www.isl.be

INSTITUT SAINT-LAURENT
enseignement de promotion sociale

rue Saint-Laurent 33 - 4000 LIEGE
04 223 11 31

Travail de fin d'études

« miel-belge.be »

Saint-André, le 12 juin 2021

Travail de fin d'études réalisé par

BUSTIN Jonathan

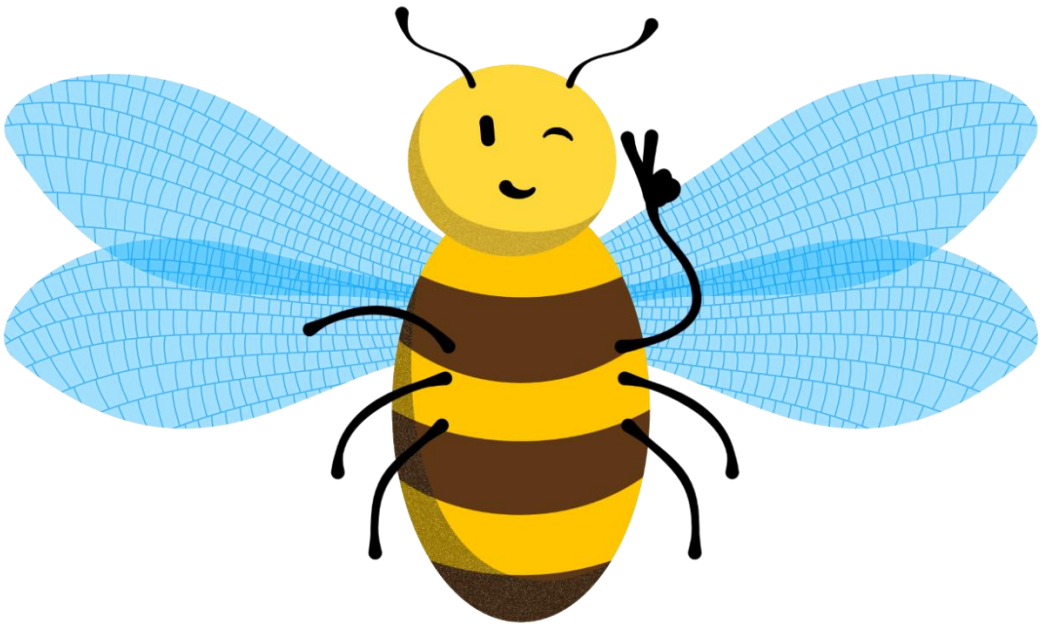
En vue de l'obtention du brevet de l'enseignement supérieur de **Webdeveloper**

Enseignement supérieur économique de promotion sociale et de type court

www.isl.be

INSTITUT SAINT-LAURENT
enseignement de promotion sociale

rue Saint-Laurent 33 - 4000 LIEGE
04 223 11 31



« miel-belge.be »

Saint-André, le 12 juin 2021

Travail de fin d'études réalisé par

BUSTIN Jonathan

En vue de l'obtention du brevet de l'enseignement supérieur de **Webdeveloper**

Enseignement supérieur économique de promotion sociale et de type court

Remerciements :

Merci à Orane et Philomène pour leur soutien,

Merci à Colette Lahaye pour la relecture

Merci à Nicolas pour la conception des images du site,

Merci aux différents professeurs de la section webdeveloper pour leur aide,

Merci à Fabien Potencier, l'équipe de Symfony SAS et SensioLabs d'avoir développé un Framework aussi sympa et une documentation riche et variée,

Merci à la communauté de Stackoverflow d'avoir répondu à beaucoup de mes questions avant même d'avoir pu les poser,

Merci aux petites abeilles sans qui ce projet n'aurait pas eu de sens.

Table des matières

1. Introduction

« Un voyage d'un millier de kilomètres commence avec le premier pas. »
Lao-Tseu

Ce premier pas, pour « miel-belge.be », a été réalisé il y a deux ans avec une première proposition de cahier des charges, de schéma de base de données et des premières lignes de code. Ayant dû mettre ce projet de côté durant un an, j'ai eu l'occasion de le repenser, d'en avoir une vision plus large et de mieux déterminer sa finalité.

Amateur de miel, j'apprécie pouvoir trouver des miels locaux et de qualité. En France, lorsque vous vous promenez sur un marché, il est facile de trouver des apiculteurs qui vendent leurs produits. En Belgique, l'apiculture est une passion et non un métier. Le miel est essentiellement vendu en vente directe (62%) et via les commerces locaux (28%) (chiffre de 2008)[1]. Malheureusement, certains apiculteurs ont du mal à vendre leur stock (manque de bouche à oreille, faible visibilité…). Cette réalité peut être un frein au développement de nouvelles ruches et de nouveaux produits.

Développer un annuaire pour les apiculteurs est donc un moyen de les mettre en avant, d'afficher leurs produits et de leur permettre d'avoir une meilleure visibilité. Cela pourrait leurs permettre d'augmenter leurs activités et, sur le long terme, de diminuer le ratio miel produit/miel importé pour la Belgique. En effet, la Belgique produit moins de deux tonnes de miel par an (15 à 25% du miel consommé en Belgique). En 2016, la production européenne avoisine les 250 000 tonnes de miel par an. 200 000 tonnes de miels ont dû être importés, dont 40% en provenance de Chine[2].

Si la réalisation de ce projet a donc pour but de mettre en avant les apiculteurs et leurs produits, il se veut une alternative aux grandes surfaces en proposant un commerce plus local, plus réfléchit et plus écologique.

Le site a été développé avec Symfony et Node.js. Dans la suite de ce document, vous pourrez mieux comprendre l'objectif de « miel-belge.be » mais également les difficultés et solutions à la mise en place d'un tel projet.

J'espère que vous aurez autant de plaisir à le lire et le tester que moi à le développer.

---

[1] **Agnès Fayet**. 2019/2020. *L'apiculture en Wallonie 2020, Contexte, analyse et pistes d'actions*, p. 12 à 35. Disponible sur : https://www.beewallonie.be/wp-content/uploads/2020/10/BW-Analyse-du-secteur-apicole-2019-fin.pdf (consulté en mai 2021)

[2] **Parlement Européen**. 2018. *Le marché du miel dans l'Union européenne (infographie)*, Disponible sur : https://www.europarl.europa.eu/news/fr/headlines/economy/2080222STO98435/le-marche-du-miel-dans-l-union-europeenne-infographie (consulté en mai 2021)

2. Description du site

2.1 Thème et brève description

Le site « miel-belge.be » est un annuaire qui a pour objectif de mettre en relation les petits producteurs de produits de la ruche belge avec différents clients via une interface de gestion simple.

Ce site est un travail fictif car il ne répond pas à une demande directe d'un prestataire ou d'une entreprise. Cependant, il a une finalité réelle étant donné qu'en faisant l'état des lieux, aucun des sites trouvés n'avait cette mission.

Si vous cherchez sur l'internet un apiculteur près de chez vous, vous trouverez deux sites web.

