DWWM

Marvine Baratto

Rapport de stage

Développeur web et web mobile

Table des matières

[I/ Remerciement 2](#_Toc72053596)

[II/ Résumé de stage 3](#_Toc72053597)

[III/ Points techniques 4](#_Toc72053598)

[IV/ Technologies utilisées 5](#_Toc72053599)

[V/ Présentation de l’entreprise 7](#_Toc72053600)

[VI/ Réalisations 8](#_Toc72053601)

[VI-1/ Projet Office Center 8](#_Toc72053602)

[VI-1-a/ Vue Router 9](#_Toc72053603)

[VI-1-b/ Vuetify 10](#_Toc72053604)

[VI-1-c/ EmailJS solution au problème d’envoi de mail 10](#_Toc72053605)

[VI-2/ Projet OSNOPS 13](#_Toc72053606)

[VI-3/ Projet Boutique du lin 13](#_Toc72053607)

[VI-3-a/ Cahier des charges 13](#_Toc72053608)

[VI-3-b/ MLD 14](#_Toc72053609)

[VI-3-c/ Diagramme de classe 14](#_Toc72053610)

[VI-3-d/ Gestion des listes 16](#_Toc72053611)

[VI-3-e/ Gestion des formulaires 19](#_Toc72053612)

[VI-3-f/ Gestion des commandes 20](#_Toc72053613)

[VI-3-g/ Passage caisse 21](#_Toc72053614)

[VI-3-h/ Paiement 24](#_Toc72053615)

[VII / Conclusion 29](#_Toc72053616)

[VIII/ Bibliographie 30](#_Toc72053617)

# I/ Remerciement

Je tiens à remercier l’AFPA de Dunkerque qui m’a permis de suivre cette formation de développeur web et web mobile, ainsi Martine Poix qui a animé cette cession. Je remercie également la région Hauts de France qui a financé ma formation.

Merci également à Office Center qui m’a permis de réaliser mon stage au sein de son entreprise et qui m’a fourni des projets à réaliser dans le cadre de ma formation.

# II/ Résumé de stage

J’ai effectué mon stage au sein de l’entreprise Office Center qui avait pour projet de créer un site vitrine pour son entreprise de prestations de services. Ce site devait être programmé sous VueJS, et devait comprendre une présentation de l’entreprise ainsi que présenter ses diverses activités. Le but de ce site est de permettre à l’utilisateur de comprendre ce que fournit Office Center rapidement et de la contacter facilement grâce à un formulaire de contact.

Par la suite, j’ai dû m’occuper d’un projet externe à l’entreprise. Ce projet est proposé par l’entreprise OSNOPS, une jeune entreprise qui souhaite être accompagnée par Office Center pour la gestion mais également pour la création de leur plateforme. Ce projet est une application qui doit être réalisée sur Wordpress à la demande du client. Étant sous contrat de confidentialité, ce projet ne sera pas développé dans ce rapport.

Ne pouvant pas parler en détails du projet de mon stage je vais présenter un projet effectué pendant la formation. Ce projet avait pour but de créer une application de gestion de stock, de ventes, de clients ainsi qu’une partie permettant de gérer une caisse pour une boutique de lin.

# 

# III/ Points techniques

|  |  |  |  |
| --- | --- | --- | --- |
| **Développer la partie front-end d’une application web ou web mobile en intégrant les recommandations de sécurité** | | | |
| Projet | Office Center | OSNOPS | Boutique |
| 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** | | | |
| Projet | Office Center | OSNOPS | Boutique |
| 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 oeuvre des composants dans une application de gestion de contenu ou e-commerce |  |  |  |

|  |  |
| --- | --- |
| **Compétences transversales** | |
| Utiliser l’anglais dans son activité professionnelle en développement web et web mobile |  |
| Actualiser et partager ses compétences en développement web et web mobile |  |

# IV/ Technologies utilisées

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Visual studio code | NodeJs | VueJS |
| Visual studio code est un outil d'environnement de développement. | Node.js est une plateforme logicielle libre en JavaScript qui permet l’exécution de JavaScript côté serveur. | VueJS, est un Framework JavaScript open-source utilisé pour construire des interfaces utilisateur et des applications web monopages. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Vuetify | EmailJS | Wordpress |
| Vuetify est une librairie de composants graphiques utilisable sous VueJs. | EmailJS est une API permettant de gérer l’envoie de mail sous VueJS. | WordPress est un système de gestion de contenu gratuit, permettant de créer et gérer son site web. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| HTML | CSS | JavaScript |
| L’HTML est un langage de balisage conçu pour représenter les pages web. | Le CSS est un langage informatique qui décrit la présentation des documents HTML. | JavaScript est un langage de programmation de scripts employé dans les pages web interactives, et est une partie essentielle des applications web dynamiques. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| PHP | WampServer | MySQL |
| Le PHP est un langage de programmation libre qui est principalement utilisé pour produire des pages Web dynamiques. | WampServer est une plateforme de développement Web. C’est également un environnement comprenant trois serveurs (Apache, MySQL et MariaDB), un interpréteur de script (PHP), ainsi que phpMyAdmin pour l'administration Web des bases MySQL. | MySQL est un système de gestion de bases de données relationnelles. |

# V/ Présentation de l’entreprise



Office Center est une SARL qui a débuté son activité en octobre 2012, elle a comme dirigeant Mr. Antoine Boucher. Office Center est localisé à Lille et propose divers services de soutien administratif aux entreprises. Elle permet aux entreprises de se lancer plus sereinement en prenant en charge par exemple la domiciliation ou la comptabilité, elle met également à disposition des locaux en location pour ces entreprises.

En plus de cette prestation de service administratif Office Center se lance dans une nouvelle activité de développement de site web. En effet, elle propose maintenant ses services pour créer des sites web ou applications web pour les entreprises qui le souhaitent.

# VI/ Réalisations

Durant mon stage j’ai été amené à travailler sur 2 projets distincts pour l’entreprise Office Center. Le 1er est un site vitrine pour l’entreprise Office Center développé sous VueJS, le second projet est une application de sponsoring développée sous Wordpress.

## VI-1/ Projet Office Center

À mon arrivée dans l’entreprise, j’ai été assigné à la création du site vitrine d’Office center. Ce site devait être une présentation de l’entreprise et de ces différents services de prestations et d’aide aux particuliers. Des maquettes étaient déjà réalisées pour l’apparence du site et des fonctionnalités présentes.

Le site comprenait donc :

* Une page d’accueil reprenant la présentation de l’entreprise ainsi que ces services
* Une page présentant la liste les différents locaux proposés par l’entreprise
* Les pages présentant en détail chaque local
* Une page sur le service de domiciliation
* Une page sur le service de création de site web
* Un formulaire de contact

N’ayant jamais travaillé sous VueJS, j’ai d’abord commencé par une lecture de la documentation de ce Framework. En suivant la documentation ainsi que d’autres tutoriels, j’ai donc installé un environnement NodeJS afin de créer un serveur et de pouvoir utiliser le gestionnaire de paquet npm, qui facilitera la mise en place du site et l’installation du Framework VueJS et des différentes librairies.

Grâce à cet environnement j’ai pu alors installer VueJS via l'invité de commande Windows et l’utiliser dans le projet avec la commande suivante :

***npm install vue***

Avec tous ces réglages il était alors possible de lancer le serveur grâce à la commande suivante :

***npm run serve***

Le dossier principal (src) du projet est définis de la manière suivante :

* Un fichier main.js contenant toutes les importations et utilisations de librairies, ainsi que l’initialisation de la Vue
* Un fichier App.vue qui définit l’architecture du site et qui affiche les différentes vues
* Un dossier views qui contient toutes les vues du site (ex : Accueil, Formulaire de contact, ...)
* Un dossier plugins contenant les différentes librairies (ex : Vuetify)
* Un dossier components contenant des éléments récurrents du site (ex : Header, Footer, ...)
* Un dossier assets contenant les images utilisées sur le site

### VI-1-a/ Vue Router

Pour pouvoir naviguer à travers les différentes vues du site, il était nécessaire d’utiliser un routeur, j’ai donc utilisé le routeur officiel de VueJS, Vue Router. Pour l’installer j’ai utilisé la commande :

***npm install vue-router***

Ce routeur permet la navigation à travers les différentes vues du site, en indiquant les différentes routes dans un fichier javascript important et utilisant Vue Routeur.

import Vue from 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

On définit les différentes routes dans un tableau de la manière suivante :

const routes = [

{

path: '/',

name: 'Accueil',

component:() => import('../src/views/Accueil.vue')

},

{

path: '/Accueil',

name: 'Accueil',

component:() => import('../src/views/Accueil.vue')

},

{

path: '/Contact',

name: 'contact',

component:() => import('../src/views/contact.vue')

}

.

.

.

]

Ces routes seront alors utilisées dans un VueRouter qui sera utilisé par l’application afin de naviguer dans les différentes vues.

const router = new VueRouter({

mode: 'history',

routes,

scrollBehavior () {

return { x: 0, y: 0 }

}

})

### VI-1-b/ Vuetify

Pour la création des vues détaillant les différents locaux fournis par l’entreprise, des carrousels était nécessaire afin de montrer les différentes photos de ceux-ci. J’ai donc utilisé une librairie permettant d’utiliser des éléments graphiques facilement sous VueJs, Vuetify. Pour l’utiliser il faut l’installer en utilisant le commande :

***npm install vuetify***

Vuetify installé il m’a fallu créer un fichier javascript dans le dossier plugins important et utilisant Vuetify

import Vue from 'vue';

import Vuetify from 'vuetify/lib';

Vue.use(Vuetify);

export default new Vuetify({

});

Avec cette librairie j’ai pu incorporer facilement certains éléments graphiques au projet.

