

# DOSSIER DE SYNTHESE

Hananel Lamberger

## L'ECOSSE ET SES WHISKIES

Un site présentant le pays où coule le liquide couleur or, et proposant de surcroît une boutique en ligne.



<b>I.</b>	<b>INTRODUCTION.....</b>	<b>4</b>
1.	CONTEXTE.....	4
2.	MON PROJET .....	5
<b>II.</b>	<b>CAHIER DES CHARGES .....</b>	<b>6</b>
1.	PAGE D'ACCUEIL.....	6
2.	LES PAGES DE PRESENTATION DES REGIONS .....	6
3.	LA BOUTIQUE.....	6
4.	LE PANIER ET LES COMMANDES.....	7
5.	LA GESTION UTILISATEUR.....	7
6.	L'INTERFACE ADMINISTRATEUR.....	7
<b>III.</b>	<b>SPECIFICATIONS.....</b>	<b>9</b>
1.	LIBRAIRES ET FRAMEWORK .....	9
2.	LANGUAGES.....	12
3.	OUTILS .....	13
4.	API.....	13
<b>IV.</b>	<b>REALISATIONS .....</b>	<b>15</b>
1.	LE MAQUETTAGE .....	15
2.	LA BASE DE DONNEES.....	15
3.	LES ENTITES ET DOCTRINE.....	16
4.	LA CARTE LEAFLET.....	17
5.	L'INTERFACE ADMINISTRATEUR.....	18
6.	LE PANIER .....	18
7.	LA COMMANDE .....	19
<b>V.</b>	<b>SECURITE ET TESTS.....</b>	<b>20</b>
1.	LA GESTION UTILISATEUR.....	20
2.	LES FORMULAIRES.....	20
3.	LES TESTS .....	22
<b>VI.</b>	<b>RECHERCHES ET EXTRAIT EN ANGLAIS .....</b>	<b>23</b>
1.	RECHERCHES.....	23
2.	EXTRAIT EN ANGLAIS.....	23
<b>VII.</b>	<b>AXES D'AMELIORATIONS .....</b>	<b>27</b>
<b>VIII.</b>	<b>CONCLUSION .....</b>	<b>29</b>



# I. Introduction

## 1. Contexte

Je m'appelle Hananel Lamberger, j'ai 26 ans.

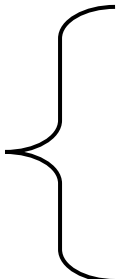
Je suis actuellement en formation de développeur web/web mobile à Elan Formation de Strasbourg avec l'objectif d'obtenir un titre RNCP de niveau 3.

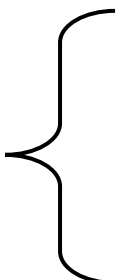
Les premiers mois de la formation, nous avons eu une mise à niveau en PHP et HTML/CSS, nous avons fait de nombreux exercices et appris à manipuler de nombreux outils. Nous avons aussi appris à concevoir et à gérer une base de données relationnelle. Afin de mettre en pratique ce que nous apprenions, nos formateurs nous soumettaient des « mini-projets » à réaliser (un puissance 4, un jeu de rôle, une interface de connexion, une calculatrice en PHP...).

A parti de fin février, nous avons travaillé sur le projet que nous présenterions pour l'examen de fin d'années avec une interruption de trois semaines à partir de fin Mars pour le stage en entreprise.

Nos formateurs nous ont laissé le choix du projet que nous devons réaliser et présenter à l'examen, avec la condition qu'il couvre les compétences requises pour le titre.

Voici les compétences à valider :

- 
- Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité
  - Maquetter une application
  - Réaliser une interface utilisateur web statique et adaptable
  - Développer une interface utilisateur web dynamique
  - Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

- 
- Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité
  - Créer une base de données
  - Développer les composants d'accès aux données
  - Développer la partie back-end d'une application web ou web mobile
  - Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

L'ensemble des compétences est couvert par mon projet en dehors de la partie concernant l'utilisation d'un CMS. En effet, réaliser mon projet avec un CMS ne m'aurait pas permis de couvrir les autres compétences. Cependant, je me suis familiarisé avec WordPress et PrestaShop lors de mon stage en entreprise.

## 2. Mon projet

Je suis un grand amateur de whiskies et en particulier des single malts écossais. Je n'ai pourtant malheureusement encore jamais eu la possibilité de visiter ce pays. De là vient mon idée de « visiter » le pays à distance en créant un site consacré à l'Ecosse et à ses whiskies.

En termes de whisky, l'Ecosse se divise en six régions produisant chacune des whiskies différents du fait de leur géographie et de leur histoire respective (ce ne sont pas des régions administratives mais des régions correspondant à un type de whisky, comme on parlerait de bordeaux, de bourgogne ou de côte du Rhône pour le vin). A l'intérieur de chacune des régions il y a des distilleries, qui ont chacune leur particularité et leur histoire, et qui produisent, pour les plus importantes plus de 10 000 litres de whisky par an.

Afin de parvenir à mon double objectif, à savoir, proposer une boutique en ligne et faire visiter l'Ecosse, j'ai choisi d'organiser mon site autour de la géographie du pays. La présentation du pays, ainsi que la boutique sont donc partagées selon les six régions du pays. Cela me permet de faire le lien entre la partie « touristique » et la partie boutique du site. Ainsi, un visiteur naviguant sur la page d'accueil verra une carte de l'Ecosse divisé en six régions. En cliquant sur une région il « visitera » une région et sera invité à voir les distilleries de la région. En sélectionnant une distillerie, il atteindra la page de la boutique proposant les whiskies de cette distillerie.

Ci-dessous, un schéma présentant le cheminement basique d'un utilisateur à partir de la page d'accueil.

Il y a bien sûr aussi une barre de navigation permettant d'atteindre n'importe quelle page du site en deux clics au maximum.



## II. Cahier des charges

Après avoir muri l'idée de mon projet, je me suis fixé un cahier des charges le plus exhaustif possible afin que, par la suite, je puis me concentrer uniquement sur la réalisation sans perdre du temps par cause d'indécisions.

### 1. Page d'accueil

La page d'accueil doit comporter une courte description, une image et une carte interactive. Sur cette carte interactive doivent apparaître :

- Les régions, reconnaissables par une coloration, ainsi que leurs noms
- Des marqueurs représentant les distilleries

-Au clic sur une région, l'utilisateur doit être redirigé vers la page décrivant la région

-Au clic sur une distillerie, un pop-up doit apparaître avec le titre, l'image principale et un extrait de la description de la distillerie. Il doit aussi y avoir un lien permettant de se rendre sur la page proposant la vente des whiskies de cette distillerie.