Le premier, est le site du CARI (www.cari.be) qui sur la page https://www.cari.be/apiculteurs/ ressence les apiculteurs membres. Il est possible d'afficher des informations relatives à l'apiculteur (adresse, photos, …) mais également de le contacter via un formulaire de contact. Cependant, ce n'est pas le but premier du CARI et le listing des apiculteurs n'est qu'une page parmi d'autre. Cela rend l'information peu accessible et difficile à mettre en avant.

Le deuxième site est le site de Vivre ici (www.vivreici.be) qui sur la page http://www.vivreici.be/article/detail_la-carte-des-apiculteurs-en-wallonie-et-a-bruxelles?id=123545 ressence les différents apiculteurs inscrits sur une carte interactive et l'affichage de données personnelles (adresse, email, …) pour contacter l'apiculteur. A nouveau, les apiculteurs ne sont qu'une petite partie des fonctionnalités proposées par le site.

Le site « miel-belge.be » veut se démarquer en mettant en avant uniquement les apiculteurs et leurs produits et en facilitant l'échange entre les apiculteurs et leurs potentiels clients.

2.2 Public cible

Le site « miel-belge.be » cible les apiculteurs et les consommateurs réguliers de miel et de produits de la ruche. Le site se veut familial, transparent, écoresponsable et soucieux de la qualité des produits mis en vente par les apiculteurs. Il s'adresse également aux personnes voulant favoriser les circuits courts et les productions régionales.

2.3 Objectifs

En tant qu'annuaire d'apiculteurs, le site « miel-belge.be » a comme objectifs principaux de :

- Favoriser la vente de produits de la ruche belge en circuit court et ainsi éviter les produits « industriels » dont l'origine est généralement UE/Non UE,
- Répertorier les apiculteurs de Belgique et afficher des informations détaillées concernant un apiculteur et ses produits,
- Permettre aux apiculteurs d'avoir une bonne visibilité sur l'internet à moindre coût et sans devoir créer et gérer son propre site,
- Permettre aux apiculteurs de vendre du matériel apicole via la newsletter mensuelle,
- Permettre aux apiculteurs d'étendre leur gamme de produits en fonction de la demande,
- Permettre aux utilisateurs de découvrir de nouveaux produits dans sa région/une autre région de Belgique,
- Permettre aux utilisateurs connectés d'écrire un commentaire sur un apiculteur afin d'enrichir le site,
- Proposer la possibilité aux utilisateurs connectés de faire une réservation auprès d'un apiculteur en mettant en relation directe le client et l'apiculteur,
- Donner des informations sur les différents produits de la ruche.

En tant que site, « miel-belge.be » a pour objectifs :

- D'être écoresponsable,
- D'assurer son autonomie financière,
- D'avoir une politique en matière de vie privée relativement stricte.

3. Exigences fonctionnelles

3.1 Fonctionnalités

3.1.1 L'utilisateur non-connecté

- Il peut consulter une partie de la fiche d'un apiculteur,
- Il peut rechercher un apiculteur,
- Il peut s'inscrire sur le site en tant qu'apiculteur ou simple utilisateur,
- Il peut afficher la liste des catégories de produits ainsi que la description associée à chaque catégorie de produit,
- Il peut changer la langue du site,
- Il peut rechercher un produit,
- Il peut visualiser l'ensemble des apiculteurs sur une carte,
- Il peut faire une demande de nouveau mot de passe,
- Il peut accepter d'être géolocalisé.

3.1.2 L'utilisateur connecté

- Il peut faire ce que l'internaute non identifié fait (excepté authentification/inscription),
- Il peut consulter la fiche complète d'un apiculteur,
- Il peut se déconnecter,
- Il peut mettre un commentaire et une note sur un apiculteur,
- Il peut contacter un apiculteur,
- Il peut consulter l'agenda d'un apiculteur,
- Il peut modifier son profil et son mot de passe,
- Il peut faire une réservation à un apiculteur,
- Il peut signaler un problème à l'administrateur,
- Il peut signaler un abus dans les commentaires,
- Il peut faire un don afin d'aider à supporter l'infrastructure de l'hébergement,

- Il peut ajouter et/ou supprimer un apiculteur à ses favoris,
- Il peut supprimer son compte.

3.1.3 L'apiculteur

- Il peut faire ce qu'un internaute identifié fait,
- Il peut s'inscrire et/ou se désinscrire de la newsletter,
- Il peut gérer l'ensemble de ses produits,
- Il peut gérer l'ensemble de ses photos,
- Il peut gérer son agenda,
- Il peut répondre à une réservation,
- Il peut répondre à un message,
- Il peut mettre en vente du matériel apicole via un formulaire.

3.1.4 L'administrateur

- Il gère l'ensemble des membres du site,
- Il gère l'ensemble des catégories et des sous-catégories de produits,
- Il gère les « abstract product categories » et les « abstract subcategories »,
- Il gère les abus signalés sur les commentaires,
- Il accède aux statistiques de dons,
- Il accède aux statistiques sur les commandes en cours,
- Il peut accéder à une conversation en cas de signalement d'un abus,
- Il gère la newsletter.

3.1.5 Le server

- Gère le statut de premium via un CRON (enlève le statut de premium après un an),
- Gère la suppression des évènements présents dans les calendriers des apiculteurs en supprimant ceux qui sont passés depuis un mois.
- Il gère la suppression d'un apiculteur après 2 ans d'inactivité sur le site (pas de connexion)

3.2 Règles métiers

- La création d'un compte doit impérativement passer par l'acceptation de la politique en matière de vie privée,
- La création d'un compte passe par l'envoi d'un email,
- Un compte est bloqué après trois essais infructueux,
- La réinitialisation du mot de passe se fait via l'envoi d'un email,
- Le mot de passe contient huit caractères dont une majuscule, une minuscule, un chiffre et un symbole,
- Un compte est caractérisé par une adresse mail unique,
- Un compte banni n'est plus affiché,
- La suppression d'un compte ou d'une entité est définitive,
- L'interface d'administration n'est accessible qu'aux administrateurs.

3.3 Interface

- Le site doit être fonctionnel sur les principaux navigateurs (Google Chrome, Microsoft Edge, Firefox, Safari, Opéra et Internet Explorer),
- Le site doit être agréable visuellement (sobriété au niveau des couleurs et des animations),
- La typographie est uniforme sur l'ensemble du site,
- La typographie et la police de caractère doivent apporter un bon confort de lecture,
- Les images fixent doivent être en rapport avec la thématique et de bonne qualité,
- Les liens cliquables doivent être évidents,
- Les temps de réponses doivent être raisonnables,
- L'utilisation des différentes fonctionnalités doit être simple et évidente quel que soit le support (smartphone, tablette et ordinateur).

4. Exigences non fonctionnelles

4.1 Conception

- Le site est conçu de façon à veiller au référencement,

- Un ou plusieurs noms de domaine seront déposés,
  - « miel-belge.be »
  - « belgische-honing.be »
  - « belgischer-honig.be »

- Une ou plusieurs adresses de contact seront déposées,
  - admin@miel-belge.be
  - contact@miel-belge.be

- Le site est compatible avec les smartphones, tablettes et ordinateurs,

- L'interface d'administration est faite sans utiliser EasyAdminBundle.