### VI-1-c/ EmailJS solution au problème d’envoi de mail

Le site permet également de contacter l’entreprise via un formulaire de contact. Afin de pouvoir envoyer un mail sous VueJS j’ai effectué des recherches et ai décidé d’utiliser l’API EmailJS.

Présentation en anglais :

EmailJS helps to send emails using client-side technologies only. No server is required – just connect EmailJS to one of the supported email services, create an email template, and use our Javascript library to trigger an email.

Email templates can optionally contain dynamic variables in almost any field (e.g. subject, content, TO address, FROM name, etc) which are populated from the Javascript call. For example, the subject can be "{{ name }}, you have a new message", and using JavaScript the name can be set to "James Dean", for instance.

Additionally, you can easily add attachments, require CAPTCHA validation, switch between the email services without making code changes, review the history of the email request, and more.

Traduction en français :

EmailJS permet d'envoyer des e-mails en utilisant uniquement des technologies côté client. Aucun serveur n'est requis, connectez simplement EmailJS à l'un des services de messagerie pris en charge, créez un modèle d'e-mail et utilisez notre bibliothèque Javascript pour déclencher un e-mail.

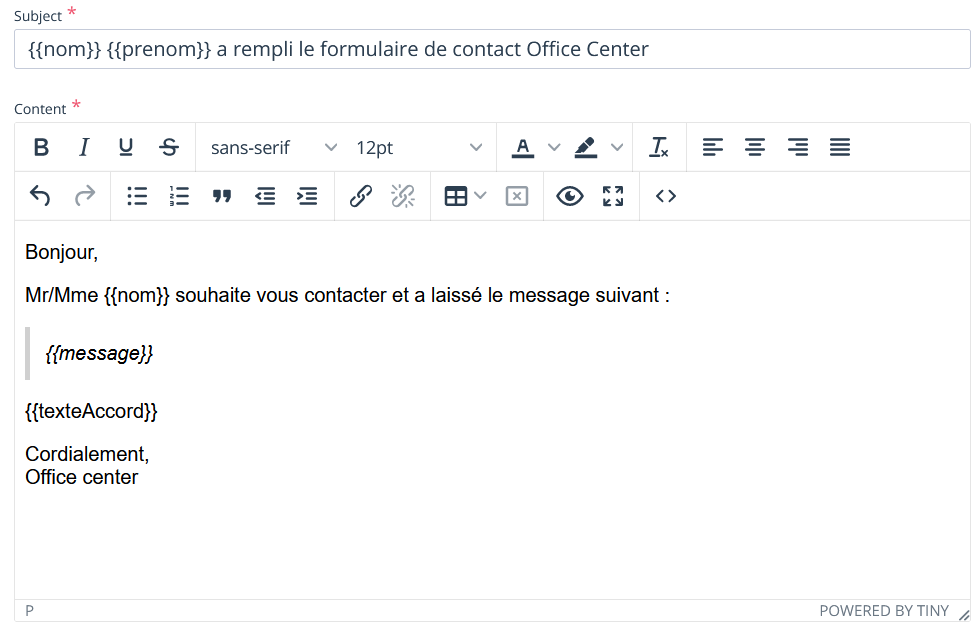
Les modèles de courrier électronique peuvent éventuellement contenir des variables dynamiques dans presque tous les champs (par exemple, le sujet, le contenu, l’adresse de réception, le nom de l’expéditeur, etc.) qui sont remplis à partir de l'appel Javascript. Par exemple, le sujet peut être "{{nom}}, vous avez un nouveau message", et en utilisant JavaScript, le nom peut être défini sur "James Dean", par exemple.

De plus, vous pouvez facilement ajouter des pièces jointes, exiger la validation CAPTCHA, basculer entre les divers services de messagerie sans apporter de modifications au code, consulter l'historique de la demande d'e-mail, etc.

Pour pouvoir l’utiliser il faut installer le kit de développement avec la commande suivante :

***npm install emailjs-com --save***

Il faut ensuite créer un compte sur le site d’EmailJS pour recevoir une clé API et pouvoir alors lier l’adresse mail de réception et créer son template de mail. Pour créer son template de mail EmailJS fournit une interface comme celle-ci :



Elle permet d’écrire ce que l’on veut en texte brut, mais également de récupérer des valeurs issues du formulaire lié à cet envoi et de les afficher. Pour afficher ces valeurs il faut mettre le name de l’input entre 2 accolades.

Par exemple : {{nom}} sera remplacé par la valeur de l’input “nom”.

Coté code, il faut ajouter un événement à la balise “form” quand on envoie le formulaire :



Cet événement appel la fonction “envoiMail” qui est définie plus loin dans le code :



La fonction ligne 59 prend en 1er paramètre l’id de la boite mail enregistré, en 2e paramètre l’id du template de mail, en 3e paramètre le formulaire et en dernier paramètre la clé API fournis à l’inscription. Cette fonction va récupérer les valeurs des champs et les injectés dans le template avant d’envoyer le mail à l’adresse mail enregistrée.

## 

## VI-2/ Projet OSNOPS

Après la réalisation du site vitrine sous VueJS, j’ai été assigné au nouveau projet lié à l’entreprise cliente OSNOPS. Ce projet avait pour contrainte d’être développé sous Wordpress afin de permettre une maintenance de la part du client. Afin de réaliser ce projet, je suis parti sur l’idée de créer un plugin fournissant des shortcodes. Ces shortcodes seront des parties de l’interface plaçable dans une page Wordpress, que le client pourra placer où il le souhaite. Il pourra alors changer l’architecture de son site quand il le souhaite. Cependant il ne m’est pas possible de rentrer plus en détail pour l’explication de ce projet, c’est pourquoi je vais présenter un projet effectué pendant ma formation.

## VI-3/ Projet Boutique du lin

Durant la formation j’ai été amené à travailler sur un projet déjà en cours proposé par la formatrice de la session. Nous étions un groupe de 3 stagiaires à reprendre ce projet, il avait été proposé par La Boutique du Lin à Quaëdypre afin de remettre leur système de vente à jour. Cette boutique propose divers produits en rapport avec le lin et possède une répartition ainsi qu’un référencement très particulier pour ces articles. Le but du projet était donc de réaliser une application répondant au mieux à leurs besoins précis.

### VI-3-a/ Cahier des charges

La Boutique du Lin nécessitait une nouvelle application qui lui permettrait la gestion de ces articles, de ces stocks, de ces clients et d’une caisse. L’application devait disposer de parties distinctes et doit être utilisable sur un écran tactile.

La partie administrateur :

Cette partie de l’application est destinée et est accessible seulement par les dirigeants de la boutique. Elle permet la gestion des diverses tables de la base de données :

* Les articles peuvent être sélectionnés via un filtre puis être modifiés, ajoutés, supprimés, gérer les stocks de l’article …
* Les clients ainsi que leurs achats et paiement sont visibles et gérables depuis cette interface.
* Les utilisateurs (les gérants de la boutique et les caissiers).
* Les différentes ventes, terminées ou non.
* Les diverses informations liées aux articles, par exemple la couleur ou l’univers.

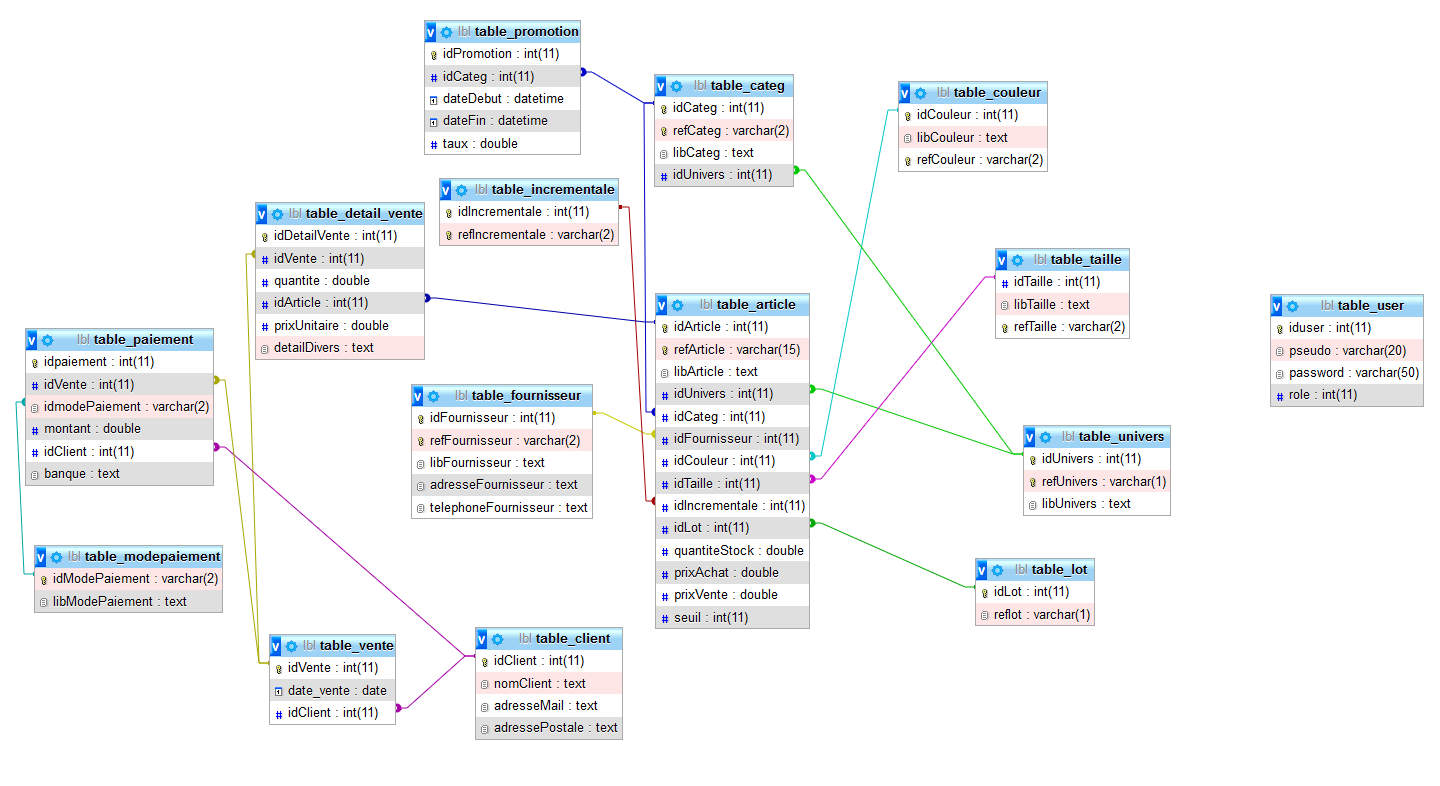
La partie caisse :

