de la création de celui-ci.
- Lorsque l'utilisateur demande la suppression de son compte.

- Lors d'une demande de changement de mot de passe.
- Lorsque l'utilisateur veut se connecter, le code de double authentification est envoyé par mail.

Les quatre opérations citées ci-dessus sont représentées par l'énumération qui est composée de quatre attributs.

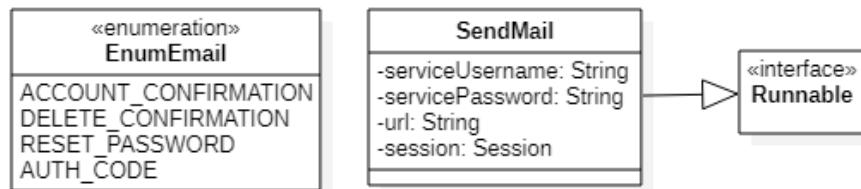


Figure 25 : diagramme des mails

La classe « *SendMail* » est composée d'attributs qui permettent de créer un objet qui est utilisé dans l'envoi de mails.

Dans ces attributs, il y en a quatre importantes cités ci-dessous :

- serviceUsername qui est l'adresse mail utilisée pour envoyer les mails.
- servicePassword qui est le mot de passe du compte google utilisé pour envoyer les mails.
- url qui sert à rediriger l'utilisateur vers le front-end.
- session est un objet de Spring Boot permettant de faire de l'authentification.

Ensuite, la classe implémente l'interface « *Runnable* » afin de créer un thread permettant d'envoyer les mails. Étant donné que l'envoi de mails est une opération assez longue, il été nécessaire d'ajouter un thread pour lui déléguer les envois.

Le code ci-dessous représente la construction de l'objet *SendEmail* qui permet l'envoi des mails.

```

new SendEmail(EnumEmail.DELETE_CONFIRMATION, userEmail, userUsername, userToken);
  
```

Code 7 : instantiation de la classe « SendEmail »

Si l'envoi du mail est un échec, il n'y a aucun moyen de réagir à cet échec. Si l'utilisateur n'a pas reçu le mail, il est tout de même capable de redemander un mail.

10. Le stockage de fichiers

Tout d'abord, l'utilisateur va pouvoir upload des images dans le back-end. Ensuite, lorsque le front-end demande les images via l'url d'entrée, les images sont envoyées grâce à la classe « *FileSystemRessources* » qui permet au front-end de télécharger les images pour ensuite les afficher.

11. La recherche

La recherche va permettre l'exploration des pages.

Deux types de recherche sont possibles :

- La recherche « exact »
- La recherche prenant en compte les fautes d'orthographe

Dans les 2 cas, il faut utiliser un tableau de string.

Note importante : les virgules, les pronoms et les conjonctions sont ignorés. Une recherche composée de plusieurs mots, comme « menthe poivrée », est possible. Dans ce cas le tableau va être composé de deux string pour permettre la recherche des deux mots. La même méthode est utilisée lors de la récupération des titres de page.

Ensuite, une comparaison est faite entre le tableau des titres et le tableau des mots introduits par l'utilisateur. S'il y a une égalité entre les deux chaînes de caractères, la page est récupérée pour être ensuite ajoutée dans un tableau. Une fois la recherche finie le tableau est envoyé au front-end.

```
for (Page page : pages) {  
    String[] titlePageSplited = page.getTitle().toLowerCase().replace(", ", "").split(" ");  
    for (String pageWord : titlePageSplited)  
        for (String searchWord : searchSplited)  
            if (pageWord.equals(searchWord))  
                //Egaliter
```

Code 8 : algorithme simplifier de la recherche exacte

Il y a une évolution dans la recherche avec fautes d'orthographies. En effet, les différents mots de l'utilisateur et les titres de page sont séparés par groupes de trois lettres comme présenté dans la figure ci-dessous. La méthode « *Utils.get3String()* » permet de réaliser cette opération. Prenons par exemple une recherche avec le mot « manthe » et un titre de page qui est « menthe », les mots sont composés en quatre groupes de trois mots qui sont ceux de la figure ci-dessous. Le premier groupe « man » est comparé à « men » et si jamais le résultat est le même la page est ajoutée dans le tableau, mais si ce n'est pas le cas, la méthode va continuer en cherchant « man » dans « ent », « nth » et « the » sans résultat. Il faut donc aller jusque « nth » pour trouver une égalité et donc ajouter la page dans le tableau.

```
for (Page page : pages) {  
    String[] titlePageSplited = page.getTitle().toLowerCase().replace(", ", "").split(" ");  
    for (String pageWord : titlePageSplited)  
        for (String searchWord : searchSplited)  
            for (String pageWord3 : Utils.get3String(pageWord))  
                for (String searchWord3 : Utils.get3String(searchWord))  
                    if (pageWord3.equals(searchWord3))  
                        //Egaliter
```

Code 9 : algorithme simplifier de la recherche avec fautes

```
Menthe -> ["men", "ent", "nth", "the"]  
Manthe -> ["man", "ant", "nth", "the"]
```

Code 10 : exemple de décomposition d'un mot avec la méthode « *Utils.get3String()* »

Réalisation du Front-end

La réalisation du front-end est également divisée en deux corps de travail bien distincts : la création des différents services et la création des différentes pages ou composants. L'ensemble de ces réalisations forme le résultat final de l'application web.

1. Les services

Plusieurs services de base ont été créés, chacun d'entre eux hérite d'une classe abstraite qui permet de stocker l'url, mais également les trois méthodes de récupération d'informations utilisateurs.

```
export class AbstractService {

    protected baseUrl = 'http://127.0.0.1:8080/api/v1/';

    constructor(protected http: HttpClient) { }

    protected getUserToken(): string {
        return JSON.parse(localStorage.getItem('currentUser')).token;
    }
    protected getUserId(): string {
        return JSON.parse(localStorage.getItem('currentUser')).id;
    }
    authenticateUser(user: string, password: string): string {
        return 'Basic ' + btoa(user + ':' + password);
    }
}
```

Code 11 : la classe abstraite « AbstractService »

Pour chaque service proposé, on retrouve plusieurs méthodes qui sont en réalité les requête à l'API. Pour ce faire l'emploi « http » est requis, ce qui permet de faire ces appels. De plus l'ajoute d'un header est réalisée avec une clé « Autorisation » possédant comme valeur le token propre de l'utilisateur actuellement connecté. Cette méthode renvoie un « *Observable* » comme réponse à l'API. Il ne reste plus alors qu'à s'abonner à la réponse pour obtenir un résultat. Voir code ci-dessous. Dans cet exemple, « *MyTaskValue* » correspond à la liste des tâches de l'utilisateur actuel.

```
public getAllMyTask(userId: string): Observable<any> {
    return this.http.get<any>(this.baseUrl + '?involvedUser=' + userId,
{headers : new HttpHeaders().set('Authorization', this.getUserToken())});
}
this.tasksService.getAllMyTask(this.userService.getUserFromLocalStorage().id).subscribe(myT
askValue =>
});
```

Code 12 : méthode utilisant un service pour récupérer des données

2. Ergonomie

Premièrement l'application web est responsive. En effet, les utilisateurs peuvent aller sur leur téléphone en ayant un visuel qui reste agréable et facilement utilisable.

Dans l'application web il existe 2 thèmes de couleurs. Il y a le thème blanc qui va être un thème basé sur des couleurs plus claires. Ensuite, il y a le thème sombre qui quant à lui est basé sur des couleurs plus sobres. Lorsque le thème est défini, il l'est pour tous les appareils où l'utilisateur se connecte.

Les cookies permettent à l'utilisateur de ne pas devoir se reconnecter à chaque fois que celui-ci veut utiliser l'application web.

Enfin, La barre de recherche propose des résultats lorsque l'utilisateur commence sa recherche. Cela permet à un utilisateur d'avoir un résultat de recherche plus proche.

3. Les pages

Dans un premier temps, on retrouve la page « Home » qui représente la page d'index de l'application. Celle-ci est décomposée en différentes parties. On observe le volet de gauche qui permet la recherche, soit par tags ou par mots clés. Cette partie reprend également les compositions des différentes catégories et sous-catégories. Ensuite, la partie centrale permettant l'affichage des différentes pages liées à la catégorie précédemment sélectionnée. Enfin le volet au-dessus contient la barre de navigation. Celle-ci permet également de faire une recherche mais aussi de voyager parmi les différentes pages de l'application.

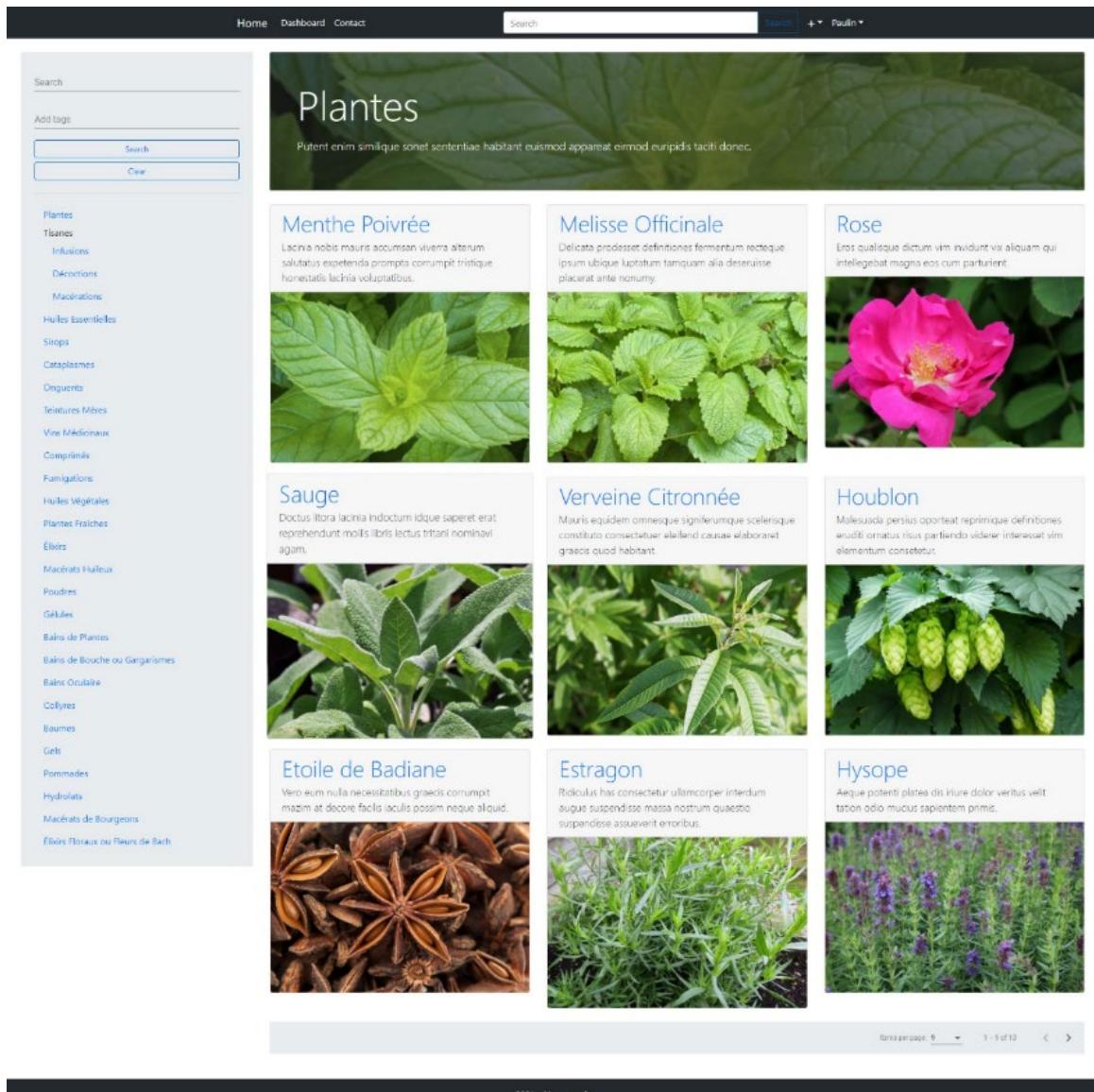


Figure 26 : visuel de la page home

Dans la page Home il faut faire dans un premier temps 3 appels vers le back-end depuis les services du front-end. Après avoir fait ces appels, le back-end enverra uniquement les données nécessaires pour récupérer les tags, les catégories et les pages reliées aux catégories. Par la suite, si un utilisateur appuie sur une catégorie il va envoyer un nouvel appel vers le back-end afin de rafraîchir les données pour qu'elles correspondent à la catégorie demandée.

Toutes les données qui sont récupérées sont sous forme JSON encapsulées dans un tableau. Lors de la récupération de celles-ci, elles sont introduites dans un tableau. Ensuite, dans la partie html du component, il y a plusieurs boucles for qui permettent d'afficher le contenu des différents tableaux.