### 2. Les pages de présentation des régions

Voici les éléments devant apparaître sur ces pages

- L'image principale en bannière
- La description principale et la secondaire si elle est renseignée dans la BDD
- Une galerie d'images
- La liste des distilleries de la région sous forme de liens permettant de se rendre sur la page de vente des whiskies de cette distillerie
- La carte d'Ecosse en plus petit que sur la page d'accueil et n'affichant que les marqueurs correspondants aux distilleries de la région

### 3. La boutique

Les pages de la boutique doivent afficher la liste des whiskies en fonction de la sélection de l'utilisateur.

On doit pouvoir sélectionner :

- Tous les whiskies
- Les whiskies d'une région
- A l'intérieur d'une région, on doit pouvoir choisir via un menu déroulant une distillerie. La page qui s'affichera montrera tous les whiskies de cette distillerie
- Un whisky en particulier ou une marque de façon plus générale via une barre de recherche présente dans chaque page de la boutique

Chaque whisky doit se présenter dans la liste avec :

- Son image (ou une image indiquant que l'image n'est pas renseignée selon le cas)
- Son prix
- Son degré d'alcool
- Une icône permettant de l'ajouter au panier
- Si un whisky est en rupture, l'utilisateur doit en être informé et ne peut pas l'ajouter au panier

Au clic sur un whisky (soit sur l'image, soit sur le titre), on arrive sur la page présentant un whisky. Voici les éléments devant s'afficher sur cette page :

- Image
- Nom
- Description
- Prix
- Lien pour ajouter au panier ainsi que la possibilité de choisir une quantité avant
- Une rubrique « détails » indiquant les caractéristiques renseignées du produit (degré, volume, région, distillerie)

## 4. Le panier et les commandes

L'utilisateur doit pouvoir afficher son panier, le modifier et passer la commande.

Il doit aussi pouvoir consulter l'historique de ses commandes.

## 5. La gestion utilisateur

Voici la liste des opérations que l'utilisateur doit pouvoir effectuer.

- S'enregistrer
- Se connecter (avec option « se souvenir de moi »)
- Se déconnecter
- Modifier et supprimer son compte y compris son mot de passe
- Consulter l'historique de ses commandes

## 6. L'interface administrateur

Le site doit comprendre une interface administrateur permettant à l'utilisateur ayant le rôle Admin de modifier le contenu des pages régions, de modifier ou ajouter des distilleries des whiskies ou des utilisateurs.



Les whiskies et les utilisateurs auront un champs « actif » qui permettra de les « supprimer » du site sans les effacer de la base de données. Il n'est en effet pas envisageable de supprimer de la base de données un whisky qui n'est plus disponible ou un utilisateur qui a supprimé son compte. Ceci pourrait créer des problèmes d'intégrité de la base de données, si une commande est liée à un whisky ou à un utilisateur qui n'existe pas.

En effet, imaginons un utilisateur qui consulte son historique des commandes (ou alors, un visiteur qui aurait conservé ou partagé l'URL direct d'un whisky supprimé). Si un des whiskies qu'il a commandé par le passé était supprimé, il tomberait sur une erreur 500. En revanche, en rendant un whisky « inactif », l'utilisateur peut voir sa commande et même parvenir à la page de de description du whisky. Seulement, il sera informé « cet article n'est plus disponible à la vente ».

# III. Spécifications

## 1. Libraires et Framework

### Symfony

#### Pourquoi Symfony

Afin de réaliser mon projet, j'ai choisi d'utiliser Symfony 4.

Symfony est un Framework PHP orienté objet qui utilise l'architecture MVC.

L'avantage d'un Framework tel que Symfony, c'est qu'il donne un cadre de travail. C'est à dire qu'il oblige à coder et à organiser son code selon son architecture propre. Les fichiers doivent respecter une structure particulière et se trouver au bon endroit. Ainsi, le code est avantageusement découpé, le front et le back sont séparés, et cela favorise la maintenance de l'application, ainsi que la réutilisation du code.

De plus Symfony est un peu comme une « boîte à outils », il comprend de nombreux outils qui permettent de ne pas réinventer la roue à chaque fois et d'aller plus vite.

En effet, la communication avec la base de données, la gestion utilisateur, la sécurité, les formulaires, la gestion de la session et encore beaucoup d'autres éléments se trouvent facilités grâce à Symfony.

Enfin de nombreux bundles peuvent être intégrés à Symfony et permettent encore des outils supplémentaires. Dans mon cas, je me suis servi par exemple de Pagerfanta pour la pagination, EasyAdmin pour l'interface administrateur et de VichUploader pour l'upload de fichier (la liste n'est pas exhaustive).

#### Structure et fonctionnement de Symfony

##### Entités et base de données

Pour commencer un projet Symfony (après avoir conçu le MCD et le MLD), il faut d'abord créer les entités, qui sont en quelque sorte une abstraction de la base de données relationnelle. Pour cela, il y a deux méthodes :

-On peut créer manuellement la base de données et laisser Symfony (grâce à Doctrine) générer les entités à partir de la base de données (il faudra malgré tout vérifier ce qui a été fait, et parfois compléter, comme dans le cas d'une table-association avec un attribut en plus des clés primaires qu'elle contient).

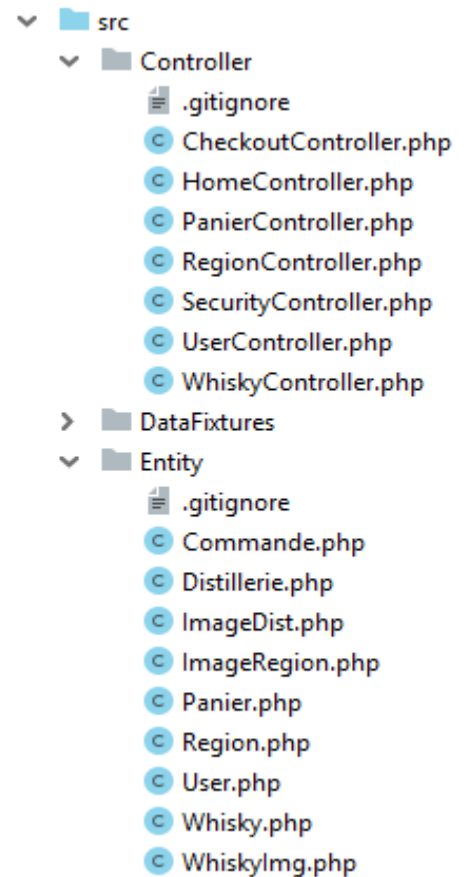
-On peut aussi générer les entités directement sur Symfony, à partir de la console. Cette solution est sans doute plus intéressante car elle oblige à comprendre parfaitement la relation entre les entités et la base de données, ce qui fera forcément gagner du temps par la suite, lorsqu'il faudra modifier quelque chose dans les entités par exemple.