Cette partie de l’application est destinée et est accessible par les dirigeants (qui peuvent également gérer la caisse du magasin) et les caissiers. Elle permet aux caissiers d’avoir une interface de caisse leur permettant de gérer une vente. Le caissier peut scanner un article afin de l’ajouter à la commande, effectuer des remises sur l’article ou directement sur la vente globale. Elle permet également de retrouver une commande terminée ou en cours afin de la poursuivre. À la fin de la commande le ticket de caisse est imprimé et peut être envoyé par mail si le client le souhaite.

C’est pourquoi une réunion avec le client avait déjà eu lieu afin de pouvoir réunir ces informations pour établir la base de données.

### VI-3-b/ MLD

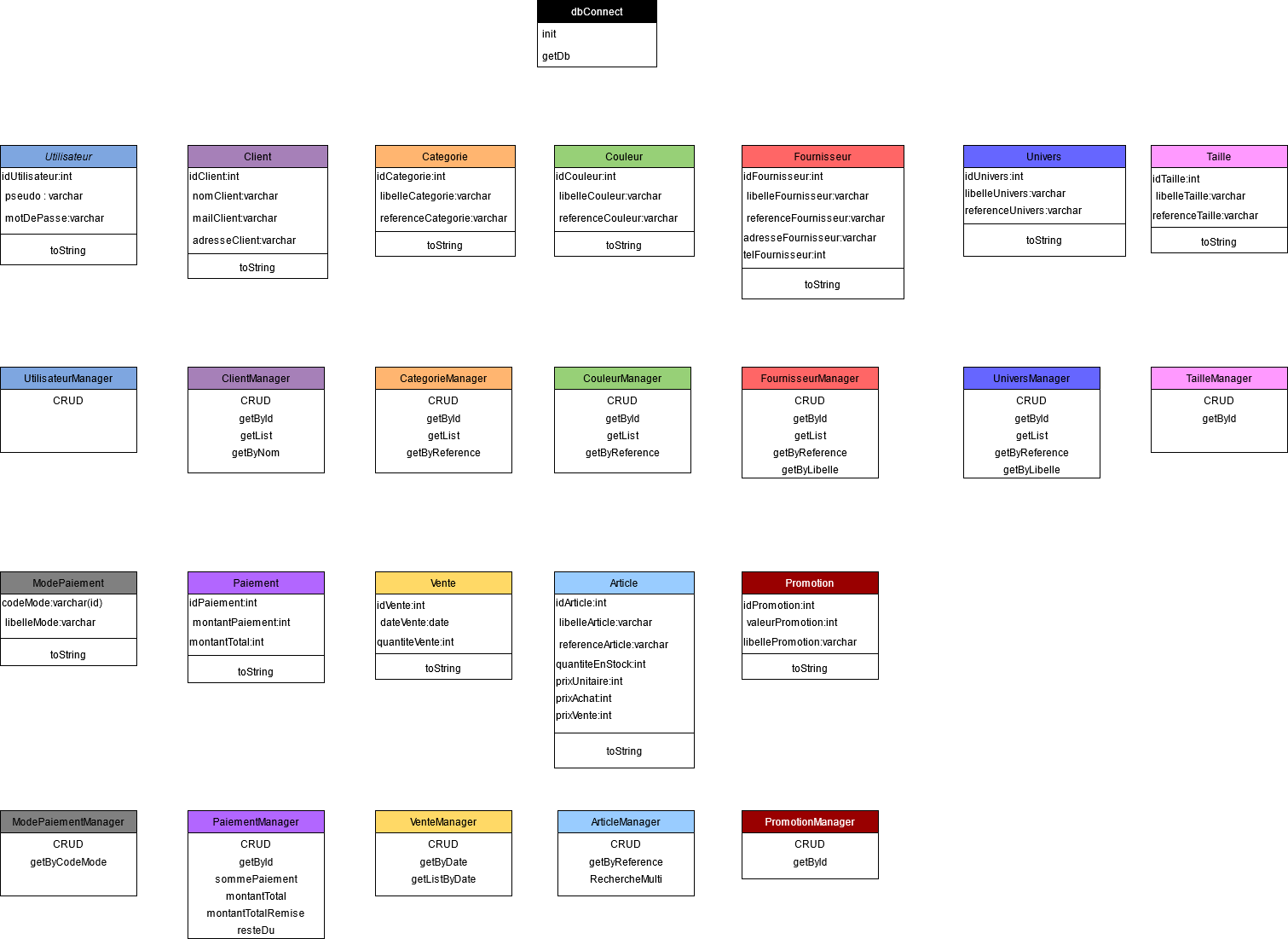
Suite à l’étude de ces informations la base de données suivantes à été établis :



Nous avons donc continué le développement de l’application en suivant et en utilisant cette base de données.

### VI-3-c/ Diagramme de classe

Avec cette base de données nous avons pu établir le diagramme de classe suivant :

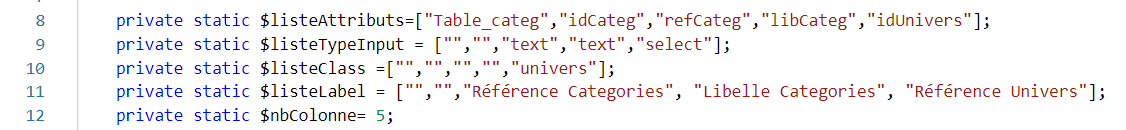


La partie administrateur nécessitant de nombreuses listes de données ainsi que de formulaires, nous avons alors décidé de mettre en place une solution permettant de les construire en fonction de la classe nécessaire. Pour ce faire, nous avons dû rajouter 5 nouveaux attributs pour les classes :

* $listeAttributs : c’est un tableau contenant le nom de la classe et les noms de ces attributs.
* $listeTypeInput : c’est un tableau reprenant les types des inputs (ex: text, date, ...) pour chaque attributs.
* $listeClass : c’est un tableau reprenant le nom des classes des attributs de la classe, s’ils en ont une.
* $listeLabel : c’est un tableau contenant les noms des labels des attributs à afficher.
* $nbColonne : la taille des 4 tableaux précèdent.

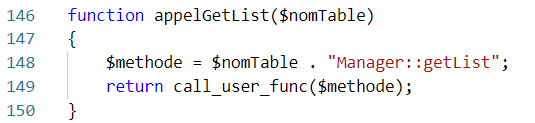
Les valeurs de ces tableaux ne doivent pas être décalé, c’est à dire le nom de l’attribut ($listeAttributs), le type d’input de l’attribut ($listeTypeInput), le nom de sa classe ($listeClass) ainsi que son label ($listeLabel) doivent être à la même position dans tous les tableaux. C’est pourquoi certaines valeurs de ces tableaux sont des string vides.

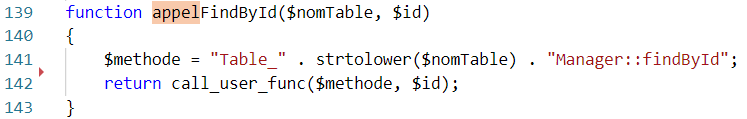
Voici un exemple pour la classe catégorie :

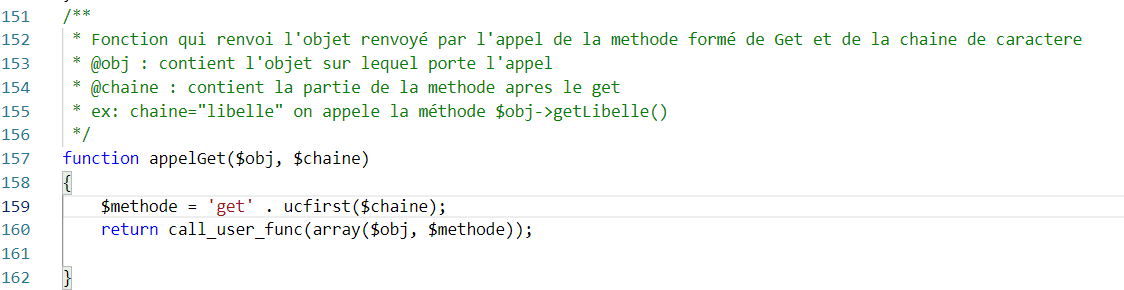


Grâce à ces nouveaux attributs nous avons pu réaliser une seule et même page qui affichera la liste ainsi qu’une autre page qui effectue la même chose pour les formulaires.