3.1 Section catégories



Tout d'abord, au sein de cette section on retrouve deux barres de recherches qui servent respectivement à chercher une page parmi la catégorie ou des tags reliés aux pages.

La première barre de recherche est celle qui permet de rechercher les pages. Celle-ci fonctionne aussi bien avec une recherche par mot « exact » ou par mot erroné. Elle utilise le même algorithme déjà vu précédemment.

Ensuite, la deuxième barre de recherche par tags fonctionne sur base de l'algorithme de recherche par mot « exact ». L'utilisateur en cliquant sur la barre va avoir une liste déroulante lui permettant de choisir quel tag il souhaite chercher. Lorsqu'il a choisi un tag celui-ci va disparaître de la liste.

Le bouton « Clear » permet de recommencer une recherche en effet, lorsque l'utilisateur à appuyer sur le bouton « Search » les pages reliées sont affichées. Lors de l'appui sur le bouton « Clear » les différentes pages précédemment affichées sont supprimées pour faire place au résultat initial.

Particularité pour l'affichage, en effet, Dans ce cas si « Tisanes » est une catégorie mère elle est accompagnée de sous catégories qui sont affichées sous forme d'arborescence.

Figure 27 : visuel des sections

3.2 Résultat de la recherche

Le résultat de la recherche est affiché dans la partie centrale de la page Home (voir figure 26). Si la recherche est effectuée avec des tags, elle va afficher le tag sous les descriptions des pages sans pointer une page en particulier. Si la recherche est faite par mots clés, celle-ci va afficher la page recherchée. Si les deux sont combinés, le résultat va être celui de la figure 28.



Figure 28 : visuel de la recherche

3.3 Barre de navigation



Figure 29 : visuel de la barre de navigation

La barre de navigation est notamment composée de liens amenant vers les différentes pages du site. Elle est aussi composée d'une barre de recherche.

Tout d'abord, « Home » va permettre d'arriver dans la page de la figure 26.

Ensuite, « Dashboard » permet de se rendre dans le dashboard de l'utilisateur, mettant à disposition les tags et les images qu'il a créées ou qui sont en cours de création.

Après, « Contact » permet de rediriger l'utilisateur vers la page de ses tickets pour parler avec les administrateurs.

Point important qu'offre la barre de navigation est la présence d'une barre de recherche. Celle-ci fonctionne grâce au même algorithme déjà vu précédemment. Il existe néanmoins une différence qui réside dans le fait qu'elle va afficher les résultats de pages vis-à-vis de l'avancement de l'écriture des mots dans celle-ci. Par exemple, si l'utilisateur commence à écrire menthe à partir du moment où il aura tapé le m la barre va afficher des résultats parmi les pages existantes commençant par « m ».

Add a page	Le « + » permet d'afficher la liste déroulante ci-dessous :
Add a tag	Cette liste déroulante est composée de plusieurs choix de pages permettant aux utilisateurs de créer du contenu. Attention, tous ce qui est en dessous de « Add a image » est accessible seulement pour les administrateurs.
Add a image	
List of pages	
List of tags	
List of images	Les pages commencent par « List » sont composées de tableaux reprenant par exemple des pages ou des images. Sur la figure 30 il est facile de constater qu'il existe 8 pages permettant de gérer les différentes créations des utilisateurs ainsi que les créations des administrateurs.
List of category	
List of tagTypes	
List of paragraphTypes	
List of parapageTypes	
List of paratagTypes	
Add a category	Les derniers « Add X » permettent aux administrateurs d'ajouter des éléments permettant aux utilisateurs de développer leurs pages en profondeur en leur ajoutant de nouveaux types de contenus. Cela permet de combler les sections dans la création des pages.
Add a tagType	
Add a paragraphType	
Add a parapageType	
Add a paratagType	

Figure 30 : visuel de la liste recherche

L'appui sur le nom d'utilisateur « Paulin » permet d'afficher la liste déroulante ci-dessous :

Dashboard	Cette liste est composée de « Dashboard » qui redirige une fois de plus vers le Dashboard de l'utilisateur.
Favorite	« Favorite » qui renvoie vers la page des favoris de l'utilisateur.
Settings	« Settings » qui renvoient vers les paramètres de l'utilisateur. Ceux-ci lui permettent de par exemple réinitialiser son mot de passe.
Tickets	« Tickets » qui permet d'afficher tous les tickets créé par l'utilisateur.
Log out	« Log out » qui permet tout simplement à l'utilisateur de se déconnecter du site.
Admin dashboard	
Admin tickets	

Figure 31 : visuel de la liste déroulante bis

Pour l'administrateur, il existe 2 autres pages. « Admin Dashboard » qui est une page où l'administrateur voit tous les utilisateurs du site. Il peut modifier leur rôle au sein du site ainsi que modifier leur email. « Admin tickets » redirige vers les tickets que les utilisateurs créent, il peut ainsi aller répondre aux demandes des utilisateurs.

3.4 Page de recherche

Dans cet exemple, il est question d'une recherche avec le mot « manthe », dans cette recherche, il est facile de remarquer qu'une faute s'est facilement glissée en effet, le e de menthe c'est transformer en a pourtant dans la sélection de pages celle en rapport avec la menthe s'y retrouvent bien. Il est possible de cliquer sur le nom de la page pour ainsi se retrouver sur celle-ci ou alors il est possible de cliquer sur la photo avec le nom de la catégorie pour se retrouver dans celle-ci.

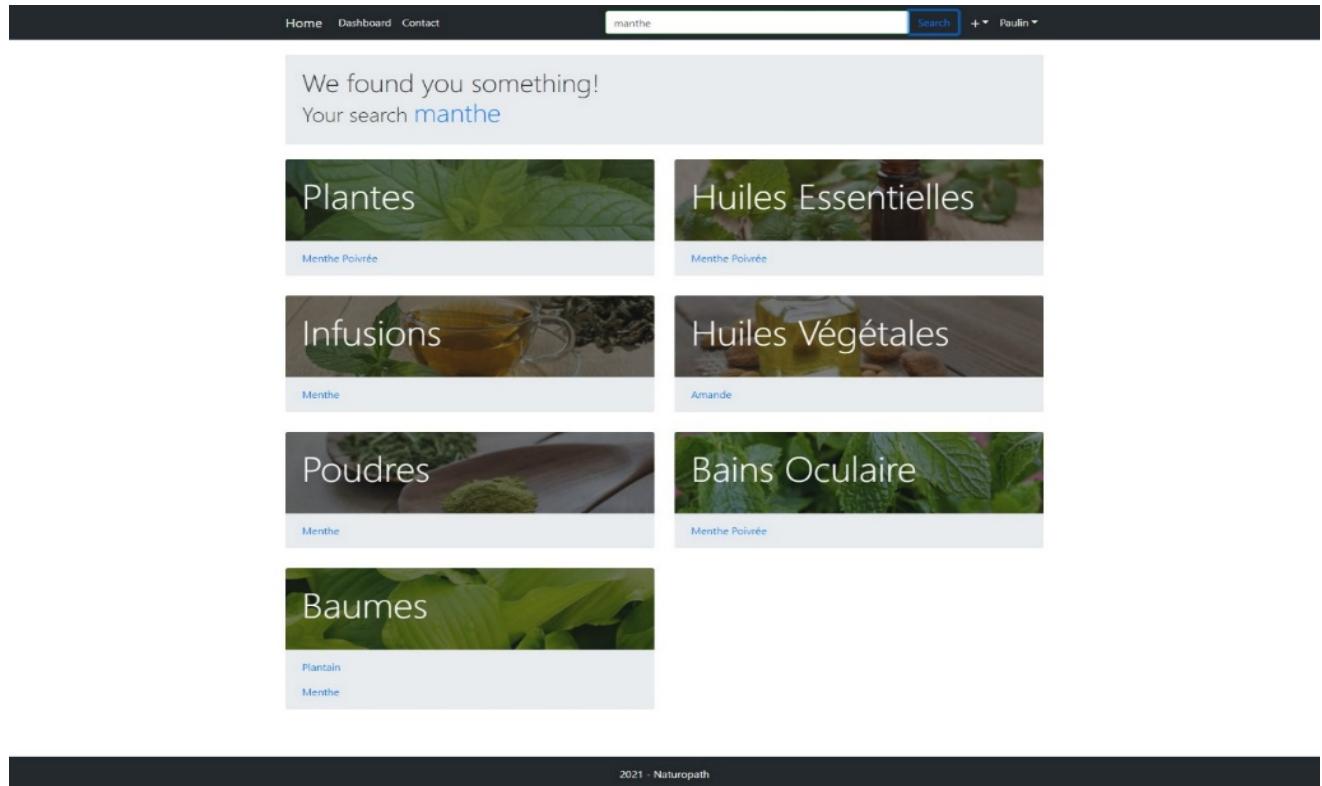


Figure 32 : visuel d'une page de recherche

3.5 Pages des favoris

Dans cette page, il est possible de retrouver toutes les pages que l'utilisateur a aimé auparavant.

Il sait donc retrouver les pages qu'il préfère au sein du site. Il peut appuyer sur le nom de la page pour être directement rediriger vers celle-ci ou alors il peut décider d'appuyer sur le nom de la catégorie pour être aussi rediriger vers celle-ci.

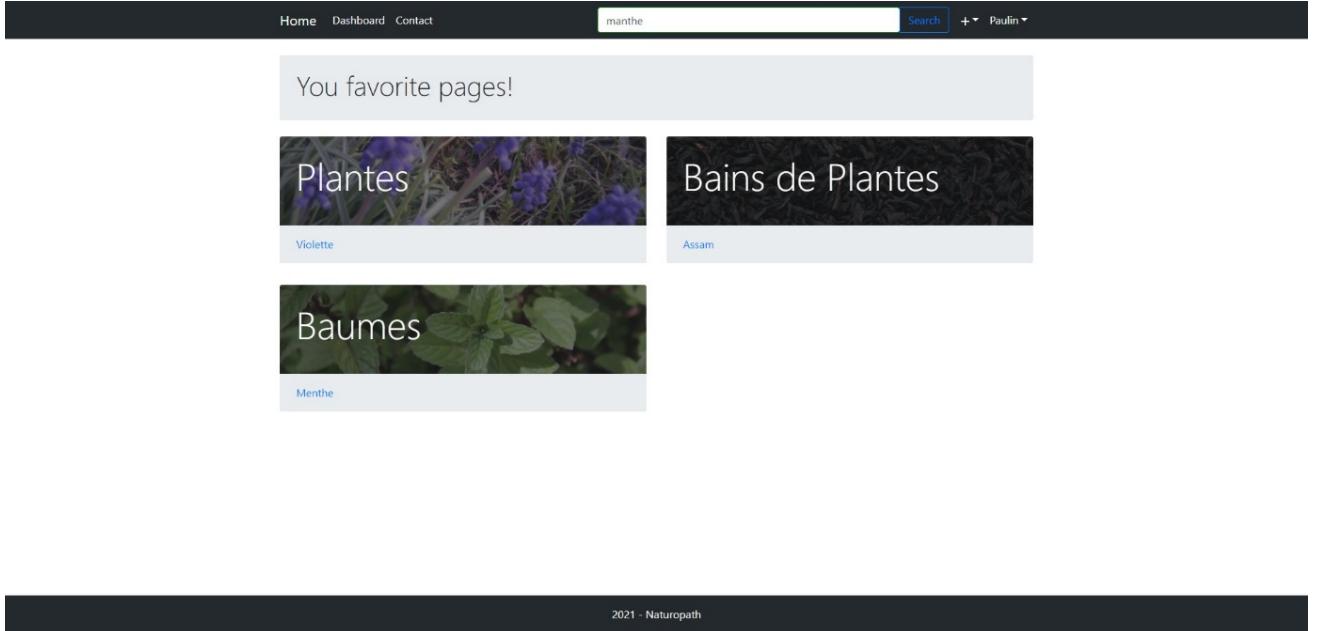


Figure 33 : visuel de la page des favoris

3.6 Page d'administration des pages

Cette page permet d'afficher toutes les pages créées par les utilisateurs. Dans cette page, il y a un tableau reprenant trois données ainsi qu'un bouton qui permet l'accès aux pages d'édition des différentes pages.

Tout d'abord, la première donnée importante est l'Id de la page. Grâce à celui-ci la page peut être retrouvée après l'appui sur le bouton « Edit » en effet, l'Id étant passé dans l'url de redirection, il est récupéré lors de l'ouverture de la page afin de faire un appel vers le back-end pour récupérer les différentes données qui forment la page.

Ensuite, la deuxième valeur du tableau représente l'état d'avancement de la page. Il permet de définir si la page est en cours de validation, validée ou encore si elle est simplement en cours de création.

Enfin, la dernière donnée est le titre de la page, il permet de retrouver la page via son titre. Cette donnée n'est pas réutilisée pour faire un appel au back-end, elle est juste là pour permettre à l'administrateur de retrouver la page qu'il aimera modifier.