Comme on peut le voir sur l'arborescence de Symfony sur la droite, il y a un dossier Entity qui contient, bien sûr, les entités, et aussi un dossier Repository. Dans ce dossier, il y a autant de fichier que d'entités. Chaque fichier est en lien avec une entité. C'est le fichier du Repository qui fait le lien entre la BDD « physique » et l'abstraction en objet, c'est aussi dans ces fichiers qu'il faudra coder si on veut faire des requêtes complexes dans la base de données.

### Les contrôleurs et les routes

Une fois les entités et les repository créés, on va pouvoir coder les contrôleurs. En général, on commencera par créer un HomeController qui « rendra » des vues dans un nouveau fichier du dossier templates qui s'appellera home.

Ensuite, chaque fonction rendra une nouvelle vue à l'intérieur de ce dossier.



```
namespace App\Controller;

use ...

class HomeController extends AbstractController
{
    /**
     * @Route("/")
     * @Route("/home", name="home")
     */
    public function index(DistillerieRepository $repo)
    {
        $dists=$repo->findAll();
        ...
        return $this->render( view: 'home/index', array('dist'=>$tableau));
    }
}
```

Ainsi, la fonction `index` du `HomeController` (voir ci-dessus) va rendre une vue dans un fichier `index.html.twig` dans le dossier `home`. Dans la fonction, on peut faire passer les variables qu'on veut et elles pourront être récupérées par twig dans le fichier `index.html.twig`. (Dans ma capture d'écran, j'ai supprimé les lignes de codes du milieu afin de ne garder que les éléments que je voulais montrer. On peut voir dans la dernière ligne que je « passe » une variable à ma vue, la variable est `$tableau`, et dans ma vue elle s'appellera « `dist` »).

Parlons à présent du système de routing de Symfony. Au-dessus de la fonction `index`, on peut voir une annotation, c'est elle qui définit la route. La première partie à l'intérieur de la parenthèse (2<sup>e</sup> ligne d'annotation) définit l'adresse dans l'url pour atteindre la page, le « `name` » en deuxième partie fait référence à la même page mais pour être appelé depuis Symfony. Le fait de donner un nom à la route permet de générer des liens hypertextes sans utiliser l'URL de la route. On pourra ainsi par la suite modifier l'URL d'une route dans l'annotation sans avoir à modifier les endroits où elle est utilisée. On peut aussi ajouter des paramètres GET dans l'URL et les récupérer dans la vue, cela se fait très simplement avec Symfony.

## Doctrine

Symfony utilise Doctrine qui est un ORM. Mais qu'est-ce qu'un ORM ?!

Un ORM (Object Relational Mapping), est un moteur qui prend en charge la relation entre la base de données et l'application. L'ORM établit une relation bidirectionnelle entre l'application et la base de données, dans le cadre d'une programmation orientée objet. En effet, les entités peuvent être créées par Doctrine à partir de la base de données, ou inversement la base de données peut être créée par Doctrine à partir des entités de l'application.

Doctrine est en quelque sorte une couche d'abstraction qui se place entre l'application et la base de données. Elle fait le lien entre les entités qui sont des objets PHP et les tables de la base de données.

## Twig

Twig est un moteur de template pour PHP qu'utilise Symfony. C'est-à-dire que lorsqu'une fonction d'un contrôleur rend une vue, celle-ci est créée dans un fichier avec l'extension « `.html.twig` ». Ce fichier ainsi que chacune des vues étend le fichier `base.html.twig` (héritage de template), ce qui a pour premier avantage de ne pas avoir de code répétitif, car il suffit d'entrer dans le fichier `base.html.twig` les données nécessaires à toutes les pages (comme une barre de navigation, le footer ou les liens css et javascript).

Mais le but principal de Twig est de séparer le code PHP du code html – autrement dit le fond de la forme. Twig est en quelque sorte une surcouche d'interprétation de PHP, c'est-à-dire qu'on va coder dans un langage qui devra ensuite être traduit en PHP. En effet la syntaxe pour Twig est plus concise et plus claire, fini les « `< ?php echo $variable ; ?>` », à présent ce sera simplement `{{ variable }}`. On peut aussi écrire du code dynamique avec twig mais là encore, la syntaxe sera simplifiée par rapport à PHP, et directement sécurisé.

# Bootstrap 4

## Qu'est-ce que Bootstrap ?

Bootstrap est un framework CSS, mais pas seulement, puisqu'il embarque également des composants HTML et JavaScript. Il comporte un système de grille simple et efficace pour mettre en ordre l'aspect visuel d'une page web. Il apporte du style pour les boutons, les formulaires, la navigation...

## Pourquoi Bootstrap ?

L'utilisation de Bootstrap permet, grâce à son système de grille et de positionnement, de mettre en place rapidement une organisation propre des éléments sur une page. Mais surtout Bootstrap est responsive, c'est-à-dire que le positionnement des éléments avec Bootstrap sera adapté quel que soit la taille de l'écran sur lequel le site est consulté.

Bootstrap est aussi utile pour apporter du style à des éléments comme la barre de navigation, les formulaires, les boutons...

## jQuery

jQuery est une bibliothèque javascript libre créée pour faciliter l'écriture de scripts. Je m'en suis servi pour les requêtes ajax, et pour coder quelques animations.

# 2. Languages

## HTML5

HTML est le format de données conçu pour représenter les pages Web. C'est un langage de balisage permettant d'écrire de l'hypertexte, d'où son nom. HTML permet de mettre en forme le contenu des pages, d'inclure des ressources multimédias telles que des images, des sons, des vidéos, des formulaires de saisie, et des programmes informatiques. HTML5 désigne souvent un ensemble de technologies Web (HTML, CSS3 et JavaScript) permettant notamment le développement d'applications.

## CSS 3

Les feuilles de style en cascade, généralement appelées CSS, forment un langage informatique qui décrit la présentation des documents HTML. CSS devient couramment utilisé dans la conception de sites web et bien pris en charge par les navigateurs web. Des évolutions récentes de CSS permettent d'inclure des animations aux applications sans utiliser nécessairement de JavaScript.