Afin de former ces listes pour toutes les tables de la base nous avons créé les fonctions :

* *appelGetList* : qui prend en paramètre le nom de la table et qui appelle la fonction getList du manager de cette table.
* 
* *appelFindById* : qui prend en paramètre le nom de la table et l’id désiré, et qui appelle la fonction findById du manager de cette table.

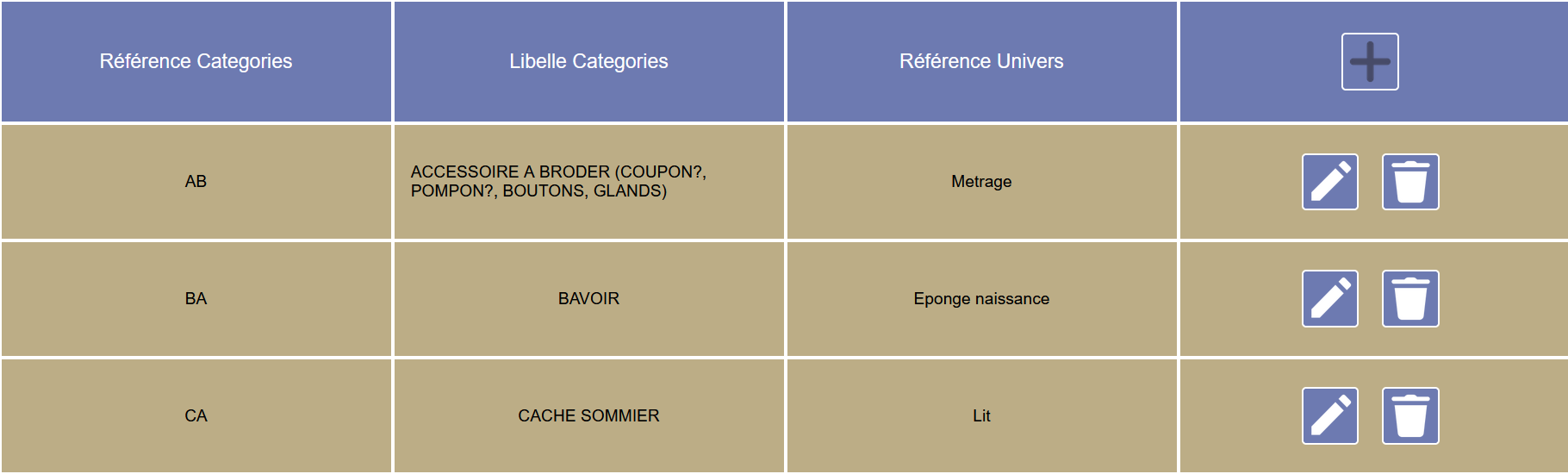


* *appelGet* : qui prend en paramètre un objet et le nom de l’attribut de l’objet que l’on souhaite et qui appelle la fonction get de cet attribut.
* 

En effet, ne sachant pas quel objet la liste ou le formulaire devra afficher, ces méthodes sont indispensables pour la création de ceux-ci.

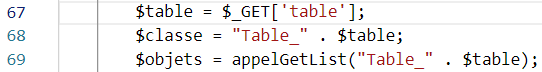
### VI-3-d/ Gestion des listes

Ces listes font partie de la partie administrateur de l’application. Elles affichent la liste des différentes données des tables de la base de données, voici un exemple avec la table catégorie :

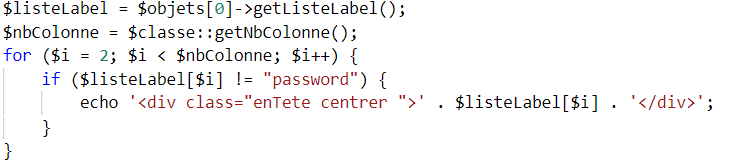


L’utilisateur peut alors cliquer sur le plus afin d’aller sur le formulaire d’ajout, le bouton en forme de crayon pour modifier les valeurs ou le bouton poubelle pour supprimer cette valeur.

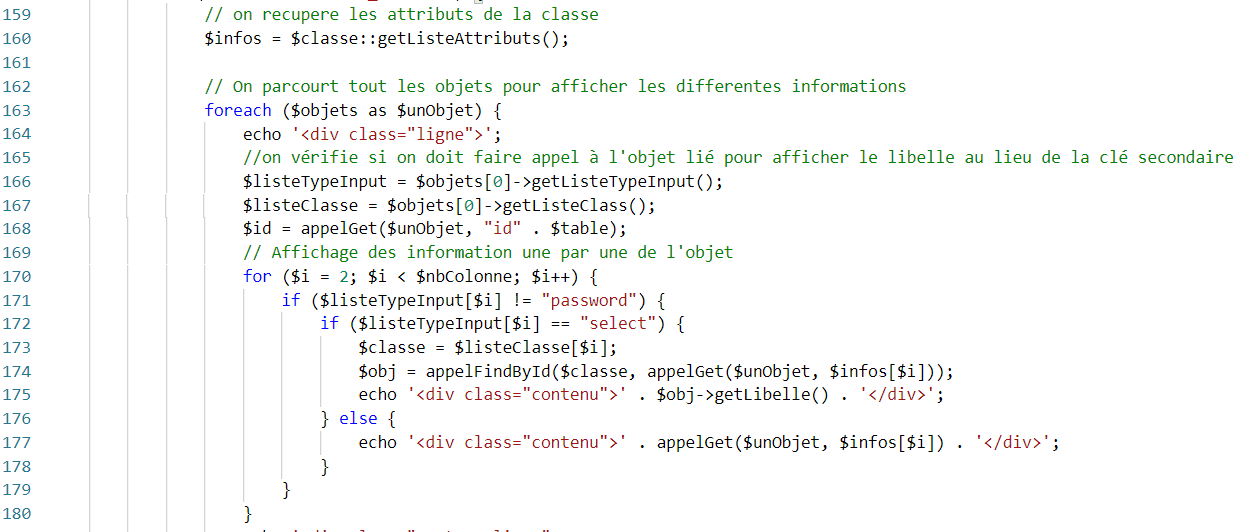
Pour afficher cette liste on récupère le nom de la table grâce au tableau GET, ce qui nous permet alors de récupérer la liste des objets avec la méthode *appelGetList* vue précédemment.



Avec ces informations, il est possible d’afficher les en-têtes de la liste grâce au tableau listeLabel de l’objet :

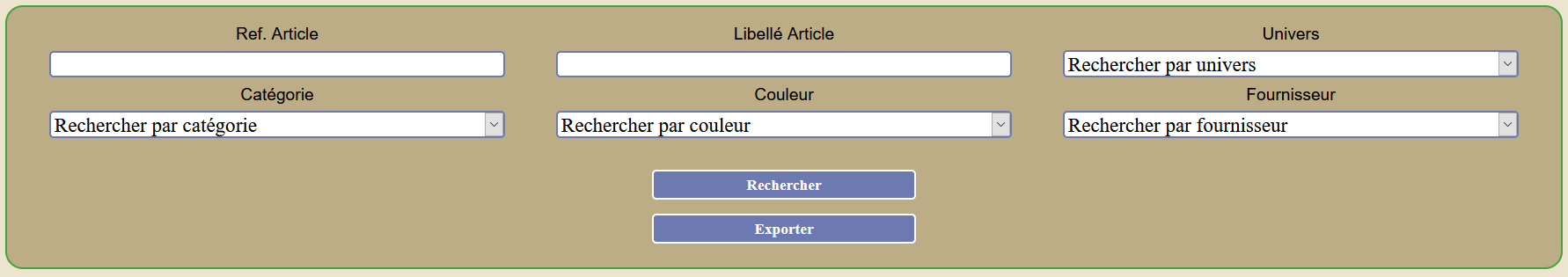


Il ne reste alors qu'à afficher les informations des objets grâce au tableau listeAttributs de l’objet :

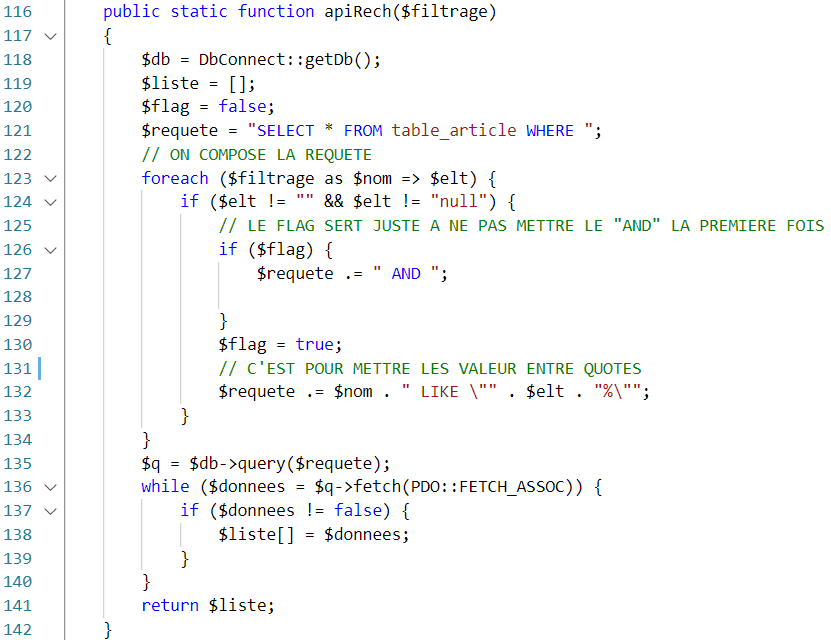


À la ligne 170, la boucle for commence à 2, car les 2 premières valeurs du tableau des attributs sont réservées au nom de la classe et à l’id de l’objet, donc inutiles ici. Enfin à la ligne 172, le if sert à vérifier si l’attribut en cours est un objet afin d’afficher plutôt le libellé de cet objet et non son id.