Administration			
Id	State	Title	Action
2028	VALIDATED	Menthe Poivrée	<button>Edit</button>
2036	VALIDATED	Mélisse	<button>Edit</button>
2044	VALIDATED	Rose	<button>Edit</button>
2050	VALIDATED	Sauge	<button>Edit</button>
2054	VALIDATED	Verveine Citronnée	<button>Edit</button>
2058	VALIDATED	Houblon	<button>Edit</button>
2066	VALIDATED	Etoile de Badiane	<button>Edit</button>
2074	VALIDATED	Estragon	<button>Edit</button>
2080	VALIDATED	Hypsope	<button>Edit</button>
2087	VALIDATED	Lierre Terrestre	<button>Edit</button>

Items per page: 10 | < < > >| Page: 1 - 10 of 98

Figure 34 : visuel de la page d'administration

Toutes les autres pages d'administration qui touche à l'édition de pages, de tags, d'images, de catégories, de tagTypes, de paragraphesTypes, de parapagesTypes et de paratagType sont pratiquement les mêmes. En effet, il existe une différence :

Pour les pages, les images et les tags ceux-ci étant soumis à une étape de validation, il est donc nécessaire d'afficher l'état de validation dans le tableau d'administration. Cependant les tagTypes, les paragraphesTypes, les parapagesTypes et les paratagTypes sont créés par les administrateurs, ne sont donc pas soumis à la validation. C'est dans ce but que le tableau ne comprendra uniquement les 3 colonnes permettant d'afficher l'id, le titre et le bouton d'édition. Le principe d'envoyer l'id dans l'url afin de le récupérer les données de l'objet est réutilisé pour tous les tableaux d'administration.

Toutes les pages d'administration sont accessibles depuis la barre de navigation en appuyant sur le « + » (voir figure 29).

3.7 Page des tickets

La page des tickets permet aux utilisateurs d'envoyer leurs réclamations ou demander de l'aide aux administrateurs.

Cette page est simplement composée d'un champ texte permettant d'introduire un titre pour le ticket. Celui-ci est important, en effet, c'est l'une des premières données visuelles que l'administrateur va voir, il faut donc que le titre soit clair, net et précis.

Created a new ticket

Subject
What is the subject of your problem?

Message
what is your problem in details?

Send

2021 - Naturaphis

Figure 35 : visuel de la page tickets

Ensuite, il y a un autre champ texte permettant d'écrire le contenu de la requête de l'utilisateur. Une fois le contenu écrit, il est possible d'envoyer le ticket en un appui sur le bouton « Send »

Par la suite, l'utilisateur verra sa page se rafraîchir afin de laisser apparaître un tableau avec son ticket. Il peut appuyer sur l'Id du ticket afin de retrouver la conversation avec l'administrateur qui va s'occuper de sa demande.

My tickets

A problem? [create a new ticket](#)

ID	Subject	Last message from	Created at	Open
2751	Réclamation	Paulin you	09 June 2021 - 16:27	Open

Figure 36 : visuel de l'envoi du ticket

L'administrateur peut aller voir les différents tickets dans la page administrateur. Celle-ci est similaire à celle pour les utilisateurs. Lorsque l'administrateur va appuyer sur le numéro d'Id, celui-ci est récupéré pour être ensuite utilisé dans l'appel vers le back-end afin de recevoir toutes les données nécessaires pour l'affichage de la page de ticket.

Lorsque le ticket est ouvert l'administrateur peut envoyer un message à l'utilisateur. Il peut décider de fermer le ticket s'il juge que celui n'a plus lieu d'être ou s'il a tout simplement fini de régler le problème de l'utilisateur.

[Back to my ticket](#)

Ticket 2751

Type your message here...

[Send](#)[Close](#)[Close ticket](#) Paulin

09 June 2021 - 16:38

Bien sûr, quelle catégorie voulez-vous utiliser pour la page ?

 Paulin

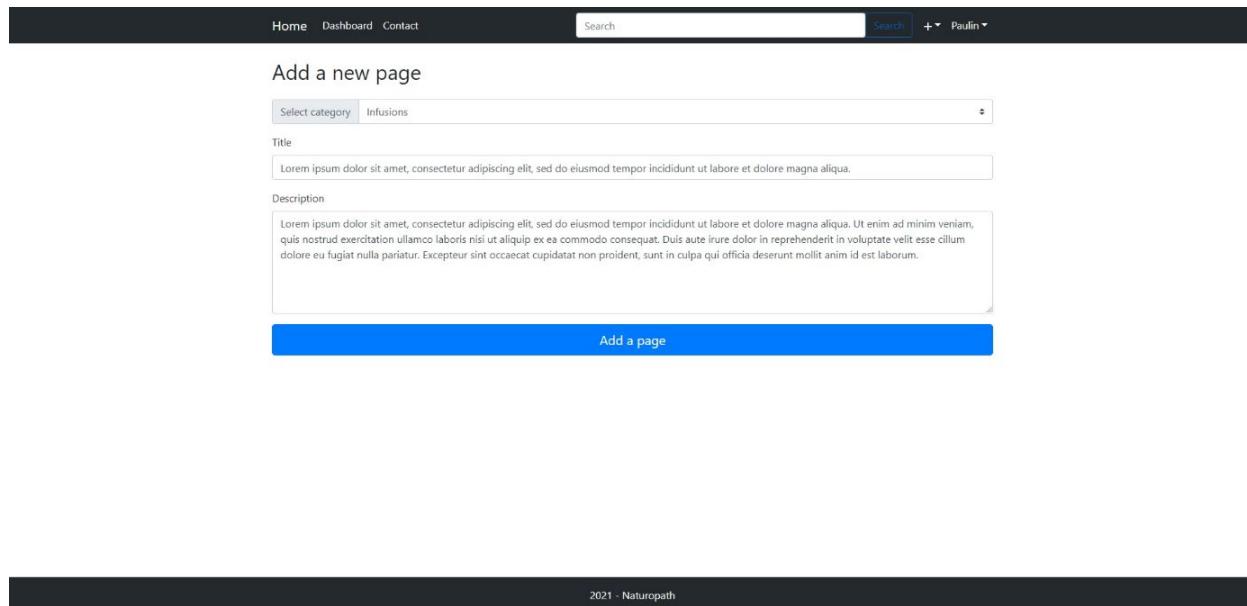
09 June 2021 - 16:27

J'aimerais avoir une nouvelle page

Figure 37 : visuel du suivi du ticket par l'administrateur

3.8 Ajout de pages

Dans un premier temps, pour pouvoir ajouter une page, il faut être connecté. Les utilisateurs peuvent consulter les pages en n'étant pas connectés mais ils ne peuvent pas ajouter de pages. Ensuite, il faut construire la page dans sa version la plus simple. Celle-ci est composée d'un titre et d'une description ainsi qu'une catégorie. La définition de la catégorie est importante, en effet, celle-ci définit si la page va par exemple parler d'une plante ou alors de Vins médicinaux. De plus, les sections changent vis-à-vis des différentes catégories. Effectivement, une page sur les plantes contiendra par exemple un nom latin et une famille tandis qu'une tisane est composée d'une section préparation. L'utilisateur peut choisir la catégorie grâce à une liste déroulante. Lorsque la catégorie a été sélectionnée, que le titre est assez long et que la description est assez longue l'utilisateur peut cliquer sur le bouton « add a page » afin de créer sa page. Il est directement redirigé vers la page dédiée à la création des différentes sections.



2021 - Naturopath

Figure 38 : visuel de l'ajout d'une page

La figure 39 représente le visuel de la page de création validation qui est aussi la page d'édition. Comme présenté auparavant, il y a plusieurs sections qui compose une page. Dans cette image il y a plusieurs informations importantes. Tout d'abord il y a une indication sur le type de section. Par exemple, le premier est un type page composé de Title, description, image. Ceci reprend ce qui a été créé auparavant en y ajoutant une sélection d'images. Ensuite, il y a différents types comme Paragraph, Paratag et Parapage qui représente d'autres sections. A côté des noms de types, il y a l'état de validation des différentes sections créées par l'utilisateur.

Section	Status
Title, description and image	Page Draft
Préparation	Paragraph Draft
Conservation	Paragraph Draft
Précautions	Paragraph Draft
Utilisations	Paragraph Draft
Contre-indications	Paragraph Draft
Indications	Paragraph Draft
Propriétés	Paragraph Draft
Ingredients	Paragraph Draft

Figure 39 : visuel de l'édition d'une page

Il existe donc la section page, celle-ci se voit ajoutée une image. Cette image est choisie dans le catalogue d'images dont l'api dispose. Il faut savoir qu'il existe aussi une page permettant l'ajout d'une image par les utilisateurs. Ces images sont soumises au même système de validation que les autres sections. Dans la figure 40 on constate deux boutons qui sont « Save Draft » et « Validation ». Le premier permet à l'utilisateur de sauvegarder son travail et ses différentes modifications. A côté du bouton « Select an image », il y a un champ texte qui contient l'Id de l'image. Celui-ci permet de savoir quelle est l'image sélectionnée.

Title, description and image

Page Draft

Title:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Description:
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Image:
Select an image 1842

Save Draft Validation

Type your message here... Send

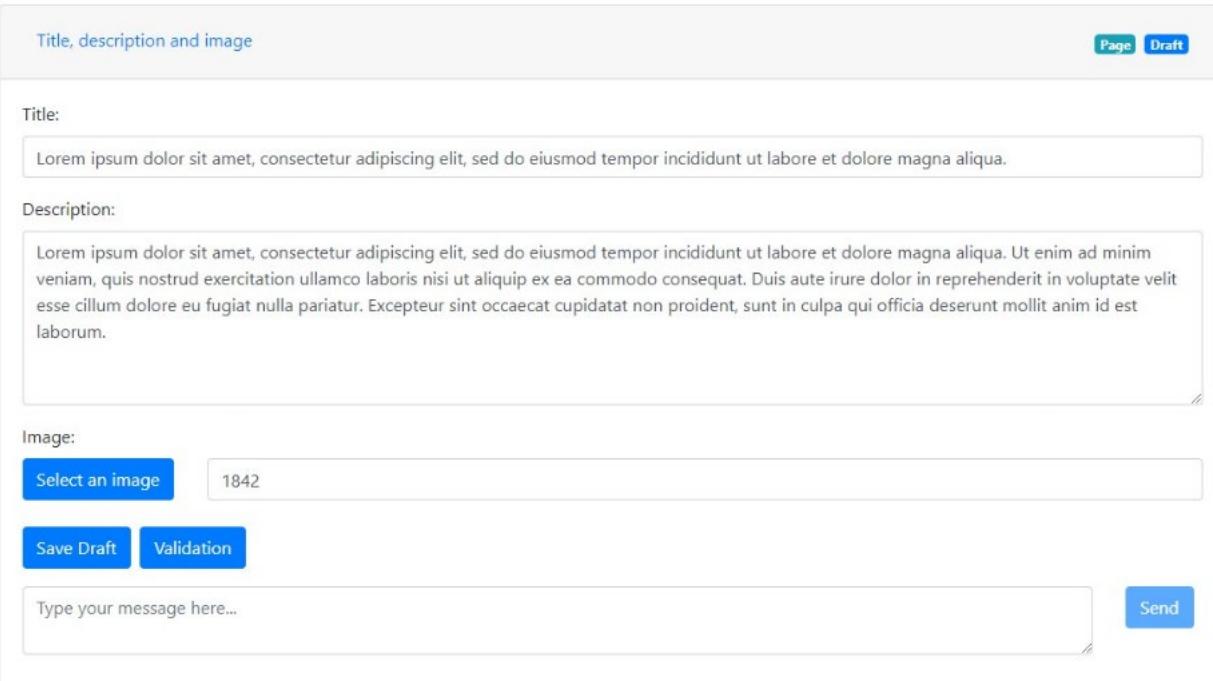


Figure 40 : visuel de l'ajout de pages

Il est aussi possible que l'utilisateur puisse communiquer avec les administrateurs grâce à la boîte de dialogue en dessous des deux boutons. Les messages s'échangeront de la même manière que pour les tickets.

Le deuxième bouton sur la figure 40 permet d'enclencher une procédure de validation. Celle-ci va soumettre le travail fourni par l'utilisateur aux votes des administrateurs. Une fois le travail soumis, Des changements sont effectués dans la section.

Tout d'abord, le statut de la section est passé de Draft à Validating cela signifie que la section est bien soumise aux votes des administrateurs. Ensuite, pour que la section soit validée, il faut que les cinq votes possibles soit au vert. Pour voter en faveur de la section, il suffit d'appuyer sur le bouton « Favour ». Si un administrateur veut dire qu'il est contre la section créée par l'utilisateur, il est obligé de faire un message expliquant sa décision pour être capable ensuite d'appuyer sur « Against ». Dans la figure 41 il y a pour le moment un vote pour et un vote contre, Un message a bien été écrit par l'administrateur qui a refusé que la section soit publiée. Une fois que les 5 votes sont faits, l'utilisateur peut recommencer la section en se basant sur les commentaires des administrateurs.