## Javascript

JavaScript (souvent abrégé en "JS") est un langage de script, orienté objet, principalement utilisé comme le langage de script des pages web. Mais il peut aussi être utilisé sur de nombreux environnements extérieurs aux navigateurs tel que node.js

## PHP

Hypertext Preprocessor, est un langage de programmation libre, principalement utilisé pour Produire des pages Web dynamiques. PHP est un langage impératif orienté objet (Comme C++). PHP a permis de créer un grand nombre de sites Web célèbres, comme Facebook, YouTube, Wikipédia, Google, etc. Il est aujourd'hui considéré comme la base de la création des sites Internet dynamiques.

## 3.Outils

### jMerise

jMerise est un logiciel dédié à la réalisation de MCD (modèle conceptuel de donnée). Il permet aussi de générer le MLD (modèle logique de donnée), ainsi que le script MySQL.

### MySQL

MySQL est un système de gestion de base de données relationnelles. C'est-à-dire que c'est un logiciel qui permet de gérer des bases de données, il utilise pour cela le langage SQL.

### WampServer

WampServeur est une plateforme de développement qui permet de faire fonctionner PHP en local. C'est un environnement qui comprend le serveur Apache, MySQL ainsi que l'interface d'administration phpMyAdmin

### PhpStorm

J'ai choisi de réaliser mon projet avec l'IDE PhpStorm, c'est en effet l'IDE avec lequel je suis le plus à l'aise, et il intègre de plus plusieurs outils très utiles pour la réalisation de mon projet.

En particulier, une console, ce qui permet de lancer des commandes Symfony directement depuis l'IDE, le plugin Emmet permettant une auto-complétions puissante sur le langage HTML, et un plugin Symfony permettant entre autres une auto-complétions des classes.

## 4.API

### Leaflet et le format JSON

#### Qu'est-ce que Leaflet ?

Leaflet est une API (Interface de Programmation applicative) qui propose une bibliothèque JavaScript libre de cartographie en ligne. Concrètement, Leaflet propose un service de cartographie gratuit ainsi qu'une documentation très étoffée permettant un haut niveau de personnalisation.

## Les marqueurs et JSON

La carte que je veux réaliser doit afficher des marqueurs correspondant aux distilleries. Or, les informations sur les distilleries (longitude, latitude, nom, image...) sont stockées en base de données, et le format de donnée exploitable par Leaflet est le format JSON.

Ainsi, dans mon contrôleur qui rend la vue affichant la carte, je récupère sous la forme d'un tableau en PHP les informations qui m'intéressent. Ensuite, je convertis le tableau PHP en tableau JSON pour enfin le faire « passer » à ma vue.

## PayPal

Le service de paiement PayPal propose une API afin d'intégrer un service de paiement en ligne sur un site. L'API permet bien sûr de payer avec PayPal, mais aussi de payer par carte bleue.

# IV. Réalisations

## 1. Le maquettage

La première étape de la réalisation du site fût de maquetter l'ensemble du site. Pour cela, j'ai commencé par établir une liste des pages que je voulais afficher sur le site et des éléments qui devraient y apparaître.

Ensuite, j'ai maqueté les différentes pages plus concrètement avec l'application draw.io.

## 2. La base de données

Voici toutes les tables de ma base de données :

- Région (nom, description1, description2, description3)
- image\_region (region\_id, titre, filename, date\_ajout)
- distillerie (region\_id, nom, latitude, longitude, description1, description2, description3, date\_ajout)
- image\_dist (distillerie\_id, titre, filename, date\_ajout)
- whisky (distillerie\_id, nom, description1, description2, description3, prix, UGS, degré, volume, date\_ajout, actif)
- whisky\_img (whisky\_id, titre, filename, date\_ajout)
- user (email, roles, password, nom, prénom, adresse, ville, code postal, pays, téléphone, sexe, date\_ajout, reset\_token, actif)
- commande (user\_id, date\_commande, contenu, nom\_adresse, adresse, ville, code postal)

Il y a aussi un champ id dans chaque table qui s'auto-incrémente et qui sert d'identifiant. Le champ actif dans la table whisky et user, permet de désactiver un utilisateur ou un whisky sans le supprimer de la base de données.

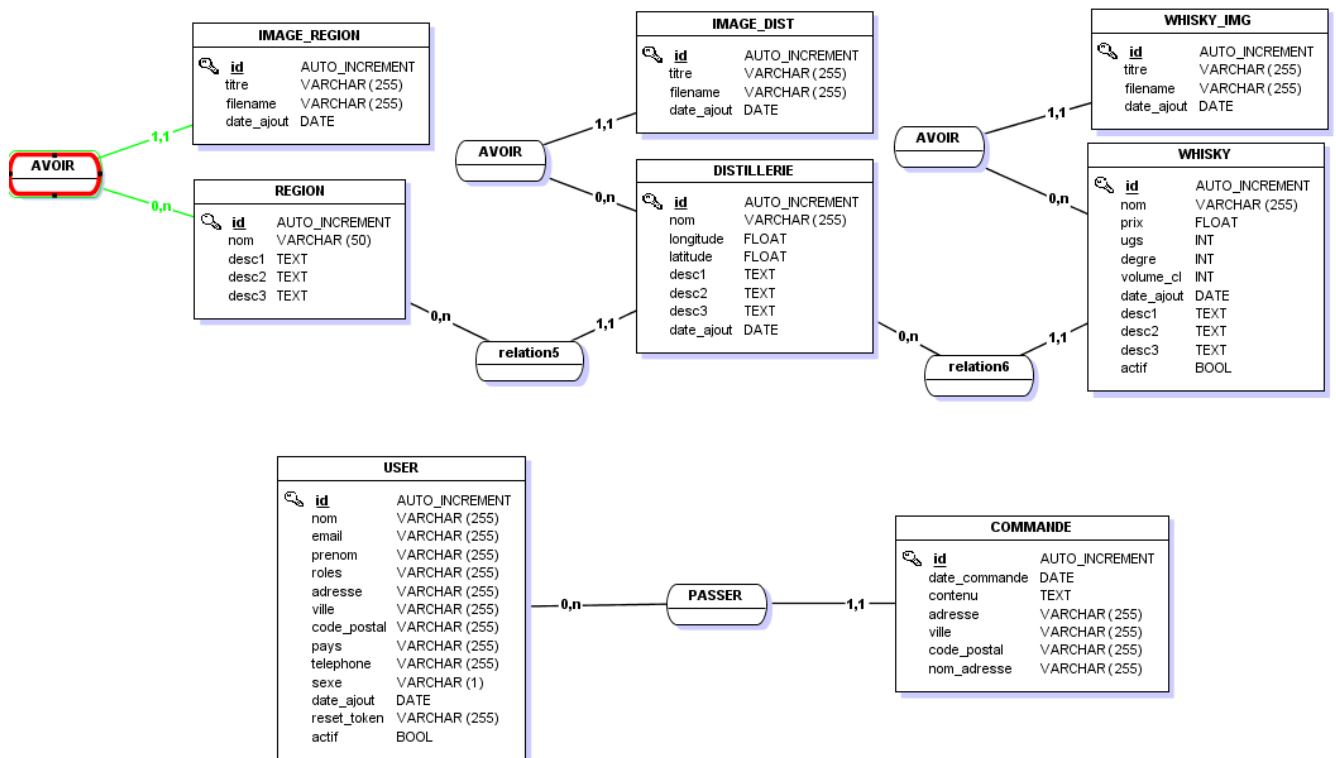
Les relations entre les tables sont les suivantes :