Tout ceci permet d’afficher toutes les listes des données de l'application en un seul fichier. Cependant, la liste des articles nécessite un champ de recherche avec filtre. C’est pourquoi, si la table est celle des articles, nous ajoutons alors ce champ dans notre liste. Ce champ permet à l’utilisateur de filtrer les articles en fonction de leur référence, libellé, univers, catégorie, couleur ou fournisseur.



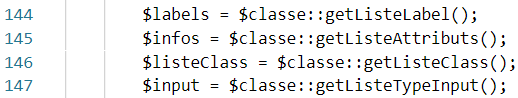
Une fois le filtre choisi, un clic sur le bouton rechercher lance alors une API interne qui recherchera alors tous les articles correspondant aux critères de recherche. Cette API utilise la fonction suivante pour la recherche par filtre :



Elle crée la requête en fonction des éléments données par l’utilisateur et retourne une liste des articles que l’on pourra alors afficher au retour de cette requête AJAX.

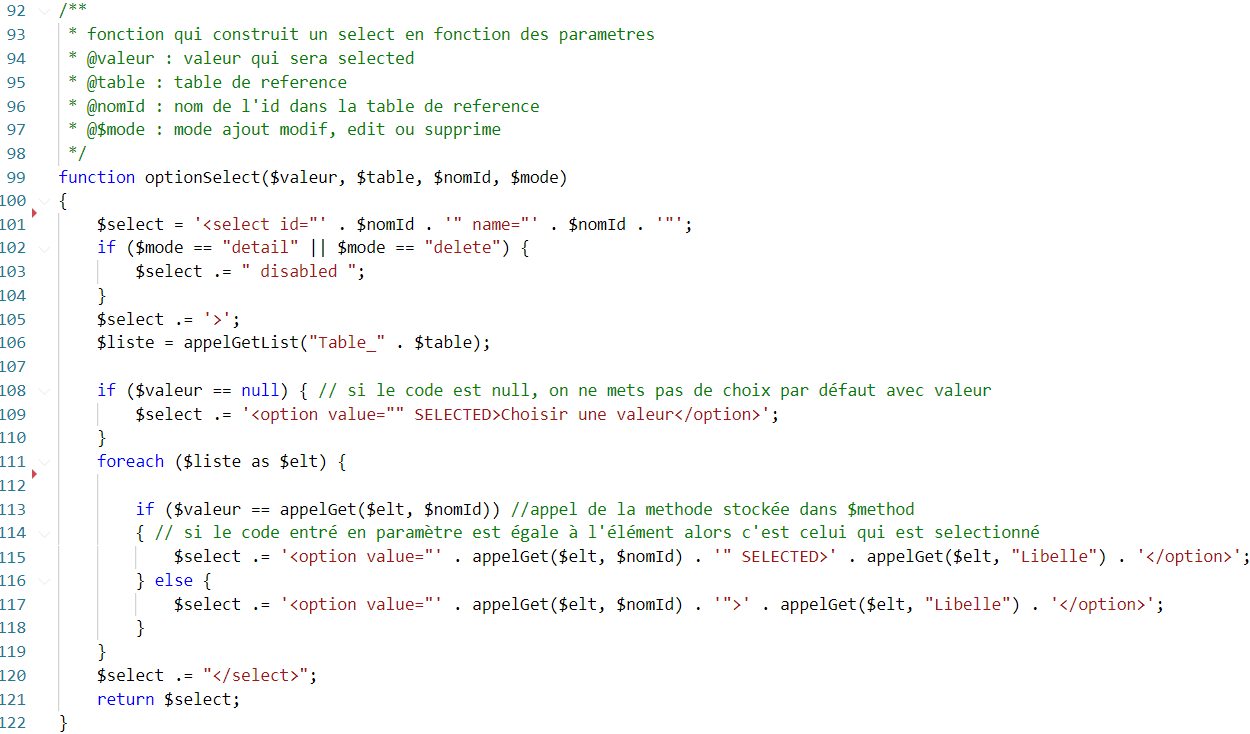
### VI-3-e/ Gestion des formulaires

De la même manière que les listes, les formulaires sont générés en fonction de la table données dans le GET et le formulaire permet de faire certaines choses ou pas en fonction du mode donné dans le GET (ajout, modification, suppression). Grâce aux tableaux inclus dans les classes on peut alors facilement créer les inputs correspondants aux valeurs de la table.



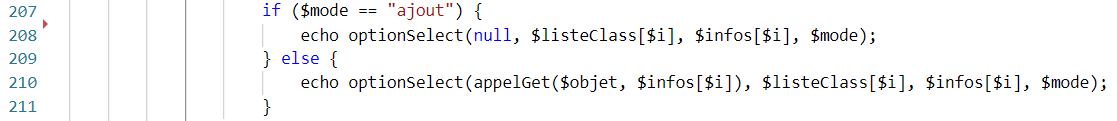
Pour les inputs de type select un traitement supplémentaire était nécessaire, c’est pourquoi nous avons créé une fonction optionSelect. Cette fonction crée un select personnalisé en fonction des paramètres donnés :

* La valeur qui sera sélectionnée par défaut.
* A quelle table ce select fait référence.
* Le nom de l’id de la table auquel ce select fait référence.
* Le mode du formulaire auquel ce select sera inclus (afin d’autoriser ou non certaines actions).



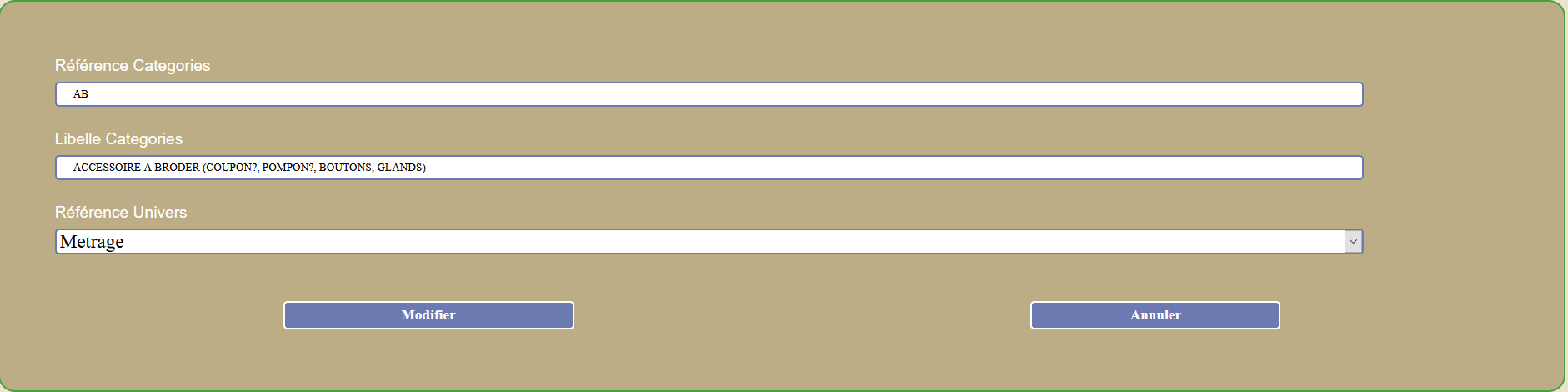
Dans cette fonction on commence par créer la balise select que l’on a besoin en fonction du nom de l’id ainsi que du mode (ne pas pouvoir changer les valeurs si on est dans le formulaire de détail ou de suppression). On récupère alors la liste des éléments de la table que l’on souhaite (ligne 106) pour créer les différentes options du select. Enfin la fonction retourne le select créé sous forme de string.

Avec cette fonction, il est alors possible de créer des select facilement dans le formulaire avec les lignes de codes suivantes :



Avec tout ceci on obtient alors une seule et unique page qui affiche n’importe quel formulaire utilisable par l’utilisateur.

Voici un exemple de formulaire de modification pour un éléments de la table catégorie :

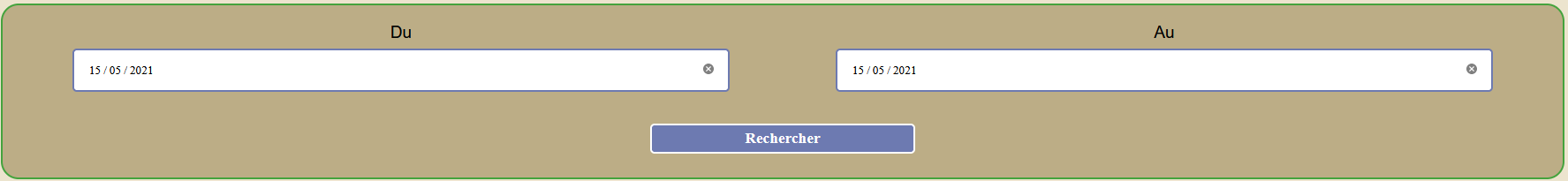


### 

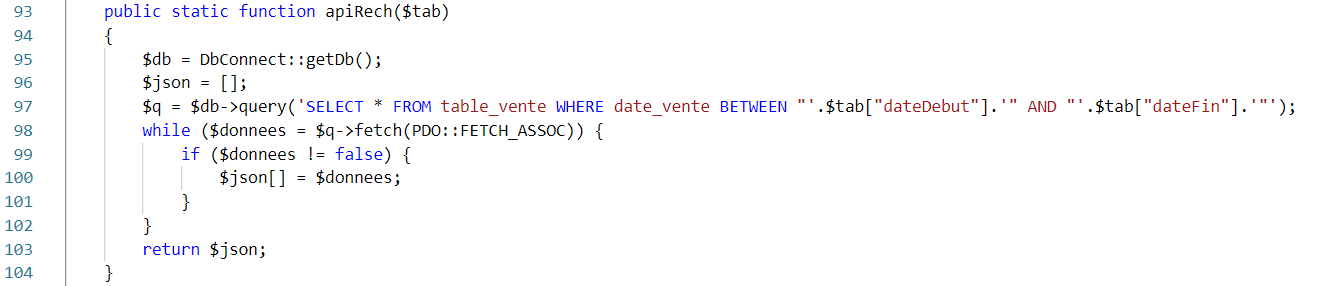
### VI-3-f/ Gestion des commandes

Cette partie est destinée aux caissiers ainsi qu’aux gérants afin de leur permettre la gestion des différentes commandes. L’interface de cette partie permet de filtrer les commandes par date ou tranche de date, ce qui permet de retrouver facilement une vente en cours et pouvoir la poursuivre ou retrouver une vente finie.