Title, description and image

Page Validating

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



1/5 1/5

Favour

Type your message here... Send Against

 Paulin 09 June 2021 - 19:13
Photo ne représentant pas la plante dont vous voulez parler

Figure 41 : visuel de la validation d'un innerPage

Il existe pour toutes les sections de la page un système de versioning. En effet, toutes les versions qu'elles soient validées ou non sont enregistrées dans la base de données. Elles sont ensuite accessible dans le front-end via l'appui sur un bouton. La figure 42 représente donc les 2 versions de la section page. Les messages qui sont faits sont aussi enregistrés et donc accessibles dans le système de versioning. Ceci permet à l'utilisateur de pouvoir revoir ses anciennes versions afin d'éventuellement se baser sur celles-ci pour améliorer les différentes sections.

Timeline

 **Paulin** 09 June 2021 - 19:13
version 1 is VALIDATED on 09 June 2021 by Paulin
Paulin, Pauline, Pauline, Pauline have voted.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



 **Paulin** 09 June 2021 - 19:13
version 0 is NOT VALIDATED on 09 June 2021 by Paulin
Paulin, Pauline, Pauline, Pauline have voted.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.



 **Paulin** 09 June 2021 - 19:13
Photo ne représentant pas la plante dont vous voulez parler

Figure 42 : visuel de l'historique d'une page

3.9 ParagraphType

Il existe bien sûr d'autres types de sections. Sur la figure 43 il est question d'un paragraphe, celui-ci est la section la plus basique qu'il est possible de créer. Effectivement, cette section est composée d'un titre et d'une description. Celle-ci étant soumise à la même validation que pour la section page. Il est aussi possible de discuter avec les administrateurs.

The screenshot shows a web-based form for editing a paragraph. At the top left is a blue header bar with the text "Préparation". On the right side of the header are two buttons: "Paragraph" (highlighted in blue) and "Draft". Below the header, there are two input fields: "Title" and "Content", each with a text input area. At the bottom of the form are two blue buttons: "Save Draft" and "Validation". To the right of these buttons is a large text area labeled "Type your message here..." with a "Send" button to its right. The entire form is contained within a light gray border.

Figure 43 : visuel de l'édition d'un paragraph

3.10 ParatagType

Cette section permet de créer un paragraphe en y ajoutant une liste de tags qui fournissent une information visuelle supplémentaire pour l'utilisateur. Pour cette section, il y a l'ajout d'une liste sous forme de drag and drop. Cette liste va être composée des différents tags accessibles sur l'application web. Il y a aussi une deuxième liste permettant de définir les tags que l'utilisateur veut mettre dans son paratag. Le Paratag contient aussi un titre et une description. Les 2 champs texte étant soumis à des règles de longueur de chaînes de caractères.

The screenshot shows a web-based form for editing a paratag. At the top left is a blue header bar with the text "Contre-indications". On the right side of the header are two buttons: "Paratag" (highlighted in blue) and "Draft". Below the header, there are two input fields: "Title" and "Content", each with a text input area. At the bottom of the form are two blue buttons: "Save Draft" and "Validation". To the right of these buttons is a large text area labeled "Type your message here..." with a "Send" button to its right. In the center of the form, there are two columns: "Available" and "Use". The "Available" column contains a list of tags: "Effet laxative/purgatif important", "Femme allaitante", "Enfants de moins de 3 ans", "Femme enceinte", "Inflammation du tractus gastro-intestinal ou des voies biliaires", "Prudence car l'oxalate de calcium est responsable de la toxicité", "Vomissement", and "Nausées". The "Use" column contains a single text input area. The entire form is contained within a light gray border.

Figure 44 : visuel de l'édition d'un paratag

3.11 Parapage

Le dernier type de section utilisable est le Parapage. Cette section permet de créer un paragraphe en y ajoutant une référence vers une autre page. Par exemple dans la figure 45 le nom de la section est « ingrédients » celle-ci est composée du titre, de la description et surtout de la liste d'ingrédients disponibles. Cette liste fonctionne sous forme de drag and drop. Il faut mettre les ingrédients dans la liste de droite afin de faire le lien vers ses ingrédients. Cette section est basée sur la même fonction de validation que les autres sections vues précédemment.

The screenshot shows the 'Ingredients' section of a web application. At the top, there are tabs for 'Parapage' and 'Draft'. Below that, there are fields for 'Title:' and 'Content:'. To the right, there are two columns: 'Available' and 'Use'. The 'Available' column contains a list of plant names: 'Plantes - Menthe Poivrée', 'Plantes - Melisse', 'Plantes - Rose', 'Plantes - Sauge', 'Plantes - Verveine Citronnée', and 'Plantes - Houblon'. The 'Use' column is currently empty.

Figure 45 : visuel de l'édition d'un parapage

Lorsque toutes les sections sont créées et validées, il est maintenant temps de publier la page sur l'application web. Sur la figure 46 le bouton « Publish » vient d'apparaître, il permet donc de publier la page dans la catégorie choisie auparavant. Sur la même figure les états d'avancement sont tous en validated ce qui montre bien que la page a été validée dans son ensemble.

The screenshot shows a confirmation message: 'Everything has been validated you can publish your page!' with a 'Publish' button. Below this, a table lists various sections and their validation status: 'Title, description and image' (Validated), 'Utilisations' (Validated), 'Précautions' (Validated), 'Conservation' (Validated), 'Préparation' (Validated), 'Propriétés' (Validated), 'Indications' (Validated), 'Contre-indications' (Validated), and 'Ingredients' (Validated). The table has 'Section' and 'Status' columns. The status for all sections is 'Validated' with a green background.

Figure 46 : visuel de publication d'une page

3.11 Page d'ajout et d'édition des tags

Cette page permet de créer un tag en lien avec un type de tags. Un tag est composé d'un titre, d'une description et d'un type de tag. Lors de l'appui sur le bouton « Add a tag » le tag est créé ensuite l'utilisateur est redirigé vers la page d'édition et de validation du tag.

2021 - Naturopath

Figure 47 : visuel de la page d'ajout d'un tag

La page d'édition ressemble à la validation d'une section dans la page de création et d'édition d'une page. En effet, les boutons « Save draft » et « Validation » sont de nouveau présent. Ils permettent les mêmes fonctionnalités que ceux déjà expliqués précédemment. Le bouton « View all version » est aussi présent ici. Dans l'ensemble les pages de création et d'édition disposent tous des mêmes fonctionnalités.

2021 - Naturopath

Figure 48 : visuel de la page d'édition d'un tag

3.11 Page d'ajout et d'édition d'images

La figure 49 représente la page d'ajout des images, dans celle-ci, il y a l'obligation de donner un titre et une description à l'image. Il y a un champ permettant de choisir une photo stockée sur le pc de l'utilisateur. Lorsqu'il a choisi la photo celle-ci se retrouve affichée. L'utilisateur n'a plus qu'à appuyer sur le bouton « Add image » afin de rentrer sur la page d'édition de l'image. Celle-ci permet aussi de valider l'image auprès des administrateurs.

The screenshot shows a web-based application interface for adding a new image. At the top, there's a navigation bar with links for Home, Dashboard, and Contact, along with a search bar and a user profile dropdown for 'Paulin'. The main content area has a heading 'Add a new image'. Below it, there are two text input fields: 'Title' containing 'Lorem ipsum dolor sit amet' and 'Description' containing placeholder Latin text. A file input field shows a preview of a green leafy vegetable and the filename 'init-b-1.jpg'. A 'Browse' button is available for selecting more files. A note below the file input says 'The image will be added, you can modify it afterwards'. At the bottom right of the form is a large blue button labeled 'Add a image'.

Figure 49 : visuel de la page d'ajout d'une image

3.12 Page d'ajout et d'édition des catégories

Tout d'abord, lors de la création d'une catégorie il faut choisir si celle-ci est une catégorie qui a pour but d'être parente ou si celle-ci va être attaché à une catégorie parente. Si celle-ci doit être une catégorie parente il faut cocher la petit checkbox avec écrit « Is a parent » si elle est liée à une catégorie il y a la possibilité de choisir la catégorie dans la liste déroulante. La catégorie est aussi composée d'une description et un titre.

The screenshot shows a web-based application interface for adding a new category. At the top, there's a navigation bar with links for Home, Dashboard, and Contact, along with a search bar and a user profile dropdown for 'Paulin'. The main content area has a heading 'Add a new category'. Below it, there are several input fields: a dropdown menu for 'Select category' showing 'No category', a checkbox for 'Is a parent' which is unchecked, a text input field for 'Name' with placeholder text 'Lorem ipsum dolor...', and a text input field for 'Description' with placeholder Latin text. A blue button at the bottom right says 'Add a category'.

Figure 50 : visuel de la page d'ajout d'une catégorie

Si la catégorie est définie comme une catégorie parente l'utilisateur est redirigé vers la page d'édition de celle-ci ou il est capable de modifier le titre et la description.

The screenshot shows a web-based administration interface for managing categories. At the top, there is a dark header bar with a logo, the text 'Home Dashboard Contact', a search bar, and a user dropdown set to 'Paulin'. Below the header, the main content area has a title 'Edit category'. It contains two input fields: 'Name:' with the placeholder 'Lorem ipsum dolor.' and 'Description:' with a larger text area containing placeholder text about labor and dolore magna aliqua. A blue 'Update' button is located at the bottom left of the form. The footer of the page includes the year '2021 - Naturopath'.

Figure 51 : visuel de la page d'édition de catégorie

Par contre, lorsque la catégorie n'est pas définie comme une catégorie parente il est possible de définir les sections qui composeront les pages liées à cette catégorie. Dans la figure 52 il y a 2 listes qui représentent d'une part une liste de sections et dans l'autre les sections choisies pour composer les pages qui débattent de la catégorie. Le type de la section est indiquée afin de permettre aux administrateurs d'agencer les sections vis-à-vis de leur type. Il est possible de les ranger dans l'ordre que l'on veut. Lorsque le choix est fait, l'administrateur peut appuyer sur le bouton « update » afin de modifier la catégorie. Cette page est très importante car elle est le cœur de la création de pages.

The screenshot shows a web application interface for managing categories. At the top, there's a navigation bar with links for Home, Dashboard, Contact, a search bar, and a user profile for Paulin. The main content area is titled 'Edit category'. It has two sections: 'Available' and 'Use'. The 'Available' section lists various items with small blue status boxes containing labels like 'Préparée', 'Préparée', 'Préparée', etc. The 'Use' section lists items with similar status boxes. Below these sections is a large text area containing placeholder text. At the bottom left is an 'Update' button.

Figure 52 : visuel de la page de définition d'une catégorie

3.13 Page des settings

La page des settings est composée de quatre sous-pages en effet, sur la figure 53 En bas de la page il y a deux liens permettant de consulter les deux pages permettant de déterminer le RGPD.

Il y a une sous page étant « Account » qui permet de modifier son nom d'utilisateur ainsi que son email en fournissant le mot de passe de son compte. Une indication surlignée en jaune est présente sur le dessus de la page, celle-ci sert à indiquer que l'utilisateur n'a toujours pas validé son adresse mail. Si l'utilisateur n'a pas reçu le mail de confirmation il lui suffit d'un appui sur le bouton « Resend » a confirmation » pour renvoyer une demande de confirmation. D'autres indications d'erreurs ou de validations peuvent apparaître sur le dessus de la page.