- Les régions, les distilleries et les whiskies ont une collection d'image (many to one)
- Une région a une ou plusieurs distilleries
- Une distillerie a un ou plusieurs whiskies
- Un utilisateur a une ou plusieurs commandes

A noter que la table « commande » n'est pas liée à la table « whisky » en base de données. La commande a simplement un champs « contenu » qui stocke sous la forme d'un tableau au format JSON l'identifiant des whiskies commandés et leurs quantités.

Bien qu'il ne soit pas nécessaire avec Symfony de générer la base de données avec un script SQL (Doctrine se charge de ça). J'ai quand même modélisé la base de données avec jMerise comme ci-dessous afin d'être sûr de son intégrité.





### 3. Les entités et Doctrine

Symfony intégrant Doctrine, il y a une relation bidirectionnelle entre la base de données et l'application. Ce qui permet de générer les entités à partir de la base de données ou inversement de générer la base de données à partir de l'application.

Mon choix a été de créer d'abord les entités (avec l'aide de la console) qui sont des objets PHP pour ensuite faire la migration vers la base de données. Cette méthode m'a paru plus intéressante, car si elle est un peu plus longue, elle me permet de mieux maîtriser les entités et leurs champs.

### Les contrôleurs et les routes avec Symfony

Les pages sont rendus par des fonctions dans le contrôleur. On pourrait en théorie n'avoir qu'un seul contrôleur qui rend toutes les pages du site. Cependant, ce n'est pas conseillé car il y aurait beaucoup trop de fonctions dans chaque contrôleur et le code ne serait pas organisé, donc difficilement lisible.

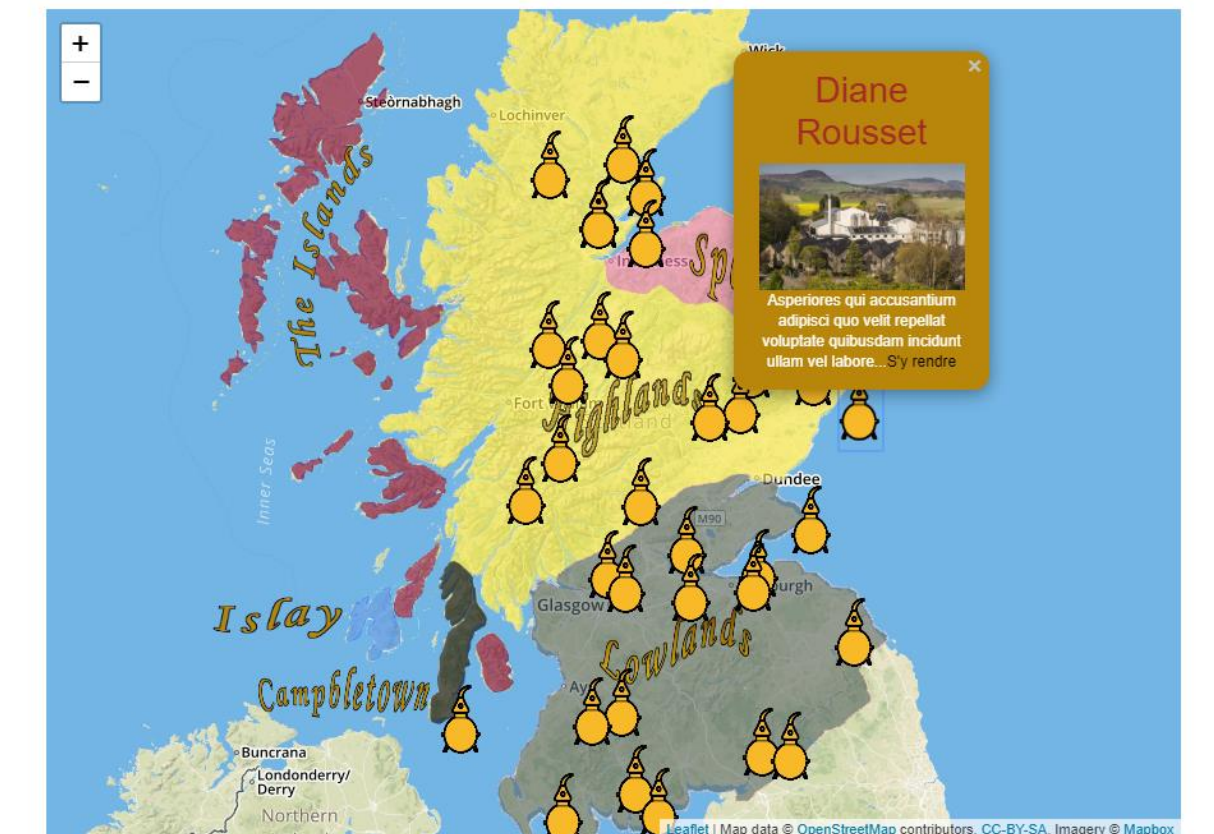
J'ai créé mes contrôleurs en suivant la logique du site :

- HomeController uniquement pour la page d'accueil
- RegionController pour les pages de visite des régions
- SecurityController pour la gestion utilisateur (inscription, connexion, déconnexion, oubli de mot de passe)

- UserController pour les pages personnelles de l'utilisateur (voir, modifier ou supprimer son compte, consulter l'historique de ses commandes)
- WhiskyController pour toutes les pages de la boutique
- PanierController pour rendre le panier et pour gérer toutes les actions sur celui-ci (ajouter, modifier, supprimer)
- CheckoutController pour passer une commande et payer.