L’utilisateur peut entrer la date à partir de laquelle il souhaite consulter ses ventes et la date jusqu’à laquelle il les souhaite.

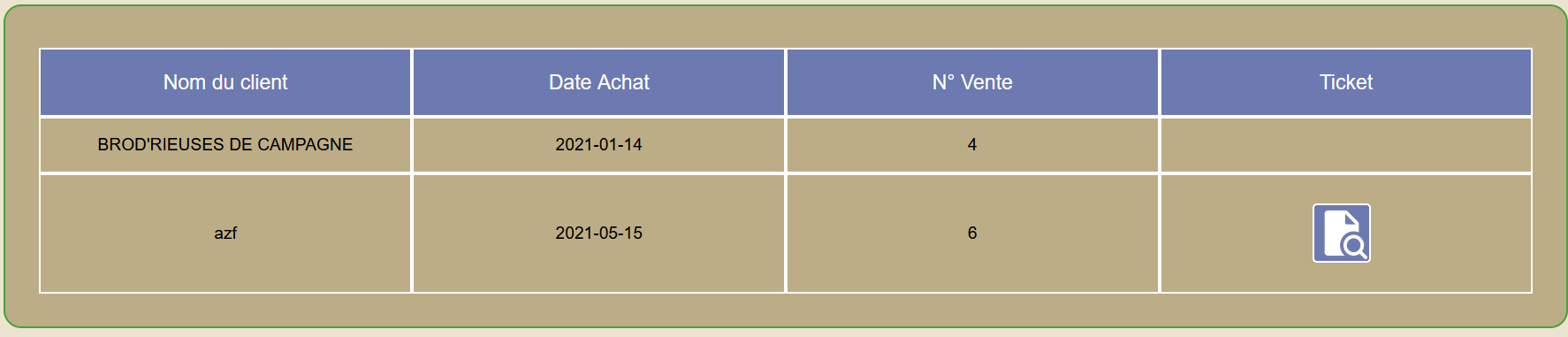


À l’appui du bouton rechercher l’application lance alors une API qui récupère toutes les ventes qui ont eu lieux pendant cette tranche de date grâce à la fonction apiRech ci-dessous:



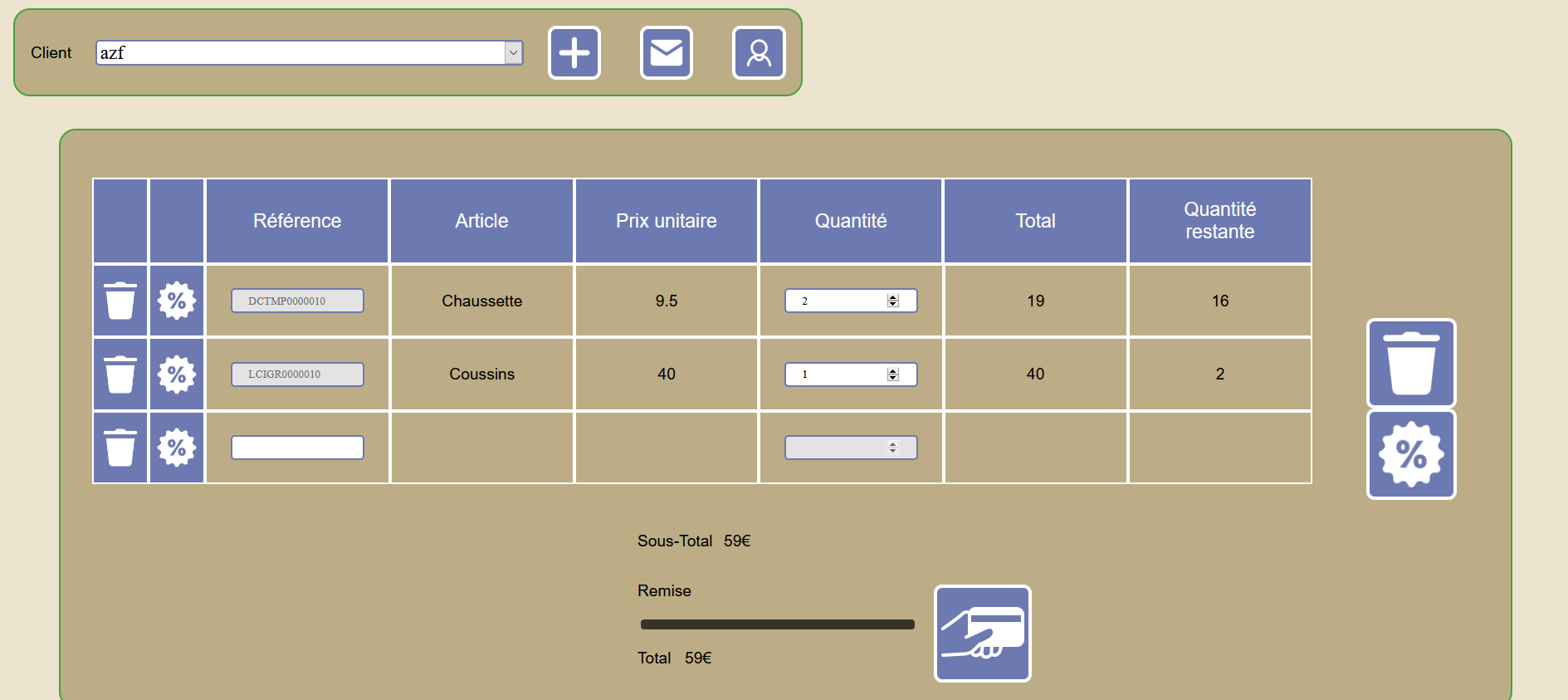
Avec cette liste on peut alors afficher toutes les ventes concernées par le filtre à l’utilisateur. Il pourra alors voir si les ventes ont été finies ou si elles sont encore en cours. En effet, si la vente est finie alors elle possède un ticket de caisse qui est alors consultable par l’utilisateur, et si la vente est encore en cours alors il peut cliquer sur la vente qu’il souhaite continuer afin d’être redirigé vers la page du passage à la caisse.

Par exemple dans le cas ci-dessous, la vente n°4 n’est pas finie et est cliquable pour pouvoir la continuer, alors que la vente n°6 est achevée et son ticket de caisse est consultable en cliquant sur l'icône en forme de document.

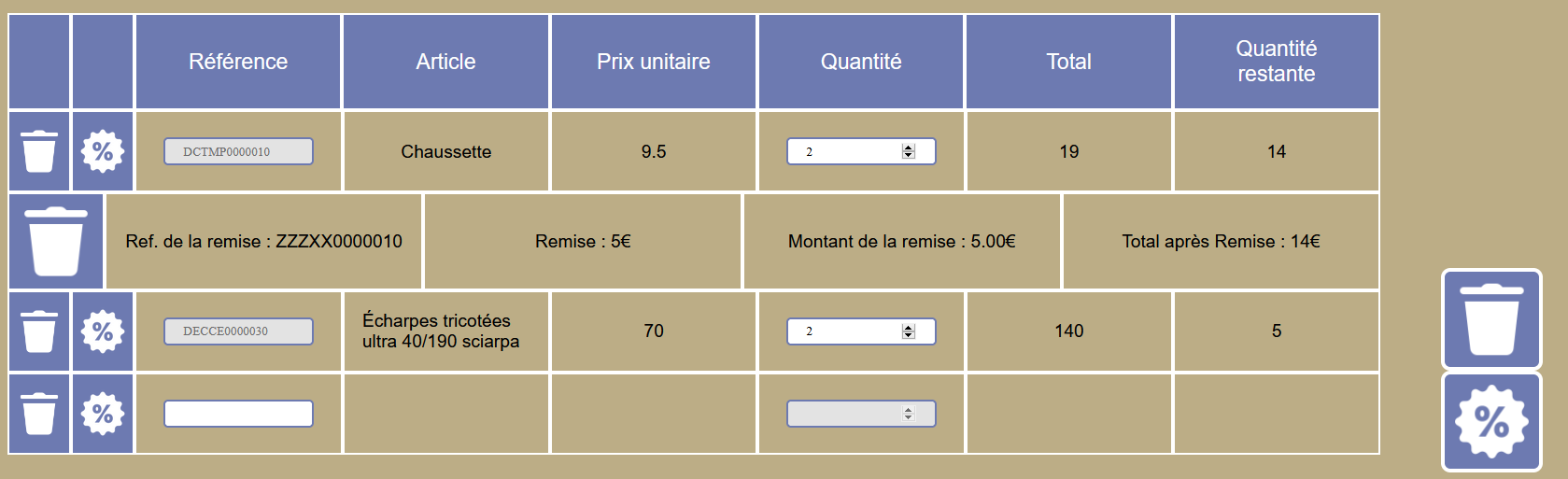


### VI-3-g/ Passage caisse

Cette partie de l’application permet à l’utilisateur de réaliser une vente. L’interface lui permet de choisir un client, et de scanner un article afin de l’ajouter à la vente et d’appliquer d’éventuelles réductions.



