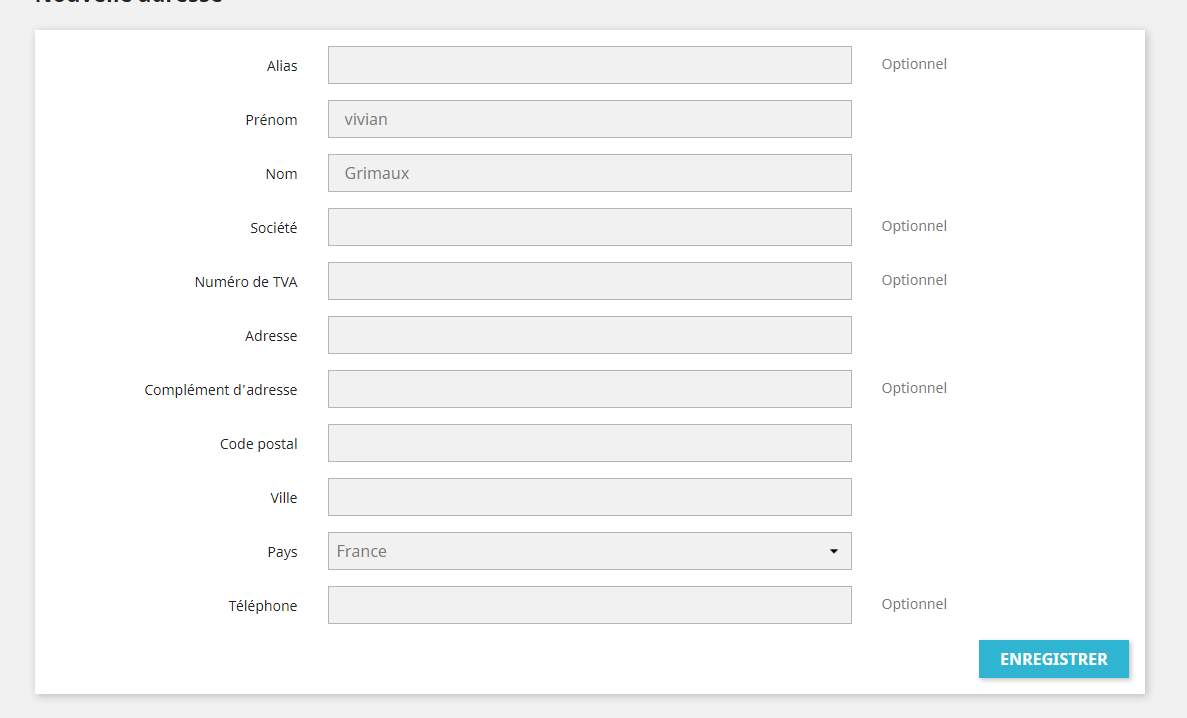
Création d’un champ obligatoire Prestashop PHASE 1

L’objectif de cette tâche est de créer un nouveau champ à renseigner dans le formulaire de création d’adresse ou de création de compte.

Ceci nous permettra d’obliger le client à renseigner des informations essentielles à nos procédés de fabrication.

Lors de notre production, nous avons besoins d’avoir à disposition le numéro de carte bancaire du client pour mettre en place les passerelles sur les terminaux destiné à la vente. Cette information est essentielle puisque la vente des TPE via le site de e-commerce va de pair avec la vente d’une passerelle (RTC, 3G, etc…).

Ci-dessus nous pouvons observer un formulaire d’adresses clients standard de PrestaShop.

Le but est d’intégrer un champs OBLIGATOIRE supplémentaire au formulaire, nous permettent ainsi de stocker ce numéro, soit par adresse soit par clients.

Ainsi, le numéro sera enregistré et le client ne saura pas obliger de le renseigner à chaque achat d’un TPE.

On retrouve le code du module de la page 2 à 4 :

Explication du code

Dans cette version du code, j’ai créé un champs numéro de carte bleue

Ce qui est en vert (précédé par // ou /\*) sont des commentaires, non actif dans le code

<?php

class FieldSupCB extends Module {

    public function \_\_construct() {

1

        $this->name = 'FieldSupCB';

        $this->tab = 'others';

        $this->author = 'VGrimaux';

        $this->version = '1.0.0';

        $this->need\_instance = 0;

        $this->bootstrap = true;

        parent::\_\_construct();

        $this->displayName = $this->l('FieldSupCB');

        $this->description = $this->l('add new fields to customer');