## 4. La carte Leaflet

Carte des régions et des distilleries d'Ecosse



- La carte est configurée de manière à ce que le zoom minimum et maximum soient limités à une utilisation adéquate de la carte. La bordure de la carte est aussi limitée afin que l'utilisateur ne puisse pas « partir trop loin ».
- Pour distinguer les régions, une image est superposée sur la carte.
- Les noms de région sont aussi une image superposée, avec un lien vers la page « région » correspondante dans un nouvel onglet.
- Les informations sur les distilleries en base de données me permettent d'afficher dynamiquement des marqueurs pour chaque distillerie. Au clic sur une distillerie, des informations basiques sur la distillerie s'affiche et un lien permet d'aller à la page vendant les whiskies de cette distillerie.

## 5. L'interface administrateur

J'ai développé l'interface administrateur à l'aide du bundle EasyAdmin. L'utilisateur qui a un rôle admin peut modifier toutes les données contenues sur le site.

Retour vers le site

Whisky Ecosse

lambix@outlook.fr

REGION

- Region
- ImageRegion

DISTILLERIE

- Distillerie
- ImageDist

WHISKY

- Whisky
- WhiskyImg

USER

- User

ID	Actif	Nom	Prenom	Email	
82	<input checked="" type="checkbox"/>	ff	ff	john@yopmail.com	Voir Éditer Supprimer
81	<input checked="" type="checkbox"/>	rem	ae	aedena.remy@hotmail.fr	Voir Éditer Supprimer
80	<input checked="" type="checkbox"/>	lamberger	hananel	ma5225il@mail.fr	Voir Éditer Supprimer
79	<input checked="" type="checkbox"/>	lamberger	hananel	aze@aze.fr	Voir Éditer Supprimer
78	<input checked="" type="checkbox"/>		hananel	azer@azer.fr	Voir Éditer Supprimer
77	<input checked="" type="checkbox"/>	modeste	jeanluc	mail@mail.fr	Voir Éditer Supprimer
76	<input checked="" type="checkbox"/>	modeste	jeanluc	jmodeste@hotmail.fr	Voir Éditer Supprimer
75	<input type="checkbox"/>	benhamou	moche	dev@mail.com	Voir Éditer Supprimer
74	<input checked="" type="checkbox"/>	lam	bloch	lambix@outlook.fr	Voir Éditer Supprimer

9 résultats

Précédent Suivant

## 6. Le panier

Mon entité « Panier » est composée d'un tableau contenant les whiskies ajoutés dans le panier avec leurs quantités. Elle contient aussi plusieurs méthodes permettant de gérer les opérations sur le panier et d'obtenir des informations. (Obtenir le nombre d'articles, changer la quantité d'un article, ajouter un article, le supprimer).

La particularité de cette entité est de ne pas être mappée, c'est-à-dire qu'elle n'a pas de liens avec la base de données. En effet, ce n'est qu'au moment de la finalisation de la commande que le contenu du panier doit être stocké et que les stocks doivent être mis à jour.

Je me sers donc de la session pour stocker mon panier jusqu'à la finalisation de la commande.

Il se trouve que Symfony possède un sous-système de session très puissant et flexible, conçu pour permettre la gestion de session via une interface orientée objet.

Ainsi, le panier est géré par l'instanciation d'un nouvel objet Panier qui est directement stocké en session. La modification du panier va donner lieu dans le contrôleur à la récupération de l'objet panier, sa mise à jour, pour enfin le stocker à nouveau en session.

## 7.La commande

Lorsque l'utilisateur finalise sa commande, un nouvel objet « commande » est instancié. Ses champs sont remplis avec le contenu du panier, la date de finalisation, l'utilisateur effectuant la commande, et l'adresse. L'adresse correspond soit à celle de l'utilisateur stockée en base de données, soit à la nouvelle adresse d'expédition renseignée par l'utilisateur.

Enfin, le panier stocké en session est remis à zéro.

# V. Sécurité et tests

## 1. La gestion utilisateur

Symfony intègre un composant de sécurité très complet et sécuritaire.

### Le pare feu

Le pare-feu est le système de sécurité de l'application, c'est lui qui gère l'authentification des utilisateurs. Il peut être configuré afin de n'autoriser l'accès à certaines pages qu'à un certains types d'utilisateur. Par exemple, interdire la page commande à un utilisateur non authentifié ou interdire l'accès à l'interface administrateur à un autre utilisateur que l'admin.

### Les rôles

Chaque utilisateur connecté a un rôle, par défaut, il a au moins le rôle USER. Ce rôle permet de configurer l'accès aux pages du site ou à certaines opérations en configurant le Firewall.

### Les mots de passe

En utilisant, le `UserInterface` de Symfony, la classe `User` a automatiquement un champ `password`, avec un cryptage configurable facilement. Ainsi, c'est la version cryptée du mot de passe qui est stockée en base de données, afin de garantir la sécurité de l'utilisateur.

## 2. Les formulaires

Une partie de la sécurité des formulaires est déjà gérée par Symfony, imaginons par exemple un formulaire basé sur une entité qui aurait un champ « nom » non nullable. Au niveau du front, l'input correspondant aura l'attribut « required ». Si cette protection était détournée, le serveur retournerait une erreur 500.

Pour encore plus de sécurité, Symfony intègre un outil de contrainte de validation. Après avoir chargé la classe correspondante, on peut par des annotations, ajouter une contrainte de validation sur un champ et personnaliser son message d'erreur.

```
/**
 * @var string The hashed password
 * @ORM\Column(type="string", length=255)
 * @Assert\Regex(pattern="/^(?=.*?[a-z])(?=.*?[0-9]).{6,}$/", message="6 caractères minimum dont au moins une lettre et un chiffre")
 */
private $password;

/**
 * @Assert\EqualTo(propertyPath="password", message="Vous n'avez pas entré le même mot de passe")
 */
private $confirm_password;
```

Dans l'exemple ci-dessus, le champ *password* devra valider la Regex définie, et le champ *confirm\_password* devra être égal au champ *password*. Si les champs ne sont pas bien remplis, la page est rechargée et les messages d'erreur correspondant s'afficheront.

Mon site n'a que deux formulaires.

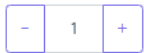
Ceux liés à l'enregistrement ou à la connexion d'un utilisateur (et réinitialisation de mot de passe).

Tous les champs de ces formulaires sont entièrement sécurisés tant au niveau du front que du back.