- La suppression des évènements se fait un mois après leur expiration. Cela permet d'avoir un visuel d'agenda « rempli » même quand on est fin du mois. Supprimer les évènements plutôt que de mettre un statut « inactif » évite de stocker des données « inutiles » sur le server. Même si l'impact est faible, le stockage de données inutiles à un impact écologique dû au cout énergétique. Le site se voulant en partie éco-responsable, il semble peut intéressant de conserver ces données.

  La même réflexion s'est faite sur les conversations. Une conversation clôturée par les deux parties est automatiquement supprimée.

- Le statut premium ne donne accès à rien de particulier pour le moment. Ce n'est pas un rôle mais simplement un champ en base de données. Lors de l'affichage des apiculteurs, les apiculteurs premium seront affichés en premier mais l'ensemble des fonctionnalités sont accessibles quel que soit le statut de l'apiculteur. Cependant, il sera possible sur un plus long terme de rendre accessible certaines fonctionnalités uniquement aux membres premium (par exemple, la création, la gestion et l'affichage des produits de l'apiculteurs). L'utilisateur (non apiculteur) premium n'accède à rien de particulier.

4.2 Acteurs

4.2.1 L'utilisateur anonyme

L'utilisateur anonyme doit avoir une expérience agréable sur le site et y trouver suffisamment d'informations pour lui donner l'envie de s'inscrire.

C'est pourquoi, la plus grande partie du site est accessible de manière anonyme. L'utilisateur anonyme pourra trouver les apiculteurs proches de chez lui, trouver des produits répondant à ses critères de recherche et afficher des informations restreintes sur un apiculteur. Il lui sera possible de se créer un panier mais ne pourra pas le valider s'il n'est pas connecté/inscrit.

Afin de respecter les données personnelles des apiculteurs, « miel-belge.be » a fait le pari de ne pas afficher d'informations « sensibles » pour des personnes non-connectées. Et si, une adresse peut être récupérée via la carte, l'adresse précise ainsi que les données de contact ne sont pas accessibles pour l'utilisateur anonyme.

« Miel-belge.be » espère que, en profitant d'une expérience utilisateur agréable, l'utilisateur anonyme décide de s'inscrire afin d'accéder à un supplément d'informations et de faire vivre le site.

4.2.2 L'amateur de miel (client)

Le client a accès à plus d'informations concernant les apiculteurs que l'utilisateur anonyme. Il peut facilement entrer en contact avec un apiculteur via la page de ce dernier, écrire un commentaire ou encore valider une ou plusieurs commandes présentes dans son panier.

Les données entrées par le client sur le site « miel-belge.be » sont : l'adresse email, un mot de passe, un nom, un prénom, un choix de langue et un avatar. Le client est également caractérisé par ses apiculteurs favoris, la date de dernière connexion, son type (apiculteur ou non), son statut premium, le nombre d'abus commis et s'il est banni.

En tant que membre de « miel-belge.be », il est possible pour un client de faire un don afin de soutenir l'infrastructure du site.

4.2.3 L'apiculteur

L'apiculteur est un client particulier. En effet, il a accès à certaines fonctionnalités supplémentaire. Etant le cœur de « miel-belge.be », il fournit davantage d'informations lors de son inscription : une petite description, son adresse complète, le nom de son rucher, s'il souhaite s'inscrire à la newsletter, son site web (facultatif), son numéro de téléphone (facultatif) et une série de photo (facultatif).

Il peut également gérer ses produits mis sur le site, son agenda de vente à domicile et proposer du matériel apicole à vendre via la newsletter.

Avoir une table client ne semble pas nécessaire pour le moment car elle ne demande aucune donnée et n'implémente aucune fonctionnalité différente de celle d'un apiculteur. Cependant, il serait possible d'envisager par la suite un champ propre au client et pas à l'apiculteur (par exemple, un apiculteur pourrait proposer une visite de ses ruches et les clients non apiculteur pourraient s'y inscrire).

4.2.4 L'administrateur

L'administrateur peut être un client ou un apiculteur. Cependant, il a la possibilité de se connecter à l'interface d'administration. L'interface d'administration lui permet de gérer les utilisateurs (bannissement), les catégories et sous-catégories de produit (et leur abstract catégories respectives). De plus, il peut afficher les statistiques des dons, gérer les commentaires signalés comme abusifs, accéder à une conversation désignée comme abusive, accéder aux statistiques des commandes et gérer la newsletter.

5. Cas d'utilisation

- Producteur de miel, Monsieur A, vend ses produits par le bouche-à-oreille. Sa production est légèrement supérieure à la demande. Il distribue sa production excédentaire gratuitement à la famille et aux voisins. Ce mode de fonctionnement ne lui permet pas de rentrer dans ses frais et encore moins d'envisager acheter davantage de ruches. Il décide de s'inscrire sur le site « miel-belge.be » afin d'avoir une meilleure visibilité dans sa région. Une fois son inscription validée (vérification de l'email et formulaire d'inscription complété), il sera repris dans le listing des apiculteurs et affiché sur la carte.

- Madame B est une apicultrice inscrite sur le site depuis plusieurs années. Cependant, l'âge faisant, elle souhaite remettre ses ruches à des personnes passionnées. Elle va pouvoir mettre un ou plusieurs articles en vente via la newsletter mensuelle. Pas très à l'aise avec sa boite mail, elle décide de ne pas afficher son adresse mail dans la newsletter mais bien son numéro de téléphone. Une fois ses articles vendus, madame B pourra se désinscrire du site et il ne restera plus aucune trace d'elle en tant qu'apicultrice.

- Monsieur C est amateur des produits de la ruche. Après s'être inscrit sur le site, il lui est possible de faire une réservation auprès d'un de nos apiculteurs. Il peut consulter et modifier sa réservation tant que celle-ci n'a pas été validée. Une fois la validation faite, l'apiculteur recevra un mail

reprenant la réservation effectuée. L'ouverture d'un canal de discussion lié à la commande sera également créé sur le site afin de finaliser les détails de la commande (paiement, moyen de livraison…).

- Madame D est administratrice du site « miel-belge.be ». Elle a reçu, à plusieurs reprises, des messages concernant l'utilisateur X qui fait des réservations sans jamais se présenter ensuite. Elle peut décider de prendre contact avec lui, d'ajouter un abus ou de le bannir du site.

- Monsieur E est apiculteur. Il possède plusieurs ruches et vend ses produits depuis plusieurs années à son domicile. Cependant, il a décidé de partir durant six mois afin de se ressourcer et découvrir le monde. Il peut gérer son agenda et son stock afin que les personnes ne se présentent plus devant son domicile pour acheter ses produits. Il décide de ne plus afficher son adresse email et son numéro de téléphone sur le site afin de ne pas être contacté durant cette période.