        $this->ps\_versions\_compliancy = array('min' => '1.7.1', 'max' => \_PS\_VERSION\_);

    }

    /\*\*

     \* @return boolean

     \*/

    public function install() {

        if (!parent::install() || !$this->\_installSql()

                //Hooks Admin

                || !$this->registerHook('actionAdminCustomersControllerSaveAfter')

                || !$this->registerHook('actionAdminCustomersFormModifier')

2

                //Hooks Front

                || !$this->registerHook('additionalCustomerFormFields')

                //Hooks objects

                || !$this->registerHook('actionObjectCustomerAddAfter')

                || !$this->registerHook('actionObjectCustomerUpdateAfter')

                //Hook validation des champs

                || !$this->registerHook('validateCustomerFormFields')

        ) {

            return false;

        }

        return true;

    }

3

public function uninstall() {

return parent::uninstall() && $this->\_unInstallSql();

}

/\*\*

\* Modifications sql du module

4

\* @return boolean

\*/

protected function \_installSql() {

$sqlInstall = "ALTER TABLE " . \_DB\_PREFIX\_ . "customer "

. "ADD numero\_carte\_bleue INT(7) NULL";

return Db::getInstance()->execute($sqlInstall);

}

/\*\*

\* Suppression des modification sql du module

5

\* @return boolean

\*/

protected function \_unInstallSql() {

$sqlUnInstall = "ALTER TABLE " . \_DB\_PREFIX\_ . "customer "

. "DROP numero\_carte\_bleue";

return Db::getInstance()->execute($sqlUnInstall);

}

/\*\*

\* Modification du formulaire d'édition d'un client en BO

\* @param type $params

\*/

6

public function hookActionAdminCustomersFormModifier($params) {

$params['fields'][0]['form']['input'][] = [

'type' => 'text',

'label' => $this->l('Numéro de carte bleue'),

'name' => 'numero\_carte\_bleue',

'class' => 'input fixed-width-xxl',

'hint' => $this->l('Numéro de carte bleue')

];

//Définition de la valeur du champ supplémentaire

$params['fields\_value']['numero\_carte\_bleue'] = $params['object']->numero\_carte\_bleue;

}

/\*\*

\* Ajout d'un champ client supplémentaire en FO

\* @param type $params

\*/

public function hookAdditionalCustomerFormFields($params) {

7

return [

(new FormField)

->setName('numero\_carte\_bleue')

->setType('text')

->setLabel($this->l('Numéro de carte bleue')),

];

}

/\*\*

\* Validation des champs du formulaire client

\* @param type $params array : Tableau des champs du formulaire client lié au module

\* Instance de Field

\* cf. https://github.com/PrestaShop/PrestaShop/pull/6374

\*/

public function hookValidateCustomerFormFields($params)

{

foreach ( $params['fields'] as $field){

/\*\* @var FormField $field \*/

8

//Validation custom du champ numero\_carte\_bleue si celui-ci n'est pas vide

if ( $field->getName() == 'numero\_carte\_bleue' && $field->getValue() != "") {

//Mise en place de notre vérification (ex: 7 caractères pour le champ)

if ( strlen($field->getValue()) != 7 ){

$field->setErrors(array($this->l('This code must have 7 characters.')));

}

}

}

//Renvoi du tableau des paramètres au validateur

return $params['fields'];

}

}

* L’encadré n°1 rouge en page n°2 représente les informations du module, son nom, la version de PrestaShop supporté, l’éditeur du code, etc…
* L’encadré n°2 en page n°2 est une fonction qui installe le module. Dans le même temps, les Hook utilisé par le module sont initialisé. Les Hook sont une façon de raccorder le code aux actions ou aux emplacements définis par PrestaShop.
* L’encadré n°3 en page n°3 est une fonction permettant de désinstaller le module.
* L’encadré n°4 en page n°3 est une fonction qui créer la colonne « numero\_carte\_bleue » dans la table « Customer » sur la base de données, nécessaire pour enregistrer les informations entrées par le client dans le nouveau champ.
* L’encadré n°5 en page n°3 est une fonction qui supprime les colonnes enregistrées dans la base de données lors de la fonction de l’encadré n°4.
* L’encadré n°6 en page n°3 est une fonction Hook qui permet la modification du formulaire d’inscription des clients dans le Back Office, ainsi l’administrateur pourra créer/supprimer ou modifier un client directement dans sa console d’administration. Plus bas dans la fonction on retrouve les paramètres qui définissent les valeurs du champ. Actuellement le champ n’est pas obligatoire, on abordera ce point un peu plus tard dans la description de la surcharge de classe.
* L’encadré n°7 en page n°4 est une fonction Hook qui ajoute le champs « Numéro de carte bleue » dans le front office pour les utilisateurs.
* L’encadré n°8 en page n°4 est une fonction qui permet de vérifier la validité de la saisie du client en termes de nombre de caractère. Ainsi, si le client ne rentre pas un chiffre a 7 caractères comme ceux d’un code de carte bleue, l’inscription sera refusée.