La partie en haut à gauche lui permet de choisir le client dans un menu déroulé, l'enveloppe affiche l’adresse mail de ce client et si des informations ne sont plus valides à propos de ce client il est possible de modifier ces informations en cliquant sur la dernière icône pour être redirigé vers le formulaire de modification du client. Si le client n’est pas encore enregistré il est possible d’en créer un avec le bouton plus qui redirigera alors l’utilisateur vers le formulaire de création de client. Si la commande avait déjà commencé (article déjà scanné), alors après la création ou la modification, l’utilisateur retourne sur cette page en ayant conservé toutes les données déjà remplies sur cette commande.



La partie au centre représente la commande totale du client. Le caissier peut scanner le code barre de l’article, ou l’écrire à la main. Si l’article est bien un article de la base de données et est encore en stock alors les informations sur l’article apparaissent dans le tableau grâce à une requête API interne. L’utilisateur peut donc voir le nom de l’article, son prix et la quantité restante en stock. Il ne lui reste plus qu’à indiquer la quantité souhaitée par le client dans l’input de la colonne quantité. Celui-ci ne peut pas excéder la quantité restante sinon l’application lui affiche un message d’erreur et invite le caissier à rentrer une valeur valide. Suite à la complétion de la ligne une autre ligne apparaît et permet au caissier de continuer sa vente.

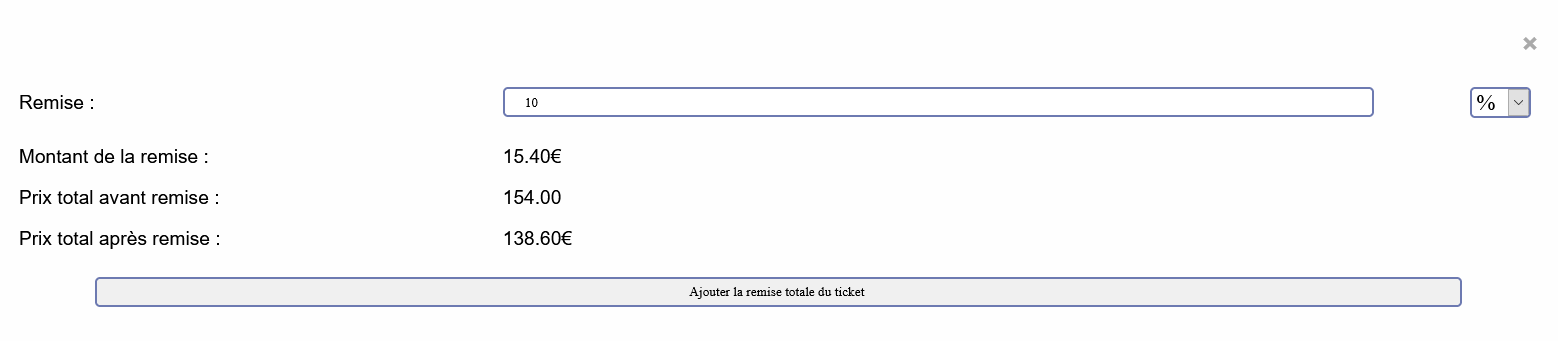
Sur chaque ligne se trouve un bouton remise qui permet au caissier d’attribuer une remise à l’article en question. En cliquant sur ce bouton un modal apparaît à l'écran :



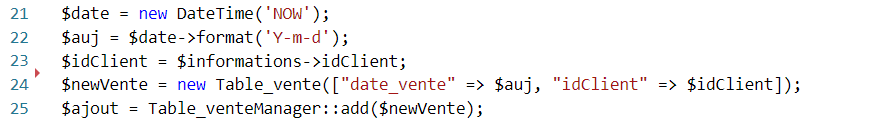
Cette interface lui permet de choisir quel type de remise il souhaite (en € ou en %), et de choisir son montant. En indiquant ceci il peut alors voir le prix de l’article après la remise, si cela lui convient il peut alors valider la remise. Une fois validé une ligne de remise apparaît sous l’article en question qui est supprimable en appuyant sur le bouton poubelle de la ligne de la remise.

De même, la ligne de l’article peut être supprimée en appuyant sur le bouton poubelle se trouvant sur sa ligne, si elle possédait une remise alors celle-ci sera également supprimée.

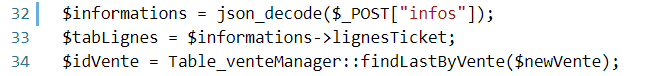
L’utilisateur peut également agir directement sur toute la commande en appuyant sur les 2 icônes à côté du tableau. En effet, le bouton poubelle permet de supprimer toutes les lignes de la commande et le bouton remise lui permet d’attribuer une remise totale sur la commande. En appuyant sur ce dernier, un modal apparaît de la même manière que pour les réductions sur les articles.

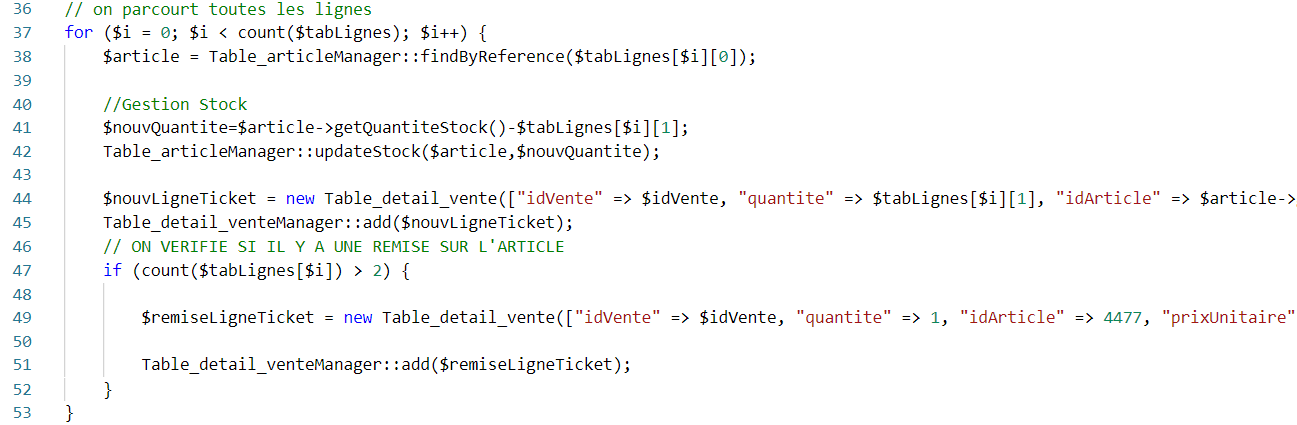


Enfin, quand la commande est finie, le caissier peut alors appuyer sur le bouton de règlement. À l'appui de ce bouton une API interne se déclenche afin de créer la vente dans la base de données ainsi que les détails de la vente, avec les informations récupérées de la page.

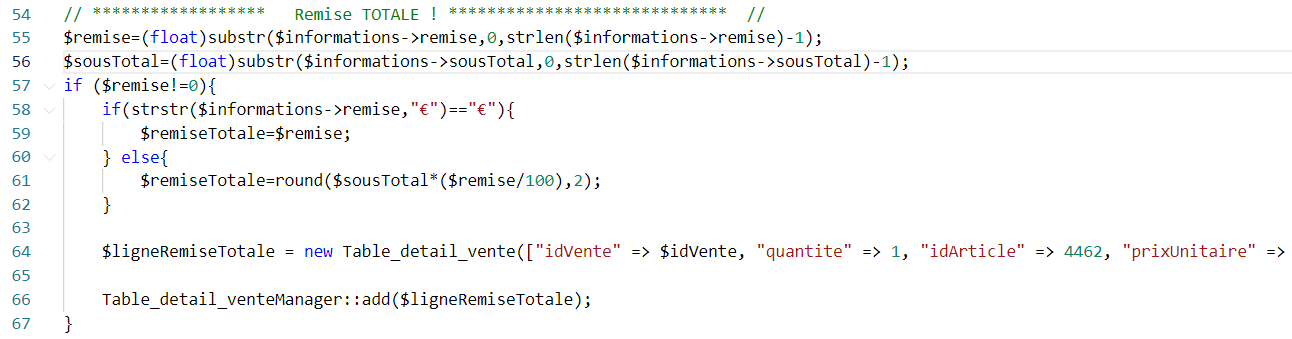


Ici on crée la nouvelle vente, qui est liée au client, que l’on va remplir avec les différentes lignes d’article de la page.





Pour créer tous les détails de la vente, on boucle sur les lignes des articles. On commence alors par décompter la quantité d'articles achetés par le client dans la base de données. Puis on crée le détail vente et on l’envoie dans la base de données. Et enfin, à la ligne 47 on regarde si une remise est appliquée à cet article, si c’est le cas on crée alors un détail vente qui aura alors un prix négatif (lié à la remise) et on l’envoie en base de données. Les remises sont gérées comme des articles avec un prix négatif.



Enfin, il reste à gérer la remise totale sur la commande. À la ligne 58, on vérifie le type de remise totale (en € ou en %) et on récupère ainsi le montant de cette remise. Et de la même manière que pour les remises par ligne, on la considère comme un article avec un prix négatif et on l’ajoute en base de données.