- Madame F veut acheter du miel et de la propolis à Api 1. Avant de finaliser sa réservation, elle regarde les autres apiculteurs de sa région et découvre Api 2 qui vend également du miel et de la propolis. Madame F décide finalement de prendre sa propolis chez Api 2 car celle-ci est un peu moins chère. Dès qu'elle sélectionne l'article, elle constate que son panier contient un élément de plus. Elle peut également aller sur son panier pour supprimer la propolis d'Api 1. Sa réservation est modifiée en temps réel. Elle pourra valider ses deux réservations indépendamment l'une de l'autre.

- Monsieur Y est apiculteur et il récupère la cire pour en faire des emballages alimentaires. Il contacte les administrateurs via le formulaire de contact afin de proposer l'ajout de cette sous-catégorie dans la cire. Après discussion entre les administrateurs, il a été décidé que cela avait bien sa place sur le site. Monsieur Admin 1 ajoute alors l'abstract sub catégorie correspondante alors que Madame Admin 2 s'occupe de créer les sous-catégories correspondantes.

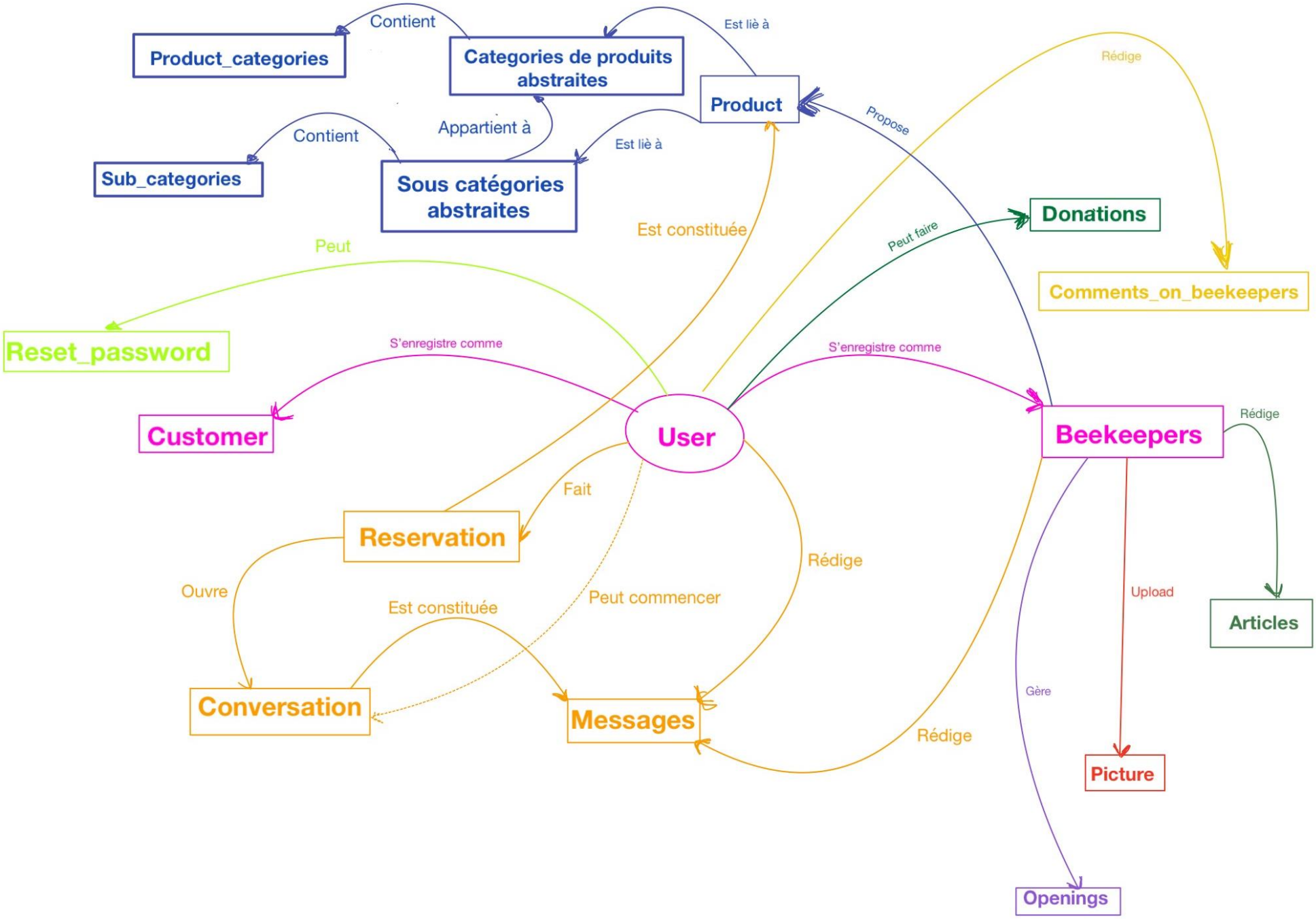
6. Description de la base de données

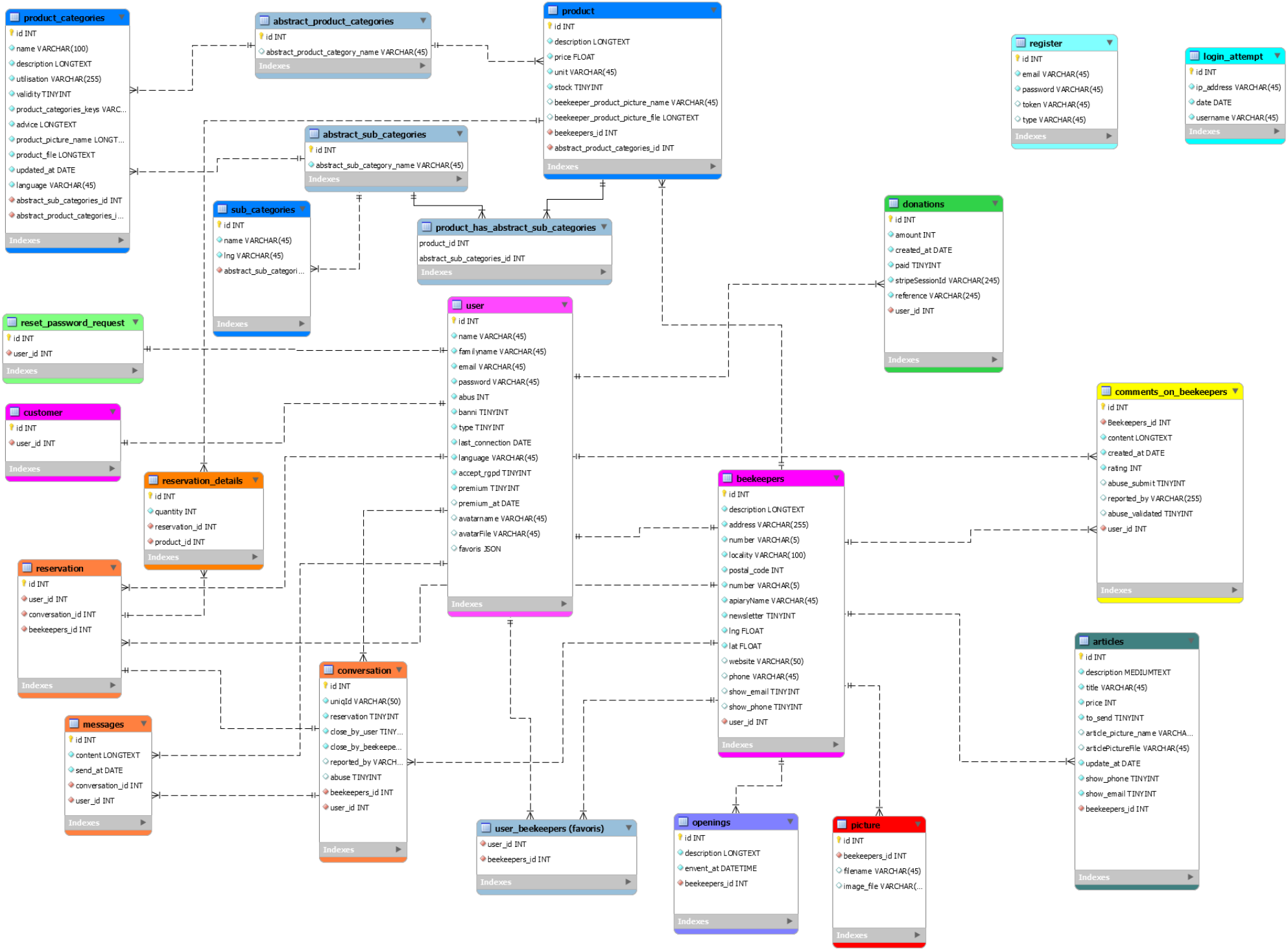
6.1 Schéma conceptuel

Voir page 20

6.2 Schéma physique

Voir page 21


Product_categories
Contient
Categories de produits abstraites
Est liè à
Product
Rédige
Propose
Sub_categories
Contient
Appartient à
Est liè à
Sous catégories abstraites
Donations
Est constituée
Peut faire
Peut
Comments_on_beekeepers
Reset_password
S'enregistre comme
S'enregistre comme
Customer
User
Beekeepers
Rédige
Fait
Reservation
Rédige
Ouvre
Upload
Articles
Est constituée
Peut commencer
Conversation
Messages
Gère
Rédige
Picture
Openings


product_categories
id INT
name VARCHAR(100)
description LONGTEXT
utilisation VARCHAR(255)
validity TINYINT
product_categories_keys VARC...
advice LONGTEXT
product_picture_name LONGT...
product_file LONGTEXT
updated_at DATE
language VARCHAR(45)
abstract_sub_categories_id INT
abstract_product_categories_i...
Indexes
abstract_product_categories
id INT
abstract_product_category_name VARCHAR(45)
Indexes
product
id INT
description LONGTEXT
price FLOAT
unit VARCHAR(45)
stock TINYINT
beekeeper_product_picture_name VARCHAR(45)
beekeeper_product_picture_file LONGTEXT
beekeepers_id INT
abstract_product_categories_id INT
Indexes
register
id INT
email VARCHAR(45)
password VARCHAR(45)
token VARCHAR(45)
type VARCHAR(45)
Indexes
login_attempt
id INT
ip_address VARCHAR(45)
date DATE
username VARCHAR(45)
Indexes
abstract_sub_categories
id INT
abstract_sub_category_name VARCHAR(45)
Indexes
sub_categories
id INT
name VARCHAR(45)
lng VARCHAR(45)
abstract_sub_categori...
Indexes