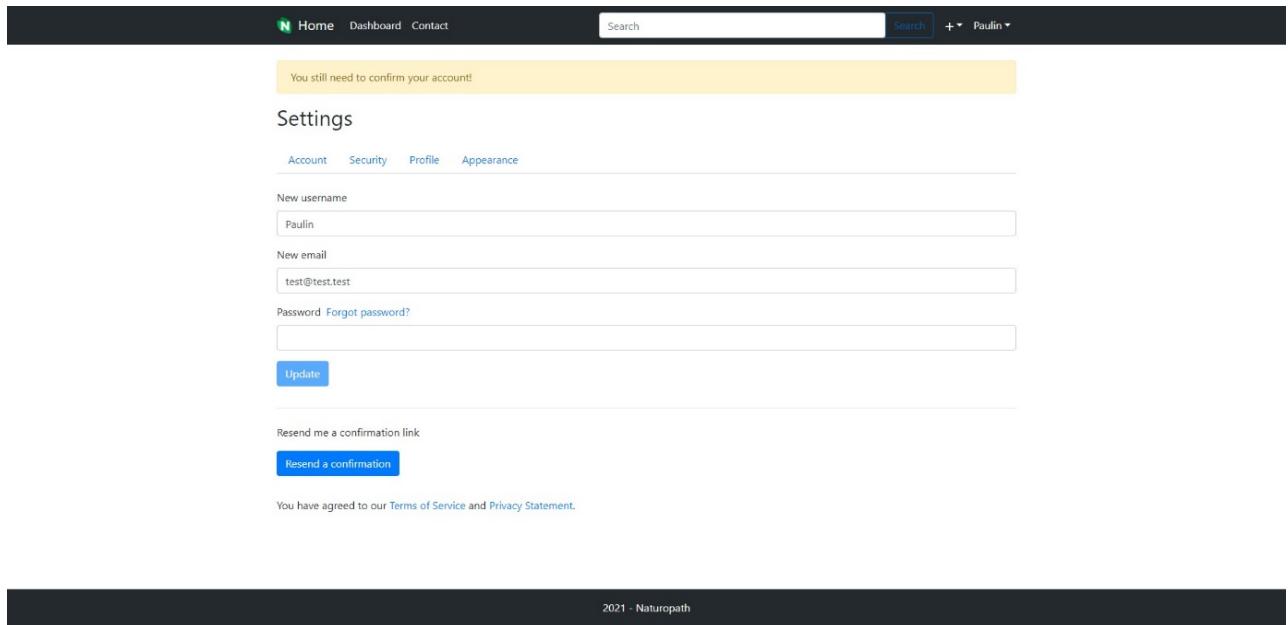


Figure 53 : visuel de la page de setting

La seconde sous page est « Security ». Celle-ci permet dans un premier temps de supprimer son compte. Il faut pour cela introduire son mot de passe. Il est aussi possible de récupérer toutes les données personnelles ainsi que les données introduites sur le site.

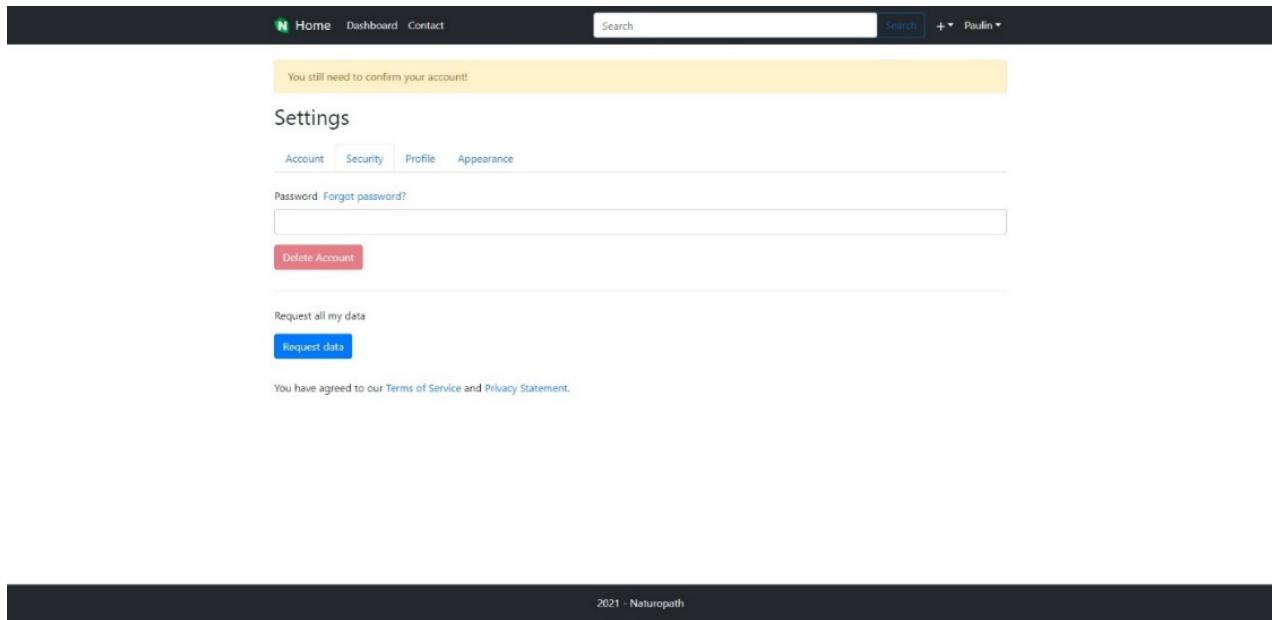


Figure 54 : visuel de la page setting - sécurité

La troisième sous page qui est « Profile » permet à l'utilisateur de changer de nom et de prénom. Il est aussi possible de changer la « profile privacy » du profil de l'utilisateur, il peut décider que celui-ci est un profil privé ou un profil public.

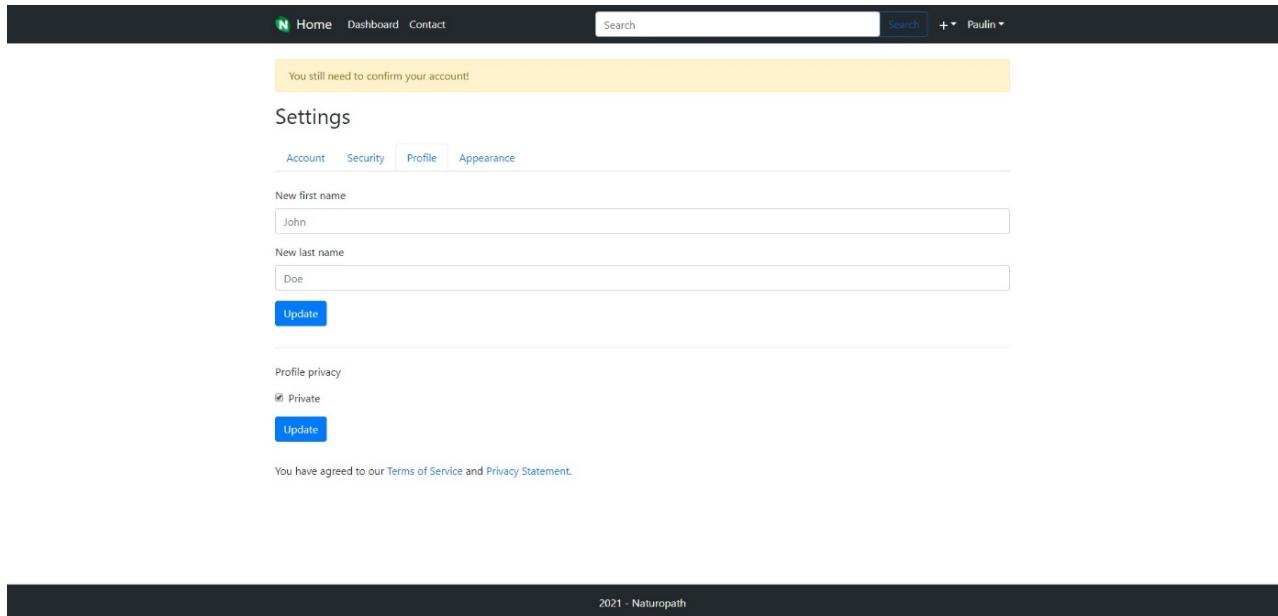


Figure 55 : visuel de la page setting - profile

La dernière sous page est « Apparence », cette sous page permet simplement à l'utilisateur de choisir un thème sombre ou un thème clair lorsque qu'il navigue dans le site.

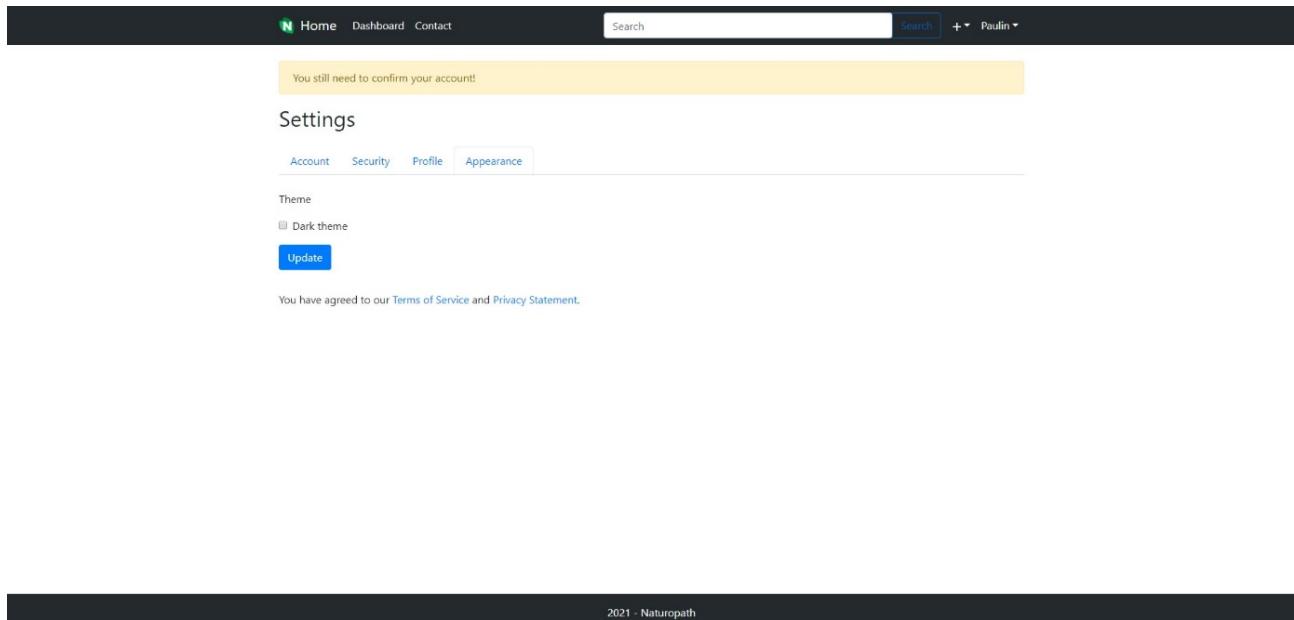


Figure 56 : visuel de la page setting - apparence

3.14 Page des conditions d'utilisations

Voici la page des conditions d'utilisation qui contient plusieurs sections.

The screenshot shows the 'Naturopath Website Terms of Service' page. At the top, there's a navigation bar with links for 'Home', 'Dashboard', and 'Contact', along with a search bar and a user profile dropdown for 'Paulin'. A blue banner at the top states: 'We've updated our Terms of Service on April 8, 2020. | Archived versions'. The main content area is titled '1. Terms' and contains a paragraph about the legal agreement. Below it is '2. Use License' with a note about temporary download rights. A list of prohibited actions follows, including reproduction, modification, and reverse engineering. A note below states that such actions will lead to termination of service. '3. Disclaimer' follows, noting that materials are provided 'as is' and no warranties are made. '4. Limitations' states that Naturopath is not liable for damages. '5. Revisions and Errata' notes that materials may contain errors and may be updated. '6. Links' states that Naturopath is not responsible for linked sites. '7. Site Terms of Use Modifications' states that terms may change. '8. Your Privacy' has a note about reading the privacy policy. '9. Governing Law' states that French law applies. At the bottom, a footer bar reads '2021 - Naturopath'.

Figure 57 : visuel de la page de conditions d'utilisations

4. Visualisation d'une page finie

Dans le haut de la page, il y a tout d'abord une indication de danger, celle-ci sert à avertir que la page contient des informations importantes qui annoncent un danger, il faut donc y faire attention.

Ensuite, il y a la possibilité de mettre un j'aime pour la page. Si l'utilisateur aime la page celle-ci va se retrouver dans la page des likes de l'utilisateur. Cette fonctionnalité permet principalement à l'utilisateur de retrouver les pages qu'il apprécie plus rapidement.

The screenshot shows a page with a red alert box at the top containing the text: 'Be careful this page contains dangers, read them thanks!'. Below this is a grey header bar with the category 'Bains de Plantes' on the left and a 'Like' button on the right. The main content area is partially visible.

Figure 58 : visuel de l'alerte et de la catégorie d'une page fini

En bas de page, il y a un système de messagerie qui permet aux utilisateurs de donner leurs ressentis sur la page. Ils peuvent aussi communiquer entre eux pour débattre de la page.

Discussion

- Paulin 10 June 2021 - 20:46
Merci d'avoir mis que certaines choses étaient dangereuses
- Paulin 10 June 2021 - 20:46
Super !
- Paulin 10 June 2021 - 20:46
Merci pour les renseignements

Items per page: 10 ▾ 1 – 3 of 3 < >

Type your message here...

Send

Figure 59 : visuel de la messagerie d'une page

Il y a aussi l'image qui est réaffichée avec le nom de la page et la description de celle-ci. Une indication informant de l'utilisateur ayant créé la page et à quelle date celle-ci fût créée.