(Exemple ci-dessous du rendu des contraintes de validation sur la classe User lors de l'enregistrement d'un nouvel utilisateur)

Nom	Prénom	E-mail
<input type="text" value="er"/>	<input type="text" value="fe"/>	<div>ERREUR Email déjà utilisé</div> <input type="text" value="lambix@outlook.fr"/>
Pays	Ville	Code Postal
<input type="text" value="fe"/>	<input type="text" value="fe"/>	<div>ERREUR Cette valeur n'est pas valide.</div> <input type="text" value="ferfe"/>
Adresse		
<input type="text" value="fefe"/>		
Téléphone (facultatif)	Civilité	
<input type="text"/>	<div>Monsieur</div>	
Mot de passe		
<div>ERREUR 6 caractères minimum dont au moins une lettre et un chiffre</div> <input type="password" value="....."/>		
Confirmation		
<div>ERREUR Vous n'avez pas entré le même mot de passe</div> <input type="text" value="Veuillez confirmer votre mot de passe"/>		

Le deuxième formulaire est celui qui permet à l'utilisateur de choisir une quantité avant de mettre le produit au panier. Ce même formulaire est présent dans la page « panier » et permet à l'utilisateur de modifier son panier.



AJOUTER AU PANIER

Voici la liste des vérifications sur ce formulaire :

Grâce à du code javascript, le bouton – ne permet pas de descendre en dessous de 1.

Toujours grâce à une fonction javascript, l'écriture autre que des caractères entre 0 et 9 est impossible.

Si l'utilisateur parvenait à contourner ces protections (par exemple en copiant-collant un nombre négatif ou des lettres), une liste de vérification seraient alors effectuées au niveau du contrôleur :

-Si le nombre est négatif, ou si ce n'est pas uniquement un nombre, ou si simplement le champ est vide, l'utilisateur est renvoyé sur la page avec une notification l'informant que le format ne correspond pas à celui attendu.

-Si l'utilisateur demande un plus grand nombre d'articles qu'il n'y en a de disponibles dans la boutique, il serait redirigé vers cette même page avec une notification l'informant du stock maximum disponible pour ce produit.

## 3. Les tests

### De fausses données

Afin de tester mon site de la manière la plus complète, il me fallait remplir la base de données. Cependant, ajouter manuellement une vingtaine de distilleries et 200 whiskies (ce qui serait un minimum pour simuler un emploi véritable du site) aurait déjà demandé beaucoup de temps.

J'ai donc décidé d'utiliser le bundle permettant d'écrire des fixtures sous Symfony. Les fixtures permettent d'insérer une série de fausses données en base de données afin de tester le site. C'est en fait simplement une boucle qui instancie des nouveaux objets pour ensuite les faire persister en base de données.

En association j'ai utilisé le bundle Faker qui permet de créer des fausses données de façon aléatoires mais avec tout de même un type prédéfini (un nom, un numéro de téléphone, une adresse mail).

J'ai ainsi 79 distilleries dans ma base de données (cela explique pourquoi certaines sont sur l'eau, les coordonnées sont aussi générées de façon aléatoires) et environ 400 whiskies.

### En production

Je n'ai pas eu l'opportunité de mettre mon site en ligne, mais j'ai pu le tester en mode « prod » sur Symfony. Presque tout fonctionnait du premier coup, mais j'ai quand même eu à résoudre quelques bugs mineurs.

# VI. Recherches et extrait en anglais

## 1. Recherches

Pour développer l'interface administrateur de mon site, j'ai utilisé le bundle EasyAdmin. Si cet outil est très facile à mettre en place, le configurer et le personnaliser est une autre affaire. On peut en effet faire un certain nombre de choses facilement, mais des choses qui peuvent paraître très basiques sont parfois compliqués à mettre en place. Dans ce contexte, j'ai passé beaucoup de temps sur la documentation de EasyAdmin afin de réaliser l'interface administrateur que je souhaitais.

L'extrait que j'ai choisi de traduire correspond à la configuration basique du bundle, cependant mes recherches ne se limitent pas à ça et toute la documentation était en anglais.

## 2. Extrait en anglais

### Site d'origine et traduction

Lien de la page :

<https://symfony.com/doc/master/bundles/EasyAdminBundle/book/list-search-show-configuration.html>

### Chapter 5. List, Search and Show Views Configuration

This chapter explains how to customize the read-only views: `list`, `search` and `show`. You'll learn all their configuration options and how to override or tweak their templates.

2.x version

[edit this page](#)

#### List, Search and Show Views ¶

The **List View** displays the items that match the given criteria and provides automatic pagination and column sorting:

The **Search View** displays the results of any query performed by the user. It reuses most of the design and features of the `list` view to ensure a consistent user experience:

The **Show View** displays the contents of a given entity:

### Chapitre 5 : Configuration de la liste, de la recherche et de l'affichage des vues



Ce chapitre explique comment personnaliser les vues en lecture seule (sans modification) : « list », « search » et « show ». Vous apprendrez toutes leurs options de configuration et comment remplacer ou modifier leurs modèles.

## Les vues « List », « search » et « Show »

La vue « List » affiche les éléments qui correspondent aux critères demandés et fournit automatiquement une pagination ainsi qu'un tri par colonne.

La vue « Search » affiche les résultats de n'importe quelle recherche initiée par l'utilisateur. Elle réutilise en majorité l'apparence et les fonctionnalités de la vue « list » pour assurer une meilleure expérience utilisateur.

La vue « Show » affiche le contenu d'une entité donnée.

## General Configuration ¶

In order to make examples more concise, this section only shows the configuration for the `list` view, but you can apply the exact same options to the other `search` and `show` views.

### Customize the Title of the Page ¶

This option refers both to the value of the `<title>` element and to the visible title displayed at the top of the page. By default the title is just the name of the entity. Define the `title` option to set a custom page title:

```
1 # config/packages/easy_admin.yaml
2 easy_admin:
3   entities:
4     Customer:
5       class: App\Entity\Customer
6       label: 'Customers'
7       list:
8         title: "Most recent customers"
9     # ...
```

The `title` value can include the following special variables:

- `%entity_label%`, resolves to the value defined in the `label` option of the entity. If you haven't defined it, this value will be equal to the entity name. In the example above, this value would be `Customers`.
- `%entity_name%`, resolves to the entity name, which is the YAML key used to configure the entity in the backend configuration file. In the example above, this value would be `Customer`.
- `%entity_id%`, it's only available for the `show` view and it resolves to the value of the primary key of the entity being showed. Even if the option is called `entity_id`, it also works for primary keys with names different from `id`.

## Configuration générale

Afin de rendre les exemples plus concis, cette section ne montrera que la configuration de la vue « list », mais vous pouvez appliquer exactement les mêmes options aux autres vues « search » et « show ».

### Personnaliser le titre de la page

Cette option se réfère à la fois à la valeur de la balise « <title> » et au titre visible affiché en haut de la page. Par défaut, le titre est simplement le nom de l'entité. Définissez l'option « title » pour choisir un titre de page personnalisé.