product_has_abstract_sub_categories
product_id INT
abstract_sub_categories_id INT
Indexes
donations
id INT
amount INT
created_at DATE
paid TINYINT
stripeSessionId VARCHAR(245)
reference VARCHAR(245)
user_id INT
Indexes
reset_password_request
id INT
user_id INT
Indexes
user
id INT
name VARCHAR(45)
familyname VARCHAR(45)
email VARCHAR(45)
password VARCHAR(45)
abus INT
banni TINYINT
type TINYINT
last_connection DATE
language VARCHAR(45)
accept_rgpd TINYINT
premium TINYINT
premium_at DATE
avatarname VARCHAR(45)
avatarFile VARCHAR(45)
favoris JSON
Indexes
customer
id INT
user_id INT
Indexes
comments_on_beekeepers
id INT
Beekeepers_id INT
content LONGTEXT
created_at DATE
rating INT
abuse_submit TINYINT
reported_by VARCHAR(255)
abuse_validated TINYINT
user_id INT
Indexes
reservation_details
id INT
quantity INT
reservation_id INT
product_id INT
Indexes
beekeepers
id INT
description LONGTEXT
address VARCHAR(255)
number VARCHAR(5)
locality VARCHAR(100)
postal_code INT
number VARCHAR(5)
apiaryName VARCHAR(45)
newsletter TINYINT
lng FLOAT
lat FLOAT
website VARCHAR(50)
phone VARCHAR(45)
show_email TINYINT
show_phone TINYINT
user_id INT
Indexes
reservation
id INT
user_id INT
conversation_id INT
beekeepers_id INT
Indexes
articles
id INT
description MEDIUMTEXT
title VARCHAR(45)
price INT
to_send TINYINT
article_picture_name VARCHA...
articlePictureFile VARCHAR(45)
update_at DATE
show_phone TINYINT
show_email TINYINT
beekeepers_id INT
Indexes
conversation
id INT
uniqId VARCHAR(50)
reservation TINYINT
close_by_user TINY...
close_by_beekeepe...
reported_by VARCH...
abuse TINYINT
beekeepers_id INT
user_id INT
Indexes
messages
id INT
content LONGTEXT
send_at DATE
conversation_id INT
user_id INT
Indexes
user_beekeepers (favoris)
user_id INT
beekeepers_id INT
Indexes
openings
id INT
description LONGTEXT
envent_at DATETIME
beekeepers_id INT
Indexes
picture
id INT
beekeepers_id INT
filename VARCHAR(45)
image_file VARCHAR(...
Indexes

7. Présentation des problèmes, des questionnements, des solutions envisagées et des choix techniques

« Il n'y a pas de problèmes ; il n'y a que des solutions. L'esprit de l'homme invente ensuite le problème »

André Gide

La réalisation de « miel-belge.be » fut l'occasion d'être confronté à des difficultés et des remises en question. Nous allons vous retranscrire les problèmes rencontrés, la façon dont le projet a évolué et comment un nouveau projet serait réfléchi.

7.1 Le cahier des charges

Le cahier des charges est le point central du développement d'une application. A refaire, le cahier des charges contiendrait probablement plusieurs pages supplémentaires décrivant au maximum l'application, notamment en ce qui concerne l'expérience utilisateur.