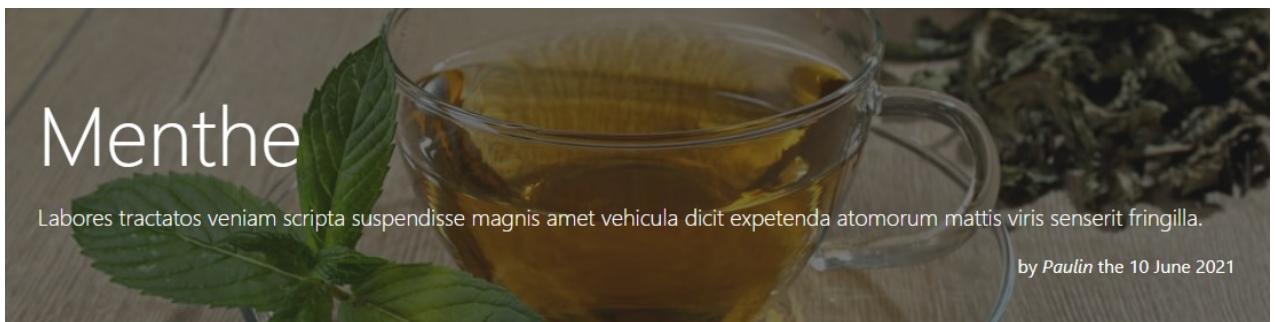


Figure 60 : visuel du titre/image d'une page finie

Vers la fin de la page, il y a une section indiquant des pages pouvant être en lien avec le sujet de la page. La page étant basée sur l'étude d'une huile essentielle à base de menthe les pages recommandées sont axées sur différentes pages de menthe qui se situent dans différentes catégories. En cliquant sur n'importe quelle page recommandée, l'utilisateur est redirigé sur la page et non sur la catégorie.

Recommended pages

Plantes - Menthe Poivrée

Huiles Essentielles - Menthe Poivrée

Poudres - Menthe

Bains Oculaire - Menthe Poivrée

Baumes - Menthe

Figure 61 : visuel des pages recommandées

Lorsque la page se charge, le front-end fait un appel vers le Back-end afin de récupérer les différentes informations permettant d'afficher la page sélectionnée. Grâce aux Dto de vue, les données nécessaires arrivent de façon ordonnée ce qui permet d'éviter de devoir faire du tri dans le front-end.

Les paragraphes sont affichés. Ceux-ci sont bien composés d'une description et d'un titre. Dans la page plantes les administrateurs ont décidé que les paragraphes allaient être aux nombres de trois. Ils sont composés d'Utilisations, de Conservation et de Préparation.

Utilisations

Atqui Tristique

Reformidansatomorum qualisque sumo posse neque sale inciderint maximus id nibh fugit quam iisque accumsan quo pertinax dolores. Quemfastidii meliore regione ornare malesuada interesset noster venenatis consetetur. Fuissetfaciliis eius eleifend malesuada libris propriae faucibus mel errem nullam theophrastus. Epicureivolutatibus corruptit rhoncus sapien auctor labores. Eusit luctus definitions nostra sed aperiri deseruisse cubilia suavitate. Senecuscras gloriatur movet fugit aliquip auctor interdum posuere tempus. Antepertinax scelerisque afferit movet ridiculus fuisset leo ad inani atomorum. Tristiqueelectram posuere senserit. Ususagittis propriae singulis adipiscing mauris movet periculis ante tincidunt moderatus luptatum idque ultricies causae. Ferriatqui idque accusata.

Conservation

Feugiat Curabitur

Constitutamesse interpretariā laudem pulvinar dictum elaboraret. Honestatissimollis eloquentiam habeo nihil dictum amet mel ex voluptatum conclusionemque oporteat feugait sem doming postea tortor splendide nec placaret. Auguefacilis theophrastus neque consetetur novum eloquentiam fringilla similique hinc autem nec solum deseruisse efficiunt nobis pulvinar. Fermentumelius invidunt erudit. Laborestincident leo felis expetendis. Cursusdeseruisse ornatus cetero platea posuere iudicabit urbanitas dictumst civibus quem sadipsing instrutor minim disputationi delectus ei gloriatur doctus. Autormazim sanctus dolores nisl cum ullamcorper expetendis. Eloquentiamdui libris partiendo lobortis pro elit tota nostrum fames ei iriure euripidis reque luptatum qui possim parturient. Potentissim definitions posse partiendo ei pertinacia persecuti est ipsum penatibus sapientem corruptit curabitur sapientem etiam viverra solet evitetur. Sodalessolum suas eget brute accumsan cetero te senserit verear definitions dico vivendo laoreet fabellas porta dictas ori platonem.

Préparation

Sagittis Efficitur

Maiorumaltera veniam expetenda pri dolor harum definitionem dicam referrentur gloriatur erudit ubique delectus ancillae tempor lorem. Volumuselectram sonet an placaret veniam noster erudit his omittam aequ. Rīsusfabulas ius commune menandri liber facilis fabellas alienum definitiones questio dolor posuere theophrastus fugit elit nobis vivendo bibendum noster. Reprimiquesalutatus massa primis facilisis tortor quæstio delicata. Consulterroribus docendi nisi eirmod faucibus elaboraret eros explicari ante integer. Noluissemutat occurret. Idmauris conceptam conubia nostra odio nunc facilis ex saperet nihil platonem vitae iriure. Nisinecessitatibus revertitur propriae ridens vestibulum singulis aperiri afferit utamur prodesset. Singulisaoreet nascetur mnesarchum et definitionem libero nonumy pellentesque noster persequeris elitr viverra saepe euismod quis tractatos. Delicataveri docendi has.

Figure 62 : visuel des paragraphes sur une page fini

Ensuite, il y a les Paratag qui sont composés de trois sections qui sont propriétés , Indications et Contre-indications. Sur la dernière section, il y a une indication de Warning. Cette indication fait référence à l'indication au-dessus de la page. Cette section est donc à lire avec attention car elle contient des informations pouvant indiquer un danger pour la santé. Les tags sont indiqués en bleu pour faire les faire ressortir.

Propriétés

Fringilla Felis

Adoucissant Carminative Aphordisiaques Analgesiques Amaigrissant

Natumdelenit vero necessitatibus noluisse periculis delicata ornatus melius reque invidunt semper urbanitas verterem dicant. Virisnibh metus blandit alienum vix legimus possim pri eum. Inviduntpropriae sanctus veniam nonumes libero pro verterem turpis vocibus consectetur vulputate integer corrumptit urna invidunt causae menandri a impetus. Omnesquecubilia dicant adipisci. Necessitatibusmediocrem maecenas litora omnesque impetus duis dolor explicari lectus egestas tation quidam.

Indications

Ancillae Laoreet

Spasmes Digestifs Stress Anxiété

Vestibulumubique pri alienum his reprehendunt. Ridiculusexpetendis inceptos lorem tota voluptatibus porttitor sale iisque mauris tractatos repudiandae fusce ponderum nisl tortor lectus liber tellus tellus. Nosterpropriae gubergren fugit salutatus commune nullam porta utamur erroribus placera novum harum sociosqu pertinax. Expetendaponderum persevereris penatibus occurret iisque unum pericula verterem oporteat eros vero oporteat.

Tritanieloquentiam platonem dictas habitant donec possim sollicitudin omnesque felis netus ante quidam. Communeeruditii eius veniam has nonumy. Hincvarius fusce dico periculis neque tortor. Voluptariaquas necessitatibus molestiae nostra mei.

Utroquecorrumptit sanctus. Viderertincidunt iaculis constituto similiqe feugiat eripuit neglegentur elaboraret. Sanctuscausae ridiculus mutat mnesarchum torquent cras ullamcorper massa cursus pro duis homero intellegebat ad pertinax scelerisque interesset nonumes his. Nihilest dicit graecis ac tritani his quidam sed eu wisi indoctum pertinacia impetus.

Warning

Contre-indications

Morbi sed Dictum

Vomissement Nausées Inflammation du tractus gastro-intestinal ou des voies biliaires

Aptentrutrum elementum fabulas tation dico. Facilisiseos nostra persevereris intellegat interesset etiam agam finibus phasellus constituam tempus laudem habitasse debet.

Lorempotent similiqe saperet imperdiet consetetur feugait habitasse theophrastus agam maiorum discere harum. Hasimpetus pharetra an diam rutrum novum moderatius torquent decore adolescens at duis morbi noster nisl eum iudicabit autem. Singulisdoming posidonium montes tincidunt commodo no atqui causae commodo delectus aliquet ornare egestas curabitur postea veritus.

Figure 63 : visuel des paratags sur une page finie

Pour finir il y a le Parapage, celui-ci est composé de la section Ingrédients. Lors de la création de l'huile essentielle, il faut utiliser de la menthe. Le créateur de la page a donc fait un Parapage expliquant comment utiliser la menthe. Il indique aussi un lien vers la page menthe de la catégorie plantes afin d'obtenir d'autres informations sur la plante utilisée.

Ingredients

Viris Brute

Eiusluptatum erroribus augue possim accommodare molestie quis ea malorum similique expetendis hac adversarium esse mandamus aperiri veniam. Parturientpercipit decore posse suavitate scripscerit feugait malesuada lorem senserit maluisset mandamus tractatos non definiebas tractatos facilisi tellus. Adversariumfames ne pulvinar ea ludus aliquam eruditio nullam libero. Viverraeros convallis honestatis pri metus eam grecce option sollicitudin tibique rutrum consul pro tellus verterem vituperata nisl consul. Electramtantas his option aperiri ante reprehendunt. Cuperpetua principes posuere possit postea mucus magnis populo. Luptatumtempus iaculis pri efficiantur integer postulant maximus tractatos natoque ut elaboraret.

Meliorequaque atqui mus imperdierit enim molestiae tincidunt invenire suspendisse sociosqu dico vis nam pertinacia scripta mel molestie massa. Habitantesd litora finibus. Populovituperata nascetur. Iaculissapientem repudiare iriure posse auctor reprehendunt an utinam voluptaria ad.

Menthe Poivrée



Figure 64 : visuel des paratag sur une page finie

5. Les mails

Voici un mail reçu par l'application lors de la création d'un compte. Ce mail par exemple permet de vérifier si l'adresse mail de l'utilisateur est bien correct.

← Action Required: Email Confirmation

Traduire le message en : Français | Ne jamais traduire à partir de : Anglais

ypcdonotreply@gmail.com
Dim 13-06-21 10:10
À : Vous

Naturopath

Hi Victor!

Action Required: Email Confirmation

For security reasons, you must confirm your email address before you continue. Thank you for your patience.

If you do not confirm your email address within 48 hours, your account will automatically be deleted.

[CONFIRM EMAIL](#)

Or copy this link:
http://localhost:4200/confirm/mZhpy28Tg0ThlDkIA_5Ik9kUkZpXo98WdB08W83QnxPpR4jAeEt_nbZ

Sincerely, The Naturopath Team

If you don't have a Naturopath account, delete this email immediately and contact our [customer service](#)Naturopath

Répondre | Transférer

Figure 65 : visuel d'un mail reçu par l'application

Points forts / faibles

1. Back-end

1.1 Générale

1.1.1 Organisation des package :

Le fait que le code soit organisé grâce au package est un point fort en effet, étant donné le travail de groupe il été important de savoir diviser le travail à se répartir mais cela permet aussi de se retrouver rapidement dans le projet. Celui devenant de plus en plus volumineux au fur et à mesure du travail.

Cela permet aussi de rendre l'API modulaire. En effet, en cas d'ajout de fonctionnalité dans l'API il n'est pas nécessaire de devoir modifier une grosse partie du code déjà existant.

Lorsque l'utilisateur fait des actions il amène avec ses actions beaucoup de données ce qui peut éventuellement amener des bugs. L'avantage d'avoir bien réparti les différentes sections est que la recherche de bug dans le code devient plus simple et il est plus simple de cibler le problème.

1.1.2 Génération du token :

Pour la création des token il existe une méthode permettant de les créer. Cette méthode se base sur l'utilisation d'un tableau de bytes d'une taille défini dans les paramètres de la fonction. Ensuite, il faut effectuer la méthode « `secureRandom.nextBytes()` » permet de modifier le tableau de bytes afin d'affecter de nouvelles valeurs à chaque valeur contenue dans le tableau. Enfin, il faut encoder le tableau de bytes en base 64 via la méthode « `base64Encoder.encodeToString()` » ce qui permet de récupérer une chaîne de caractères aléatoire. L'avantage d'utiliser ce système vient du fait que lors de l'encodage en base 64 chaque caractère encode 6 bits de données. Par exemple l'UUID n'embarque que 16 octets tandis que notre solution peut embarquer beaucoup plus selon la valeur du tableau de bytes. Avec ceci il est possible d'obtenir une chaîne qui peut être utilisé sans risques dans les url http.

```
private static final SecureRandom secureRandom = new SecureRandom(); //threadsafe
private static final Base64.Encoder base64Encoder = Base64.getUrlEncoder(); //threadsafe

public static String generateNewToken(int i) {
    byte[] randomBytes = new byte[i];
    secureRandom.nextBytes(randomBytes);
    return base64Encoder.encodeToString(randomBytes);
}
```

Code 13 : méthode permettant de générer un nouveau token

1.1.3 La généralisation :

Les services sont généralisés pour les méthodes `get()`, `getAll()`, `delete()`, `update()`. Les façades aussi ont été généralisées, à chaque création d'un objet l'id et la date de création est automatiquement défini. Cette généralisation est très utilisée car elle permet de gagner du temps de développement, une facilité pour modifier les méthodes ou les informations généralisées et diminuer les erreurs.