La valeur du champ « title » accepte aussi les variables particulières suivantes :

%entity\_label%, est déterminée par la valeur définie dans l'option « label » de l'entité. Si vous ne l'avez pas définie, cette valeur sera la même que le nom de l'entité. Dans l'exemple ci-dessus, cette valeur serait « customers ».

%entity\_name%, est déterminée par le nom de l'entité, qui est la clé YAML utilisée pour configurer l'entité dans le fichier de configuration du back-end. Dans l'exemple ci-dessus, cette valeur serait « customer ».

%entity\_id%, cette variable n'est disponible seulement pour la vue « show » et est déterminée par la valeur de la clé primaire de l'entité affichée. Même si l'option est nommée « entity\_id », ça fonctionnera aussi avec les clés primaires avec un autre nom que « id ».



In Symfony applications, YAML values enclosed with % and % have a special meaning (they are considered container parameters). Escape these values doubling the % characters:

```
1 # config/packages/easy_admin.yaml
2 easy_admin:
3   entities:
4     Customer:
5       class: App\Entity\Customer
6       label: 'Customers'
7       list:
8         title: '%entity_label% listing'
9       # ...
```

If several entities use the same custom title, you can define the default title for all entities in the global list.title and show.title options (these global titles are always overridden by the title defined by each entity):

```
1 # config/packages/easy_admin.yaml
2 easy_admin:
3   list:
4     title: 'list.%entity_label%'
5   show:
6     title: '%entity_label% (#entity_id%)'
```

⚠ ! Dans les applications Symfony, les valeurs YAML délimitées par « % » en début et en fin de variable ont une signification particulière (elles sont considérées comme des variables avec des paramètres). Échappez ces valeurs en doublant le caractère « % ».

Si vous avez déterminé le même titre personnalisé pour plusieurs entités, vous pouvez définir un titre par défaut dans les options globales « list.title » et « show.title » (ces valeurs seront toujours réécrites par le « title » définie par chaque entité).

## VII. Axes d'améliorations

### Rendre plus dynamique visuellement les pages de visite

Comme je l'ai expliqué au début, l'idée de mon site était de permettre une visite de l'Ecosse. Cependant, je me suis rendu compte lors de la conception des pages de visite, que faire quelque chose qui soit à la fois esthétique, interactif demandait beaucoup de temps. En effet, l'étape seule de concevoir une idée précise du visuel de la page et de ses fonctionnalités demande un travail lourd en tant que tel.

Finalement, après plusieurs essais sans réussir à créer quelque chose qui me plaise véritablement, j'ai choisi de faire quelque chose de simple et lisible. Je n'ai pas voulu perdre trop de temps sur cette partie afin de me concentrer sur la programmation plutôt que de faire un travail qui serait plutôt celui d'un designer web. Cependant, j'aimerais par la suite, proposer une véritable visite de l'Ecosse sur laquelle on pourrait passer plus de temps avant d'arriver à la boutique.

### PayPal

Sur mon site j'ai simulé l'interface de paiement. Le lien PayPal est en fait un bouton « submit » qui finalise la commande (enregistrement de la commande, actualisation de la quantité).

Par la suite, j'aimerais intégrer véritablement la possibilité de payer avec PayPal ou en carte de crédit.

### RGPD

Afin d'être en accord avec les RGPD, l'utilisateur a la possibilité de modifier toutes les informations le concernant et il peut aussi supprimer son compte.

Cependant par souci d'intégrité de la base données (comme je l'ai expliqué dans la sixième partie du cahier des charges), il n'est pas possible de le supprimer totalement.

Il faudrait donc proposer l'option à l'utilisateur de supprimer toutes les données le concernant de la base de données. Lorsqu'il le ferait, les données le concernant (nom, prénom, adresse mail...) seraient remplacées par de fausses données afin de garder un utilisateur avec son identifiant.

### Une interface pour le vendeur

Il y a une chose qui peut sembler évidente mais qui ne m'est apparu que à la fin de mon projet. Il faut une interface pour le vendeur. En effet, l'interface administrative de mon site est très complète et permet de modifier presque tous les éléments du site. Cependant, si elle est très utile pour la maintenance du site, elle n'est pas particulièrement adaptée pour le vendeur.

En effet, beaucoup des possibilités de l'interface administrateur ne sont pas utiles au vendeur, et à l'inverse les fonctionnalités utiles au vendeur doivent être plus normées afin que le vendeur ne puisse pas se tromper lors de l'ajout d'un nouveau produit par exemple.

Voici les fonctionnalités envisagées pour cette interface :

- Notifications lorsqu'il y a une commande
- Gérer le statut d'une commande (en attente, envoyée, reçue)
- Envoi d'un mail automatique lorsque la commande passe au statut « envoyé »
- Possibilité de gérer les stocks des whiskys existants
- Ajouter un nouveau produit
- Importer plusieurs produits à la fois via un fichier CSV

### **Difficultés d'organisation**

J'ai rencontré des difficultés d'organisation et de respect du planning lors de la réalisation de mon projet. En effet, certaines choses que je pensais être rapides m'ont pris beaucoup de temps et inversement j'ai fait très rapidement des fonctionnalités que je pensais longues à coder. Cependant, ces imprécisions sont liées au fait que beaucoup de choses étaient nouvelles pour moi et que le temps que j'ai imparti à chaque tâche n'a pas toujours pu être respecté. Avec le temps et l'expérience, je pense que je serai mieux capable de définir le temps nécessaire à la réalisation d'une tâche et de mieux organiser mon travail.

## VIII. Conclusion

Je n'avais jamais fait de programmation avant d'intégrer cette formation, et je n'aurais pas imaginé apprendre autant de choses en seulement quelques mois.

La réalisation de mon projet m'a procuré beaucoup de plaisir et je suis fier du résultat. Coder avec Symfony fût une expérience très enrichissante. Parfois, ce framework m'agaçait car j'avais l'impression de ne pas pouvoir faire ce que je voulais. Cependant, au fil du temps, j'ai appris que presque tout est personnalisable sur Symfony à condition de respecter son cadre de travail, et de rechercher les solutions sur sa documentation, plutôt que de vouloir se montrer plus intelligent que lui.

Si je suis très heureux de tous ce que j'ai appris cette année, ma curiosité ne s'arrête pas là et j'aimerais approfondir le champ de mes connaissances, peut-être en intégrant une licence professionnelle l'année prochaine .

Enfin, je souhaite remercier l'organisme ELAN et en particulier mes formateurs Mickaël MURMANN et Virgile GIBELLO pour l'apprentissage qu'ils m'ont dispensé et pour le soutien qu'ils m'ont apporté tout au long de la réalisation de mon projet.