Des sprints avaient été créés afin de fixer des objectifs et de faciliter le développement. Ces sprints ont montré les failles de cette organisation. Le premier sprint envisagé devait permettre d'afficher le listing des catégories de produits, celui des apiculteurs, faire une recherche, afficher une catégorie de produits, afficher un apiculteur, afficher le résultat d'une recherche, l'inscription et l'authentification. Outre les trop nombreuses fonctionnalités pour un sprint, il se divisait entre deux grandes entités : les apiculteurs et les catégories de produits. Le centre du site étant les apiculteurs, la gestion des catégories de produits devait venir bien plus tard. A l'inverse, l'affichage des apiculteurs sur une carte est une priorité beaucoup plus élevée pour le site[3].

Dans les points suivants, lorsqu'un point pourrait être présent dans le cahier des charges afin de l'enrichir et de faciliter le travail de développement, cela sera mentionné.

[3] Vous pouvez consulter en annexe les sprints et les annotations montrant leurs évolutions.

7.2 Le choix des technologies

7.2.1 La connaissance des technologies

Lors de la création du projet, un manque d'optimisation fut la mise en place des traductions. Outre l'aspect base de données sur lequel nous reviendrons plus loin, pas mal de temps fut consacré à ajouter des balises *{% trans %}* et *{% endtrans %}* dans les fichiers Twig. En prenant préalablement connaissance du fonctionnement de Symfony vis-à-vis des traductions, ces balises auraient été placées anticipativement permettant de gagner en temps et en efficacité.

S'il n'est pas possible de maitriser tout un Framework de A à Z, pour des fonctionnalités importantes comme les traductions, le temps réel…, il est judicieux de regarder la documentation au tout début du projet afin d'avoir une idée dont le Framework fonctionne et ainsi optimiser son temps de travail.

7.2.2 Le « piège de Symfony »

Le choix de Symfony comme Framework me semblait être une bonne possibilité de découvrir et approfondir ses fonctionnalités. Au début, le site a été créé avec la même logique que le Framework : une route pour le listing, une route pour créer, une route pour modifier, une route pour afficher et une route pour supprimer. Evidemment, cela fonctionne bien et répondait parfaitement au cahier des charges établis (exemple : créer un produit pour un apiculteur).

Pourtant, en procédant de la sorte, l'expérience utilisateur est rendue moins agréable, moins fluide. Le chargement d'une nouvelle page ou l'édition d'une entrée nuit à l'expérience utilisateur. C'est pourquoi, une partie du code a été modifié afin d'insérer des modales et utiliser des requêtes ajax pour la création ou l'édition d'un évènement. Par exemple, pour Openings, la modification, l'ajout et la suppression d'un évènement se fait directement sur la même page. L'idée de procéder par requête ajax n'alourdit pas énormément le code de la page. Cela rend l'expérience utilisateur beaucoup plus agréable en termes de temps de réponse. La même approche a été utilisée pour la création et l'édition des articles de la newsletter.

C'est une notion qui aurait dû être développée dans le cahier des charges.

7.3 Librairies et bundles

7.3.1 La connaissance des technologies

Un autre problème rencontré est le simple choix d'installer ou non un bundle ou une librairie tierce. Il est clair que certaines librairies sont tellement bien développées qu'il aurait été peu professionnel de ne pas les utiliser. C'est le cas pour des librairies comme Chart.js, Fullcalendar, JQuery… qui sont continuellement améliorées. La grande communauté autour de ces librairies (et la documentation

associée) permet de mettre rapidement en place un produit avec une grande modularité. Un développement manuel demanderait un temps bien plus important sans pour autant obtenir un produit aussi performant, efficace et évolutif.

Cependant, lors de la mise en place du système de don, l'idée fut de placer un input de type « range » avec un output où la valeur changeait avec le range. La première idée fut d'utiliser JQuery, pour le faire simplement. Pourtant, d'après la documentation, il est nécessaire d'utiliser JQuery UI pour ce type de fonctionnalité. Dans ce cas, l'installation de JQuery UI n'a pas été envisagée étant donné qu'un simple bout de code en Javascript fait largement l'affaire. L'installation d'une librairie tierce doit se faire uniquement si cette librairie est entièrement exploitée par l'application.

7.3.2 L'interface d'administration

L'interface d'administration a été faite « maison » sans utiliser le bundle EasyAdmin de Symfony. Les questions par rapport à l'utilisation d'un Bundle sont souvent les mêmes :

- Le bundle apporte-t-il une facilité importante lors du développement de l'application ?
- Le bundle offre-t-il une grande flexibilité ?
- Le bundle est-il conseillé par la communauté Symfony ?
- Quel est l'impact de l'utilisation ou la non utilisation d'un Bundle sur la sécurité du site ?

Dans le cas d'EasyAdmin, il y a peu de contraintes liées à l'utilisation du bundle et cela entraine une facilité lors du développement. Cependant, le choix de ne pas l'utiliser n'est pas non plus un problème étant donné qu'au niveau sécurité, il utilise le même type de restrictions que sur les autres parties du site.

Finalement, dans un tel cas, l'utilisation du Bundle est plus un choix personnel qu'un réel gain de temps ou un gain au niveau de la sécurité.

7.3.3 Les traductions en base de données sans translatable

Le point le plus complexe lors du développement du site a été la gestion des traductions en base de données. Dans un premier temps, l'idée était de créer des catégories de produits dans une langue spécifique et lui attribuer une clé. Ainsi, les catégories de produits Miel – Honing – Honig – Honey auraient été liées par la clé miel-key. Avec la clé, il était possible, lors d'une recherche, de récupérer tous les produits quel que soit la langue de la catégorie. Ainsi, la recherche sur Honey permettait de récupérer tous les miels.

Cependant, cette approche amène deux problèmes. Le premier est le changement de langue sur une page représentant une catégorie de produits. Dans ce cas, Doctrine va récupérer l'entité dans la langue initiale et non dans la nouvelle langue. Cela aurait pu être corrigé en chargeant un menu différent sur la page (un peu barbare mais fonctionnel).

Le second problème est le changement de langue lors de la création ou de l'édition d'un produit. Un apiculteur créant un produit en français, aurait été confronté à un problème s'il souhaitait revenir sur ce produit en anglais. Le produit étant associé à une catégorie en français, la vue aurait renvoyé le nom de la catégorie en français.

La solution envisagée a été la création d'une entité « abstraite » liant une même catégorie de produits dans des langues différentes. Un peu comme l'extension Translatable de doctrine. En procédant de la sorte, on n'affiche non pas une catégorie particulière sur une page représentant une catégorie de produits mais une catégorie « abstraite ». En filtrant par la langue, il est ensuite possible d'afficher l'ensemble des données quel que soit la langue demandée. En outre, cela fonctionne au niveau des pages des catégories de produits mais également au niveau du formulaire d'édition ou de création d'un produit.

En parallèle, un autre problème est survenu dans la création des produits. En effet, un produit est lié à une catégorie de produits abstraite. Cependant, au niveau de la vue, il ne faut pas afficher les catégories abstraites mais les catégories de produits en fonction de la langue utilisateur. Dans une même logique, un produit est lié à une ou plusieurs sous-catégories abstraites. Au niveau de la vue, il ne faut pas afficher une sous-catégorie abstraite mais les sous-catégories en fonction de la langue utilisateur. En outre, afin de rendre l'expérience utilisateur agréable, le champ sous-catégories se met à jour en fonction de la catégorie sélectionnée (via un FormEventListener).