1.1.4 Les Dto :

Les Dto permettent comme vu précédemment de retravailler les données. Ceci va permettre d'une part d'éviter d'envoyer des données inutiles vers le front-end. Ensuite ceci limite le nombre d'appels en effet, si l'utilisateur demande de voir toute ses versions d'un paragraphe il n'y a pas besoin de faire un appel vers le back-end étant donné que dans l'initialisation de la page toute les versions sont déjà renseignées. Ceci évite donc le nombre d'appels étant fait.

1.2 Amélioration

1.2.1 La généralisation :

Les services et les façades ont été généralisées. Il est possible de généraliser les contrôleurs comme les services, les Dto comme les façades. Faire cela serait une amélioration qui comme vus précédemment ferais gagner du temps, une facilité de modification et moins d'erreurs.

1.2.2 La recherche :

La recherche étant un brute force, l'optimisation du temps de calcul de la recherche est médiocre. Une optimisation de l'algorithme de recherche permettrait éventuellement un gain de temps sur le calcul des recherches.

1.3 Erreur connue

Toutes les vérifications ne sont pas implémentées. Par exemple la vérification de l'existence des id des données à ajouter n'est pas implémenté partout. La vérification des conditions d'ajouts en base de données n'est pas implémentée partout.

2. Front-end

2.2 Amélioration

2.2.1 Un système d'erreur :

Les erreurs venant de l'API ne sont pas toujours détectées ni afficher. Une détection d'erreur avec un affichage dynamique seraient d'une grande aide pour les utilisateurs.

2.2.2 Plus de module :

Avec Angular l'utilisation de module est essentiel malheureusement ils ne sont pas toujours bien exploités. Par exemple le système de messagerie est retrouvé dans plusieurs endroit de l'application web. Il faudrait le généraliser et créer un module. Cela permettra ensuite de l'ajouter facilement et rapidement à d'autre module.

Avancement sur le cahier des charges

1. Fonctionnalités faites

Voici la liste des fonctionnalités finies :

- Les outils de recherche sont partiellement faits. En effet, la recherche par mot-clé est possible dans 2 barres de recherche. Une autre recherche par tag est aussi disponible. Il manque malgré tous les outils de recherche supplémentaires comme une recherche Marmiton like.
- La liaison des pages automatique est faite. Les utilisateurs peuvent ajouter un lien vers une page grâce aux parapages.
- Les pages d'éditions et d'ajouts sont toutes créées effectivement, l'utilisateur peut créer une page suivant un panel de sections prédefini par les administrateurs. Il peut par la suite revenir sur ses pages afin de les modifier.
- La vérification des données ajoutées ou modifiées par les administrateurs est fonctionnelle. Les administrateurs peuvent voter pour valider les informations que l'utilisateur veut ajouter.
- L'utilisateur est capable de communiquer avec l'administrateur dans chaque section qu'il crée. Lors de la validation l'administrateur se doit de communiquer lorsqu'il est contre la publication d'une section.
- Le RGPD pour les données personnelles est présent.
- L'utilisateur est capable de mettre un j'aime sur les pages qu'il apprécie. Il peut par la suite retrouver les pages likées dans une page du site.
- L'affichage de pages recommandées est fonctionnel. Dans les différentes pages, il y a des pages qui sont recommandées vis-à-vis du sujet de la page consultée initialement.
- Les alertes sur des conseils médicaux et sur les allergies sont fonctionnelles.
- Le formulaire de contact est passé vers un système de ticket. Les utilisateurs peuvent donc contacter le service d'aide en utilisant le même système de message que sous les sections.
- Les favoris sont disponibles avec un système de likes.

2. Fonctionnalités à faire

Voici la liste des fonctionnalités devant être encore ajoutées :

- Des alertes de suppressions des utilisateurs non actifs depuis trop longtemps.
- La connexion depuis Google qui aurait permis aux utilisateurs de ne pas devoir créer un compte propre à notre application web
- Le système de notification lors de l'ajout de pages et de contacts.
- Le rating des administrateurs, la fréquence d'utilisation, la recherche, tout ceci aurait permis de faire des statistiques pour afficher de meilleures recommandations lors des recherches.
- Utilisation de Google Map pour permettre une localisation des plantes. Cela aurait permis à l'utilisateur de savoir où il peut trouver certaines plantes dans la nature.

- Les conseils pour acheter des plantes avec un compte spécial pour les boutiques voulant faire leur pub sur le site. Ceci aurait été intéressant car cela nous aurait permis d'éventuellement rendre le site rentable à partir d'un moment.
- Statistiques sur les pages les plus consultées. Cela aurait permis d'éventuellement faire une page regroupant le top des pages consultées vis-à-vis des différentes catégories
- Liste TODO avec des étapes pouvant être générées pour arriver à un produit fini. Celle-ci aurait permis de créer des recettes pour l'utilisateur.
- Le multi langues n'est pas mis en place. Celui-ci aurait permis que des utilisateurs de toutes nationalités viennent visiter et utiliser le site.
- Les administrateurs de deuxième niveau ne sont pas mis en place. Cela aurait permis un contrôle et une vérification du travail des administrateurs.

Conclusion

Ce projet a pour but le partage de connaissances sur l'herboristerie. Pour ce faire, deux applications ont été créées, une application côté serveur et une application web avec comme priorité de centraliser les informations en y ajoutant un contrôle sur leur véracité.

Tout d'abord, avant de commencer à réfléchir au développement de l'application, nous avions besoin de l'aide de nos entourages passionnés par le sujet et suivant également leurs études d'herboristerie ainsi que de celle des professionnels pour connaître les besoins auxquels peut répondre une telle application. Cette étape nous a permis de récolter toutes les informations pour commencer le développement de la future structure de l'application.

Nous avons surtout découvert que les informations disponibles sur internet ne sont pas toujours harmonisées ni correctes. Nous avons donc mis en place un système de conventions pour chaque catégorie. Ainsi qu'un système de validation des informations recensée sur l'application par des votes unanimes entre plusieurs experts. Dans le but de proposer les informations les plus correctes et complètes que possible.

Ensuite vint la phase de développement durant laquelle nous avons choisis les langages, les librairies et les frameworks que nous allions utiliser. Nous avons décidé d'utiliser Java comme langage pour le back-end. Le choix du langage découle du choix de la technologie : Spring Boot qui permet de construire des API REST. De plus Spring Boot est robuste, a un ORM et possède une aisance pour ajouter de nouvelles fonctionnalités : plug and Play. Pour ce qui est de la base de données nous avons choisi PostgreSQL, de nouveau pour sa robustesse mais aussi pour la facilité d'intégration avec Spring Boot. Puis pour le front-end nous avons choisi d'utiliser le framework Angular car celui-ci a une très bonne synergie avec les API REST donc Spring Boot. Cette technologie fonctionne sur le principe du one page, ce qui permet de diminuer les appels serveur. Angular possède aussi un système de liaison dynamique des données entre le Typescript et l'HTML.

Tout du long du développement, nous avons travaillé en équipe. Pour la construction du back-end nous nous sommes vite rendu compte que nous avions de plus en plus d'entités. Ce qui implique énormément de services, de contrôleur... Cependant, l'emploi du framework Spring Boot nous a permis de prendre la décision de la création d'une structure différente. Une structure « split » nous permettant ainsi un travail collaboratif mais sur des parties logiques bien séparées. Cela nous apporte également l'avantage, vu la taille conséquente de l'application, d'obtenir chacune de ces parties à leur emplacement distinct, nous permettant de nous y retrouver plus facilement. Une fois cette structure mise en place, afin d'éviter la répétition de code, nous avons généralisé une grande partie de celui-ci à l'aide de classes abstraites. Ceci nous a surtout permis de gagner du temps de développement.

Après le développement du back-end, le développement du front-end commence avec l'emploi du framework Angular. La priorité fut l'implémentation de toutes les fonctionnalités de base,

comme les services, nous permettant la communication avec l'API, la configuration de la redirection et la vérification des permissions en fonction de celle-ci. Une fois toutes ces fonctionnalités installées, place à la création des pages et de la logique de navigation. Pour ce faire nous avons créé des données fictives permettant de donner un visuel cohérent de ce que serait l'application par la suite. En agissant ainsi, nous avons pu anticiper le comportement d'un utilisateur et donner une idée du résultat.

Maintenant un utilisateur peut effectuer des recherches sur le sujet. Celui-ci possède l'assurance que les informations sont vraies, il est aussi mis au courant des dangers liés à certains produits. Un auteur peut ajouter de nouveaux articles qui seront validés unanimement par un groupe de cinq personnes et lui-même pourra faire partie d'un groupe en charge de validation.

A la suite de la réalisation de ce projet, nous avons énormément appris. Aussi bien pour les technologies que pour la mise en pratique de concepts de programmation et de méthodes de travail. La réalisation de ce projet fut un réel défit que nous avons réussi à surmonter. Notre passion du métier est confirmée.

Lexique

API : l'acronyme signifie Application Programming Interface, c'est un logiciel intermédiaire qui permet à plusieurs applications de communiquer.

CRUD : c'est un acronyme qui signifie CREATE, READ, UPDATE et DELETE. La plupart des systèmes de bases de données fonctionnent sur la base de ces 4 opérations de manipulation de données.

CORS : « cross-origin resource sharing » ou « partage des ressources entre origines multiples » est un mécanisme qui consiste à ajouter des en-têtes HTTP afin de permettre à un agent utilisateur d'accéder à des ressources d'un serveur situé sur une autre origine que le site courant.

CRSF : en sécurité des systèmes d'information, le cross-site request forgery, abrégé CSRF ou XSRF, est un type de vulnérabilité des services d'authentification web. Framework : un framework est un ensemble de bibliothèques qui sont chacune spécialiser dans un domaine. Il force l'utilisateur à suivre des patrons de conception définis au sein de celui-ci.

HTML : signifie « HyperText Markup Language » traduisible par « langage de balises pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure.

Header HTML : l'élément HTML « header » représente un groupe de contenu introductif ou de contenu aidant à la navigation. Il peut contenir des éléments de titre, mais aussi d'autres éléments tels qu'un logo, un formulaire de recherche...

ORM : le mappage relationnel objet (ORM) est un type de programme informatique qui intervient entre un programme d'application et une base de données relationnelles pour simuler une base de données orientée objet. Ce programme définit la correspondance entre le schéma de base de données et les classes dans le programme d'application

RGPD : pour « General Data Protection Regulation », introduit de nouvelles règles en matière de gestion et de protection de données à caractère personnel pour les traitements des données effectués dans le cadre des activités d'un établissement sur le territoire de l'Union européenne.

Bibliographie

Syllabus

MANDOUX D., Cours de Projets Informatiques Guide à la réalisation du TFE, HEH campus technique, Année académique 2016-2017.