Suite à cette création de vente l’utilisateur est redirigé vers la page de règlement.

### VI-3-h/ Paiement

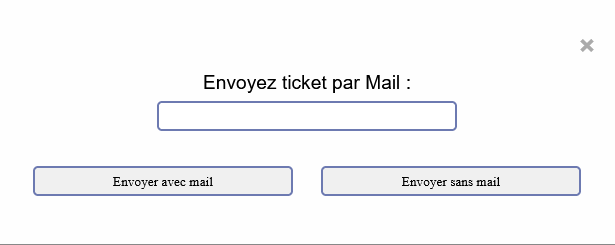
Suite à la vente, le caissier se retrouve sur l’interface suivante :



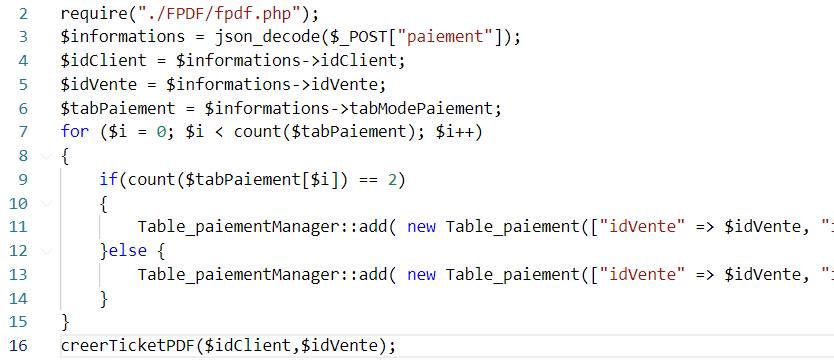
Il peut alors choisir les divers moyens de paiement que le client veut utiliser. Le menu déroulé à droite lui permet de choisir ce moyen de paiement et de l’accepter en appuyant sur le bouton plus. Ce bouton fait alors apparaître une nouvelle ligne de mode de paiement à gauche de l’interface, si ce mode de paiement est un paiement par chèque alors le caissier doit alors entrer le nom de la banque du client. Il peut alors choisir le montant que le client souhaite payer par mode de paiement. Le montant sera alors décompté du total à régler, et le reste dû sera alors mis à jour.

Il est également possible de supprimer la ligne du mode de paiement en appuyant sur l’icône poubelle de la ligne.

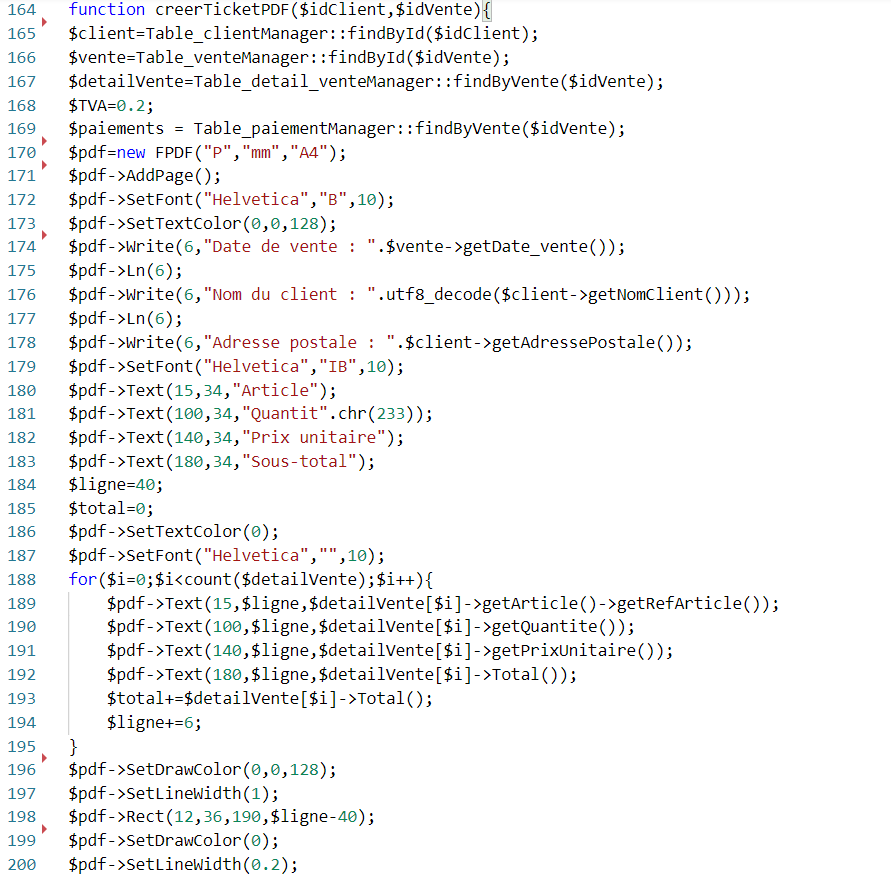
Une fois le reste dû à 0 il est alors possible d’appuyer sur le bouton payer et le modal suivant apparaît.



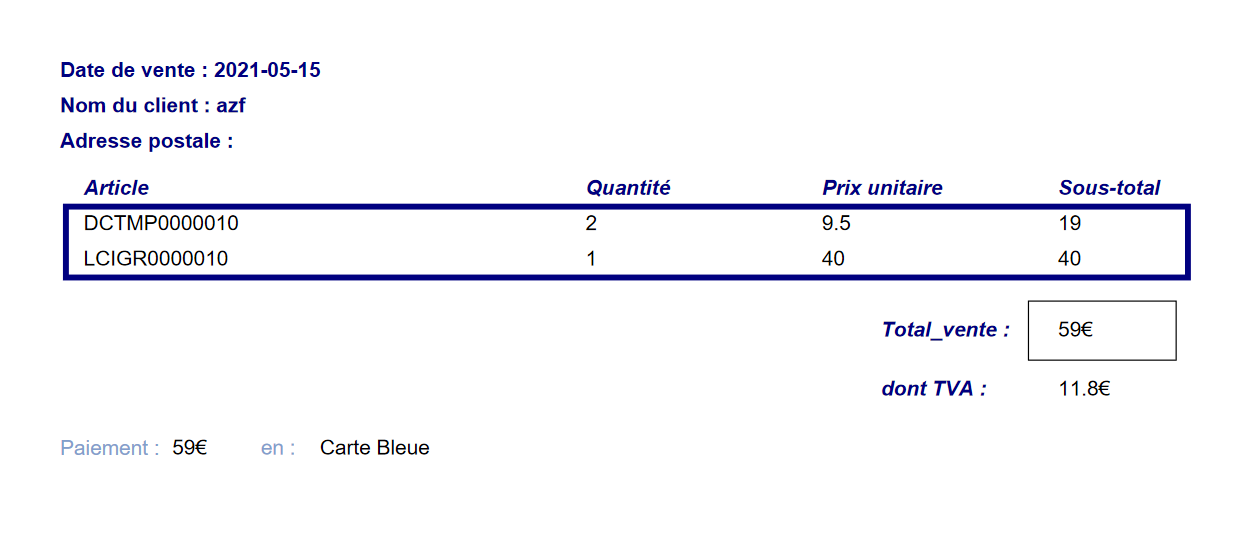
Le client peut alors demander à recevoir le ticket de caisse par mail ou non. Si le client est déjà enregistré et qu’il possède une adresse mail le champ est déjà rempli sinon le caissier peut alors le remplir. À l'appui de l’un de ces 2 boutons, un programme se lance alors pour créer le ticket de caisse et l’imprimer, et dans le cas où le client le souhaite de l’envoyer par mail.

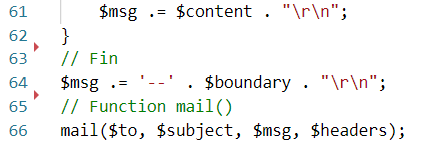


Pour pouvoir créer ce ticket de caisse nous avons utilisé la librairie FPDF qui permet de créer des fichiers PDF en PHP. Il permet de naviguer dans le fichier PDF et d’écrire ce que l’on veut où on le souhaite. Nous avons également créé la fonction creerTicketPDF qui prend en paramètre l’id du client et l’id de la vente, et qui crée alors le ticket de caisse en fichier PDF.



Suite à ce code nous obtenons alors un ticket de caisse sous forme de fichier PDF qui ressemble à l’image ci-dessous :





Si le client a demandé de recevoir le ticket de caisse par mail alors on lui envoie le mail avec le code ci-dessus. La ligne 29 permet de créer un boundary qui sert à découper les différentes parties de notre mail, le header, le message à afficher et les pièces jointes (compris dans le message). Enfin, à la ligne 66 on envoie le mail grâce à la fonction PHP mail qui prend en paramètre, le destinataire, le sujet, le message à envoyer et les en-têtes du mail.

Avec ce code le client reçoit alors le mail avec le ticket de caisse lié à son achat à la Boutique du Lin.

# 

# VII / Conclusion

Ce stage m’aura permis de travailler sur des projets concrets établies par des cahiers des charges complets et complexes. Devant travailler en complète autonomie durant l’un de ces projets, j’ai pu améliorer ma confiance en moi et améliorer mes compétences en développement. J’ai également pu découvrir de nouveaux langages ainsi que de nouveaux Framework qui élargiront davantage mes connaissances en développement.

Cette formation m’a permis de me remettre dans le domaine du développement informatique et de m’améliorer dans ce domaine en me spécialisant dans le développement d'applications web et web mobile.

# VIII/ Bibliographie

NodeJS

VueJS

EmailJS

Vuetify

PHP Manual

W3School

Wordpress developer

Mozilla developer

Stack overflow