Le fait de lier un produit à une catégorie de produit abstraite n'a pas posé de problème. C'est au moment de récupérer les sous-catégories en fonction de la catégorie et de la langue que pas mal de problèmes sont survenus. En utilisant directement les entités abstraites, afficher les noms dans la langue utilisateur ne fonctionne pas. Si on utilise un ChoiceType plutôt qu'un EntityType dans le formulaire, on peut récupérer les éléments dans la langue voulue mais si on utilise un FormEventListener pour afficher les sous catégories en fonction de la catégorie, Doctrine remarque que l'entité n'est pas la bonne et cela ne fonctionne pas. En simple, le fait de vouloir afficher des informations concernant une catégorie de produits dans un champ prévu pour les catégories de produits abstraites (et idem pour les sous catégories) pose un réel problème dans la mise en place du formulaire et de sa gestion par doctrine.

La dernière idée, qui a été développée, est d'utiliser des champs non persistés en base de données afin de faire la jonction entre les catégories abstraites et non

abstraites. Ainsi, le formulaire propose à l'utilisateur de choisir une catégorie et une ou plusieurs sous catégories de produits. Une fois le formulaire validé, le Controller se charge de transformer la catégorie et les sous-catégories en catégorie et sous-catégories abstraites. Cela demande plus de travail au Controller et diminue la vitesse de traitement mais c'est la seule alternative trouvée afin de maintenir l'EventListener au niveau des sous-catégories. Ainsi, l'apiculteur ne peut pas sélectionner une sous-catégorie qui n'est pas liés à la catégorie de produits tout en pouvant afficher son formulaire de création ou d'édition de produits dans n'importe quelle langue.

La mise en place des traductions aurait été possible avec le Bundle de Doctrine translatable. Pourtant, après avoir regardé la documentation, cette solution a semblé complexe pour finalement réaliser quelque chose qui semblait relativement « simple ». D'ailleurs, le nombre d'étoiles et d'utilisation sur Github sont relativement faibles (799 étoiles et 892 utilisateurs). C'est avec l'évolution du projet et la mise en place de la structure finale de la base de données que les problèmes sont apparus.

La solution est-elle parfaite ? Respecte-t-elle certaines règles de bonnes pratiques ? Je n'en suis pas sûr à 100% mais elle est fonctionnelle et permet d'avoir une expérience utilisateurs agréable. Malgré les problèmes rencontrés, à aucun moment je n'ai eu l'impression qu'utiliser le Bundle aurait été plus simple (à tort ou à raison).

7.4 Le temps réel avec Mercure

Une fonctionnalité que je n'ai pas encore réussi à implémenter est le temps réel dans les conversations. Il serait intéressant d'avoir du temps réel afin de ne pas devoir rafraichir la conversation par du javascript toutes les minutes. Malheureusement, malgré la documentation, la lecture de différents articles et la vision de différents tutoriels, je n'ai jamais réussi à lancer Mercure.

Le problème ne se trouve pas vraiment au niveau du code mais au niveau de la machine. Beaucoup de tutoriels proposent d'utiliser Mercure avec Linux. Ma machine locale fonctionnant sur Windows, je n'ai jamais réussi à le lancer. D'ailleurs, même la documentation de Symfony rester évasive sur l'utilisation avec Windows et ce contente d'un simple « sur Windows : » sans instructions.

Un autre problème à prendre en compte serait le passage en production. En effet, les serveurs étant généralement sous Linux, même si j'arrive à faire fonctionner le hub de Mercure en local, cela fonctionnera-t-il en production ?

Une alternative serait d'utiliser les « websockets » mais Symfony conseil l'utilisation de Mercure. C'est pourquoi je réfléchis toujours à la façon dont je pourrais l'utiliser.

7.5 Les règles de bonnes pratiques

Lors du premier développement d'un projet aussi grand, certaines questions finissent par être redondantes :

- Les règles de bonnes pratiques sont-elles respectées dans le code ?

  Dans l'entité apiculteur, il n'est pas possible de récupérer les commentaires via l'apiculteur lui-même. Lorsqu'une relation est créée entre des tables, Symfony propose d'insérer un champ dans l'entité ciblée afin de pouvoir accéder à l'entité propriétaire. Dans un premier temps, l'entité apiculteur contenait donc un champ commentaires. Dans la page d'un apiculteur, les commentaires étaient affichés 4 à 4 en retirant simplement la classe « hidden » au fur et à mesure des demandes. Cependant, cela alourdissait la page car un apiculteur avec 1000 commentaires se retrouvait avec 996 commentaires présents dans le fichier html sans être affichés. L'idée a alors été de faire des requêtes ajax afin de les charger 4 par 4.

  Si cela fonctionne, lorsque le Controller récupère l'apiculteur, il récupère également tous les commentaires. Donc, même s'ils ne sont pas affichés, le serveur traite ces données. A ce stade, deux possibilités ont été envisagées : créer une requête SQL qui me permet de récupérer tous les champs sauf les commentaires ou bien ne pas mettre de champ « commentaires » dans l'entité apiculteur. La deuxième solution a été retenue car elle semble être la plus logique.

  Lorsqu'une entrée présente en base de données est appelée, l'ensemble des champs lié à cette entrée doit être récupéré car susceptible d'être utilisé. D'ailleurs, aucune documentation ne propose de récupérer un utilisateur, sans récupérer le mot de passe. Récupérer l'ensemble de l'entité semble être la norme.

  Cependant, lors de la mise en place du système d'étoile, il a fallu récupérer les moyennes de chaque apiculteur et le nombre de commentaire. Cela fonctionne également via des requêtes Ajax. Cependant, dans le listing des apiculteurs, les étoiles s'affichent au fur et à mesure des requêtes Ajax. Cela doit être un avantage au niveau expérience utilisateur car la page s'affiche plus rapidement (les calculs des moyennes et du nombre de commentaire n'ont pas ralenti l'affichage) mais l'affichage complet de la page est probablement plus lent (à cause des requêtes Ajax successives).

- Le code est-il réutilisable ?

  A différents endroits du site, des boutons switch liés à des requêtes Ajax ont été utilisés pour améliorer l'expérience utilisateur (stock, publication d'un article, bannissement…). En paramètres de mes « switch » un lien permettant de faire la requête vers la bonne route a été placé. Le Controller

se charge de vérifier les autorisations mais du coup, le fichier javascript est réutilisable.

En parallèle, la même méthodologie a été adoptée pour la soumission des formulaires en javascript (articles et agenda). La soumission de ces deux formulaires renvoie vers le même fichier javascript. En fonction de la route à suivre, il fait les vérifications nécessaires. On peut dès lors se poser la question de réutiliser un tel fichier. Vaut-il mieux faire faire deux fichiers javascript distincts ? Vaut-il mieux faire faire un fichier générique qui en fonction de la route importe un fichier javascript propre à la route ? A ce stade, aucune information concernant les règles à suivre n'a été trouvée et cela est peut-être simplement un choix personnel.