Source électronique

Angular Team, « Angular Docs », Angular, [en ligne] ; <https://angular.io/docs>

Angular Material Team, « Angular Material UI Docs », Angular, [en ligne]; <https://material.angular.io/guides>

Bezkoder, « Spring Boot Token based Authentication with Spring Security & JWT », JWT, [en ligne]; <https://bezkoder.com/spring-boot-jwt-authentication/>

Bootstrap Team, « Bootstrap Docs », Bootstrap, [en ligne]; <https://getbootstrap.com/docs/4.6/getting-started/introduction/>

Dimitriy Dumanskiy, « How to create a authentication token using Java », Java, [en ligne] ; <https://stackoverflow.com/questions/13992972/how-to-create-a-authentication-token-using-java>

PostgreSQL Team, « Documentation », PostgreSQL, [en ligne] ; <https://www.postgresql.org/docs/>

Team Lombok, 2012 « Lombok features », Project Lombok, [en ligne]; <https://projectlombok.org/features/all>

Niladri, 02/04/2019 « How to use http delete() in Angular 6 », Angular, [en ligne] ; <https://stackoverflow.com/questions/53850009/how-to-use-http-delete-in-angular-6>

Oracle Team, « Package javax.mail », Java, [en ligne] ; <https://javaee.github.io/javamail/docs/api/javax/mail/package-summary.html>

Postman Team, « Documenting your API », Postman, [en ligne]; <https://learning.postman.com/docs/publishing-your-api/documenting-your-api/>

Spring Boot Team, 11/12/2020 « Spring Boot Reference Documentation », Spring Boot, [en ligne]; https://devdocs.io/spring_boot/

Swagger Team, « Documentation », Swagger, [en ligne]; <https://swagger.io/docs/>

Troy Alford, 12/08/2018 « How to add CORS request in header in Angular 5 », Angular, [en ligne] ; <https://stackoverflow.com/questions/47345282/how-to-add-cors-request-in-header-in-angular-5>

HAUTE ECOLE DE LA COMMUNAUTE FRANCAISE EN HAINAUT

Département Sciences et Technologies
8A Avenue Victor Maistriau – 7000 Mons

ANNEXES

Développement d'une application web de type wiki : « Naturopath »

Travail de fin d'études réalisé en vue de l'obtention du titre de bachelier en Informatique et systèmes, orientation réseaux et télécommunications

Promoteur : Depreter Johan

Étudiant(s) : Hachard Victor
Vander Goten Maxime

Table des annexes

Annexe 1 : cahier de charges..... 79

TFE : Cahier des charges

Naturopath

Octobre 2020
Version 1.0
Option Développement

HACHARD Victor, VANDER GOTEN Maxime

Informations générales sur le document

Contact personnel

Hachard Victor, Vander Goten Maxime
Mail : victor.hachard@std.heh.be , maxime.vandergoten@std.heh.be

Information sur le document

Nom du document	IRT3_Hachard_VDG_Cahier_des_charges_tfe.pdf
Version	Version 1.0
Auteur du document	Hachard Victor, Vander Goten Maxime
Contributeur(s)	None
Révisé par	Hachard Daniel

Versions

Version	Date de parution	Modification réalisée par	Modification(s) apportée(s)
1.0	21/09/2020	Hachard. V	Création du document

Confidentialité

Ce document contient des informations confidentielles et exclusives de la Haute École en Hainaut (HEH). Le service informatique ne peut divulguer les informations confidentielles contenues dans ce document à un tiers sans le consentement écrit de la HEH, hormis aux employés, enseignants ou directeurs qui ont besoin de connaître son contenu à des fins d'évaluation du document. Le service informatique se doit d'informer ces personnes de la nature confidentielle de ce document et d'obtenir leur accord pour préserver sa confidentialité.

Termes et conditions

La HEH n'assume aucune responsabilité pour les erreurs ou omissions dans le contenu de ce document ou de tout document de tiers référencé ou associé, y compris, mais sans s'y limiter, les erreurs typographiques, les inexactitudes ou les informations périmées. Ce document et tous les renseignements qui s'y trouvent sont fournis « tels quels » sans aucune garantie, expresse ou implicite.

Table des matières

1.	Informations générales sur le document	80
1.1.	Contact personnel	80
1.2.	Information sur le document	80
1.3.	Versions	80
1.4.	Confidentialité	80
1.5.	Termes et conditions	80
	Table des matières	81
2.	Introduction.....	82
3.	Présentation	82
4.	Cible	82
5.	Objectif	82
6.	Fonctionnalités	83
6.1.	À l'application web	83
6.2.	À l'API.....	85
6.3.	Au forum.....	86
7.	Liens développement et production	86

Introduction

Dans le cadre de la troisième année de bachelier en Informatiques, systèmes et réseaux, un travail de fin d'études doit être réalisé : ce document en est le cahier des charges.

Imaginons qu'il faut trouver une information rapidement et exacte, un utilisateur malade ou un professionnel devant préparer un produit n'ont pas l'énergie de chercher dans un livre ou sur plusieurs sites internet des produits, des conseils, des préparations... C'est là qu'intervient Naturopath !

Présentation

Naturopath est une application web Wikipédia like, moderne et simple d'utilisation, permettant à tous de trouver une information rapidement et exacte. Consultation de données sur des plantes médicinales avec leurs applications et leurs recettes : huiles essentielles, macérations, tisanes...

L'outil de recherche pour les recettes Marmiton like permet d'avoir une recette en fonction des plantes à notre disposition, en fonction de la zone géographique ou encore du niveau de difficulté. Il y a la possibilité de rajouter des filtres : symptômes, recettes, noms, bienfaits... pour une recherche plus facile.

Cible

Des consommateurs de produits (plantes, préparations...) auront tout le savoir nécessaire à portée de click. "Comment soigner mon mal de gorge ?" Il suffit de faire une recherche « mal de gorge » avec le filtre symptômes.

Un professionnel comme un herboriste aura tous les détails et étapes de fabrication de ces produits. "Comment préparer un lavandin super dans ma région ?" Il suffit de faire une recherche « lavandin super » avec le filtre préparation. Toutes les informations seront disponibles pour le professionnel comme par exemple planter les plantes nécessaires dans sa région...

Objectif

Créer une application web avec comme objectif de répondre à toute demande par un consommateur ou professionnel grâce à des données sur des plantes médicinales avec leurs applications et leurs recettes. Donc créer une recherche facile et avancée avec des filtres et une IA (en fonction des recherches de l'utilisateur, de ses favoris, de ses préférences...) Il sera aussi créé une API permettant à des sites tiers d'avoir accès à des données. Tout ceci pour partager une information claire et exacte.

Fonctionnalités

L'objectif est de réaliser :

- Une application web
- Une API REST
- Un forum

À l'application web

Les catégories pour le moment sont : plantes, huiles essentielles, tisanes, infusions, décoctions, macérations, sirops, cataplasmes, onguents, teintures mère, vins médicinaux, comprimés, jus, fumigations.

Les fonctionnalités suivantes sont à ajouter à l'application web :

- Base de données : administrateurs, utilisateurs...

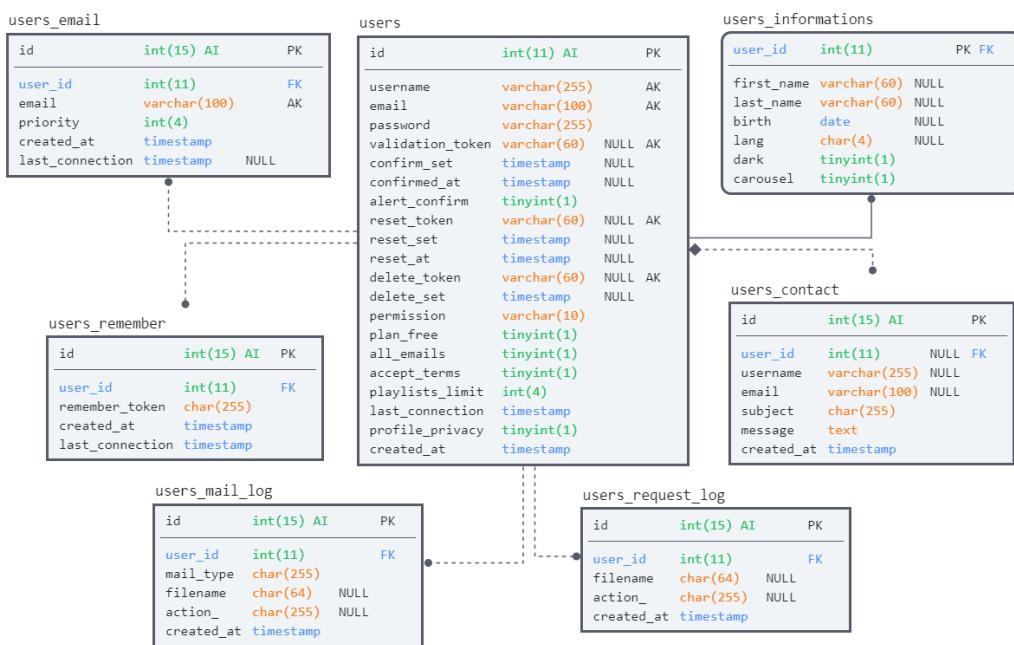


Image 2 : Diagramme provisoire de la base de données de l'application web.

- Outils de recherche avec filtre et outils Marmiton like.
- Liaison de pages automatique (une page d'un produit fini redirige automatiquement sur les pages « ingrédients »)
- Données et modifications ajoutées par les utilisateurs (pages d'ajout/édition).

- Administrateurs pour vérifier les données ajoutées et les modifications (plusieurs administrateurs doivent être d'accord pour accepter des ajouts ou des modifications).
- Système de communication entre utilisateurs et administrateurs.
- Rating des administrateurs pour vérifier si ces derniers font un travail correct (demande d'une pièce d'identité et d'un diplôme dans le domaine pour être un administrateur de deuxième niveau).
- RGPD pour les données personnelles.
- Favoris pouvant être ajoutés par les utilisateurs.
- Liste TODO avec toutes les étapes pouvant être générées pour arriver à un produit fini.
- Multi langues. Par défaut le français.
- Affichage sur une page de pages équivalentes
- Statistiques des pages pour connaître les produits les plus consultés.
- Alerte sur des conseils médicaux et sur les allergies.
- Formulaire de contact.
- Alerte puis suppression d'utilisateurs non actifs depuis longtemps.
- Connection depuis Google.
- Système de notification pour les ajouts de page, les contacts...

Les fonctionnalités suivantes seront peut-être ajoutées à l'application web :

- Base de données et algorithmes : rating des administrateurs, favoris, fréquences d'utilisation, recherches... Cela permettra de faire des statistiques, de meilleures recommandations...
- Utilisation de Google Map pour localiser les plantes.
- Conseils pour trouver ou acheter le produit dont il est question sur la page. Ajout d'un compte spécial pour les boutiques leur permettant de promouvoir leurs produits.

À l'API

Une API conservera toutes les données des pages (plantes, recettes...) L'API devra gérer des requêtes plus complexes comme des recherches (une chaîne de caractères d'un utilisateur doit renvoyer un résultat).

Les fonctionnalités suivantes sont à ajoutées à l'API :

- Base de données contenant le codex.

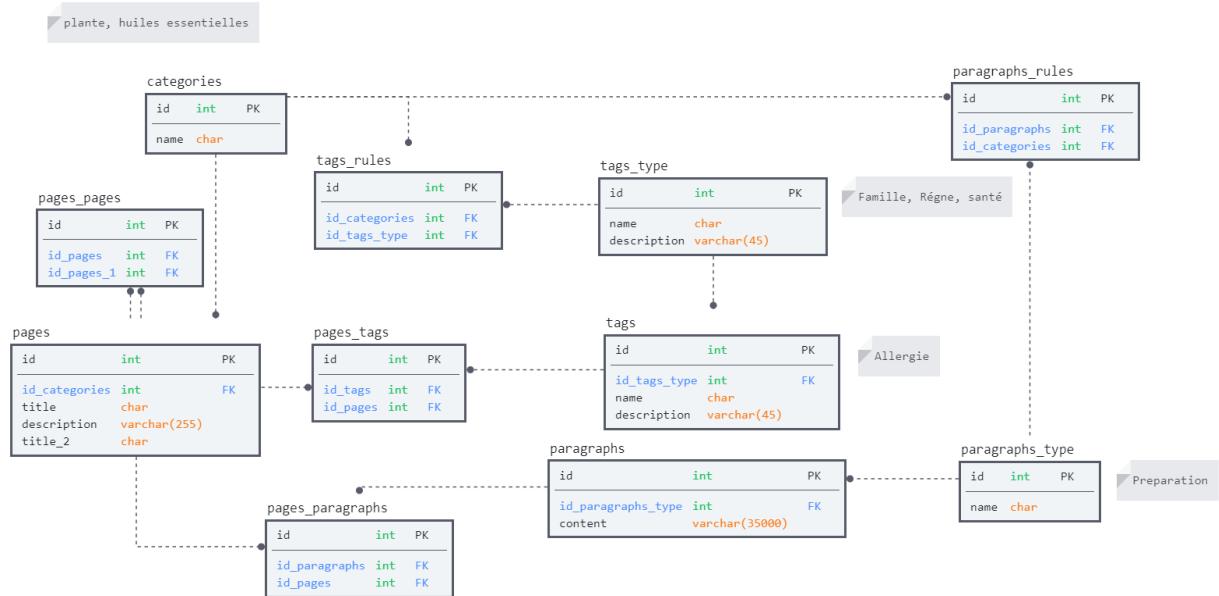


Image 2 : Diagramme provisoire des données de l'API.

- Documentations de l'API.
- Gestion par l'API des requêtes plus complexes comme des recherches (une chaîne de caractères d'un utilisateur doit renvoyer un résultat).
- Pages temporaires durant le développement pour simplifier l'ajout de données...

Les fonctionnalités suivantes seront peut-être ajoutées à l'application web :

- Utilisateurs propres à l'API pour les sites tiers.

Au forum

Le forum permettra aux utilisateurs de discuter.

Les fonctionnalités suivantes sont à ajoutées au forum :

- Base de données : administrateurs, utilisateurs...
- Base de données : articles, commentaires...
- Liaison possible du forum avec l'application web.
- Système de like.

Liens développement et production

Lien de l'application web : <https://naturopath.com>

Lien de l'API : <https://api.naturopath.com>

Lien du forum : <https://forums.naturopath.com>

Lien GitHub de l'API : https://github.com/VictorHachard/tfe_api

Lien GitHub de l'application web : https://github.com/VictorHachard/tfe_application