- Les fichiers sont-ils bien organisés ?

La façon dont les fichiers ont été organisés semble cohérente. Mais si quelqu'un reprenait le projet, aurait-il aussi facile de s'y retrouver que sur un autre projet ?

- Les classes sont-elles bien nommées ?

Certaines classes auraient pu avoir un nom plus judicieux. Par exemple, l'entité « Openings » qui représente les heures d'ouvertures de la vente directe aurait pu avoir un nom plus générique comme « Event ».

8. Conclusion

En tant qu'annuaire, « miel-belge.be » rempli sa fonction et propose plusieurs fonctionnalités permettant à la fois d'enrichir le site et d'améliorer l'expérience utilisateur.

Lorsque j'ai imaginé « miel-belge.be » la première fois, j'imaginais avoir, à la fin, un site terminé. Pourtant, au fur et à mesure du développement du projet, je me suis rendu compte qu'il n'aurait probablement pas de fin au sens développement du terme.

N'ayant jamais travaillé dans le web, je me suis toujours posé la question de savoir ce que l'on entendait par maintenance d'un site. Au terme de ce travail, j'en ai la réponse. Le site « miel-belge.be » ne sera jamais terminé car il y aura toujours une idée, une nouveauté et une fonctionnalité à développer. Que ce soit pour améliorer l'expérience utilisateur, pour avoir un bon référencement, pour enrichir le site ou, tout simplement pour apprendre à utiliser de nouvelles technologies, le travail de maintenance est un vrai challenge.

Les fonctionnalités Messenger et Mercure ne sont pas encore implémentées. Elles permettraient d'enrichir l'expérience utilisateur en diminuant le temps d'attente (Messenger) ou via du temps réel (Mercure).

Outre les deux fonctionnalités citées ci-dessus, il serait possible de développer les statistiques, améliorer le design, proposer un mode « dark » ou encore travailler au développement d'une petite application (Android / …) permettant de recevoir des notifications (application en JAVA). Cela demanderait de maitriser d'autres techniques mais pourrait offrir un vrai plus à l'utilisateur.

L'utilisation de Symfony pour développer l'application est un choix que je ne regrette pas. Le Framework est vraiment très fonctionnel, avec une documentation riche, claire et variée. Les questions que je me suis posées ont très souvent été postées préalablement sur Stackoverflow et ont trouvé des réponses pertinentes.

9. Bibliographie

1. **Agnès Fayet**. 2019/2020. *L'apiculture en Wallonie 2020, Contexte, analyse et pistes d'actions*, p. 12 à 35. Disponible sur : https://www.beewallonie.be/wp-content/uploads/2020/10/BW-Analyse-du-secteur-apicole-2019-fin.pdf (consulté en mai 2021)

2. **Parlement Européen**. 2018. *Le marché du miel dans l'Union européenne (infographie)*, Disponible sur : https://www.europarl.europa.eu/news/fr/headlines/economy/20180222STO98435/le-marche-du-miel-dans-l-union-europeenne-infographie (consulté en mai 2021)

3. **Etienne Bruneau.** 2008. *Structure du secteur miel en Région wallonne et à Bruxelles (francophone*), Disponible sur https://www.cari.be/article/structure-du-secteur-miel-en-region-wallonne-et-a-bruxelles-francophone/ (consulté en mai 2021)

4. **Symfony SAS**. 2005-2021. *Symfony Documentation*. Disponible sur https://symfony.com/doc/current/index.html (consulté en mai 2021)

5. **Fabien Potencier**. 2020. *Symfony 5 : The Fast Track (En route pour Symfony 5)*. Disponible sur https://symfony.com/doc/current/the-fast-track/fr/index.html (consulté en mai 2021)

6. **Nouvelle-Techno.fr**. 2016-2021. *Création de sites web - Tutoriels en ligne - Formateur indépendant (tutoriel)*. Disponible sur https://nouvelle-techno.fr/ (consulté en mai 2021)

7. **Grafikart.fr**. 2021. *Tutoriels en ligne (tutoriels).* Disponible sur https://grafikart.fr/ (consulté en mai 2021)

8. **Vladimir Agafonkin**. 2010-2021. *Leaflet API reference (documentation).* Disponible sur https://leafletjs.com/reference-1.7.1.html (consulté en mai 2021)

9. **FullCalendar LLC**. 2021. *Getting Started (documentation)* https://fullcalendar.io/docs/getting-started (consulté en mai 2021)

10. **Chart.js contributors**. 2014-2021. *Chart.js documentation.* Disponible sur https://www.chartjs.org/docs/latest/ (consulté en mai 2021)

10. Annexes

10.1 Les fonctionnalités sprint par sprint (annotées)

1. Affichage des données (durée estimée : 3 semaines)
    - Listing catégories de produits (page catégories et menu)
    - Listing apiculteurs (page apiculteurs)
    - Bouton de recherche (3 critères) (toutes les pages)
    - Afficher une catégorie

Priorité plus faible, il faut mettre l'accent sur les apiculteurs (passe en niveau 5 avec la création des produits)

    - Afficher un apiculteur
    - Afficher un résultat de recherche
    - Inscription : authentification et modification

2. Demande d'inscription (durée estimée : 2 semaines)
    - Inscription
    - Connexion et déconnexion
    - Modification
    - Mot de passe (modification ou oubli)
3. Formulaire d'ajout / suppression / modification de produits pour un apiculteur (durée estimée : 1 semaine)

    Passe en niveau 5 avec les catégories de produits.

4. Image (durée estimée : 2 semaines)
    - Photo de profil
    - Photos de l'apiculteur
5. Visualisation des apiculteurs sur une carte (durée estimée : 2 semaines)

Priorité plus élevée (niveau 3 car c'est le but du site)

6. Gestion des traductions (durée estimée : 1 semaines)

Doit-être réfléchi et développé au fur et à mesure des Template et des entités

7. Don (durée estimée : 1 semaine)
8. Agenda (durée estimée : 2 semaines)

9. Interface administrateur (durée estimée : 2 semaines)
    - Gestion des utilisateurs
        - Apiculteurs
        - Consommateurs
    - Gestion de la newsletter
    - Gestion des commentaires
    - Gestion des commandes
    - Gestion des dons
    - Catégories de produits

Il est plus agréable de développer l'interface d'administration au fur et à mesure des entités (meilleure vision de l'entité)

10. Formulaire catégories de produits (1 semaine)
11. Réservation (2 semaines)
12. Newsletter (2 semaines)
13. Commentaires (2 semaines)
    - Pour un apiculteur et pour un produit
    - Signaler un abus / problème

Priorité plus élevée car enrichit le site et améliore l'expérience utilisateur (priorité 9)

11. Favoris et géolocalisation par le navigateur

Ne pas oublier de passer en prod car il risque de ne jamais être complètement fini…

14. Optionnel : mise en production
15. Optionnel : Autres statistiques (nombre de connexions sur la page d'accueil)
16. Optionnel : Commentaire
    - Réponse à un commentaire