Ensuite, après la création du module il est nécessaire de modifier le classe Customer (Client) en lui créant une surcharge :

<?php

class Customer extends CustomerCore {

//Nouveaux paramètres de classe

public $numero\_carte\_bleue;

9

public function \_\_construct($id = null) {

//Définition du nouveau champ numero\_carte\_bleue

self::$definition['fields']['numero\_carte\_bleue'] = [ 'type' => self::TYPE\_INT,

'required' => false, 'size' => 7

];

parent::\_\_construct($id);

}

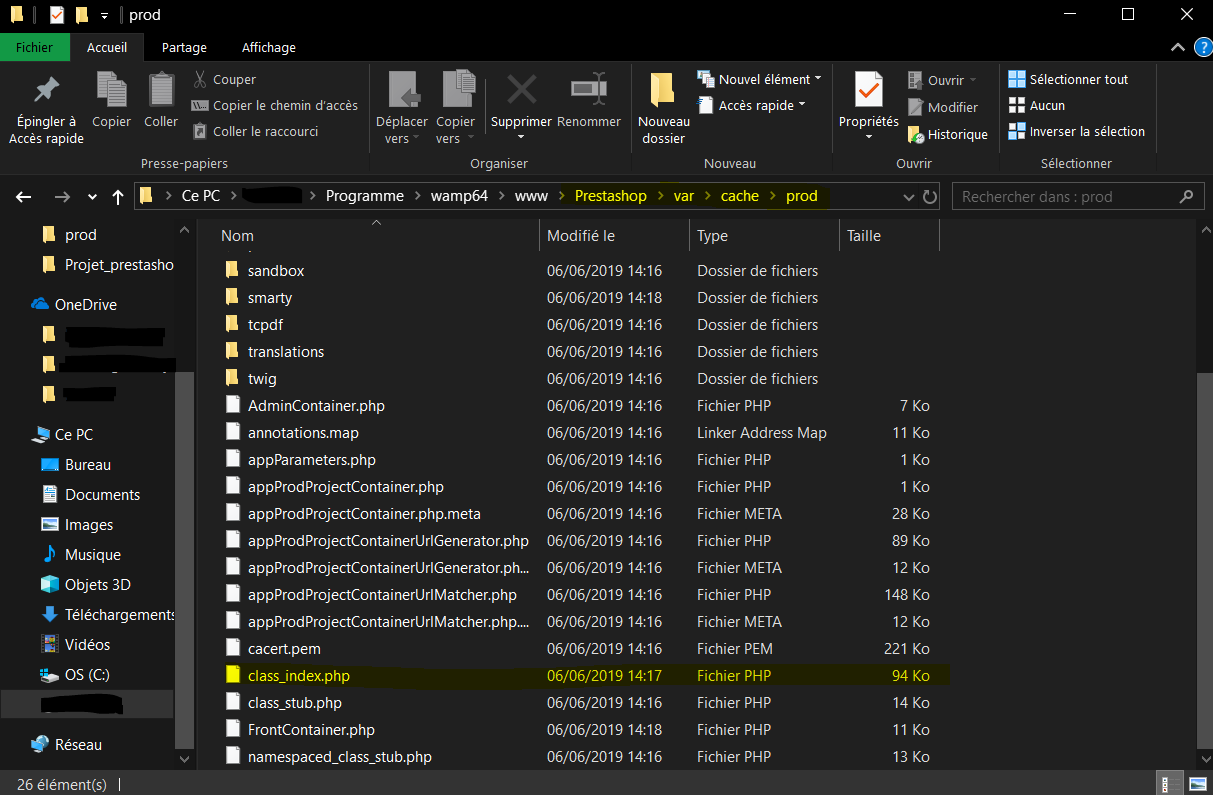
}

* Dans l’encadré n°9 en page n°5 on retrouve le nouveau paramètre « $numero\_carte\_bleue » de la classe et la définition du nouveau champ.   
  Lors de la définition du nouveau champ, on définit le type de l’information (ici in chiffre entier), son nom, sa taille et s’il est obligatoire ou non.

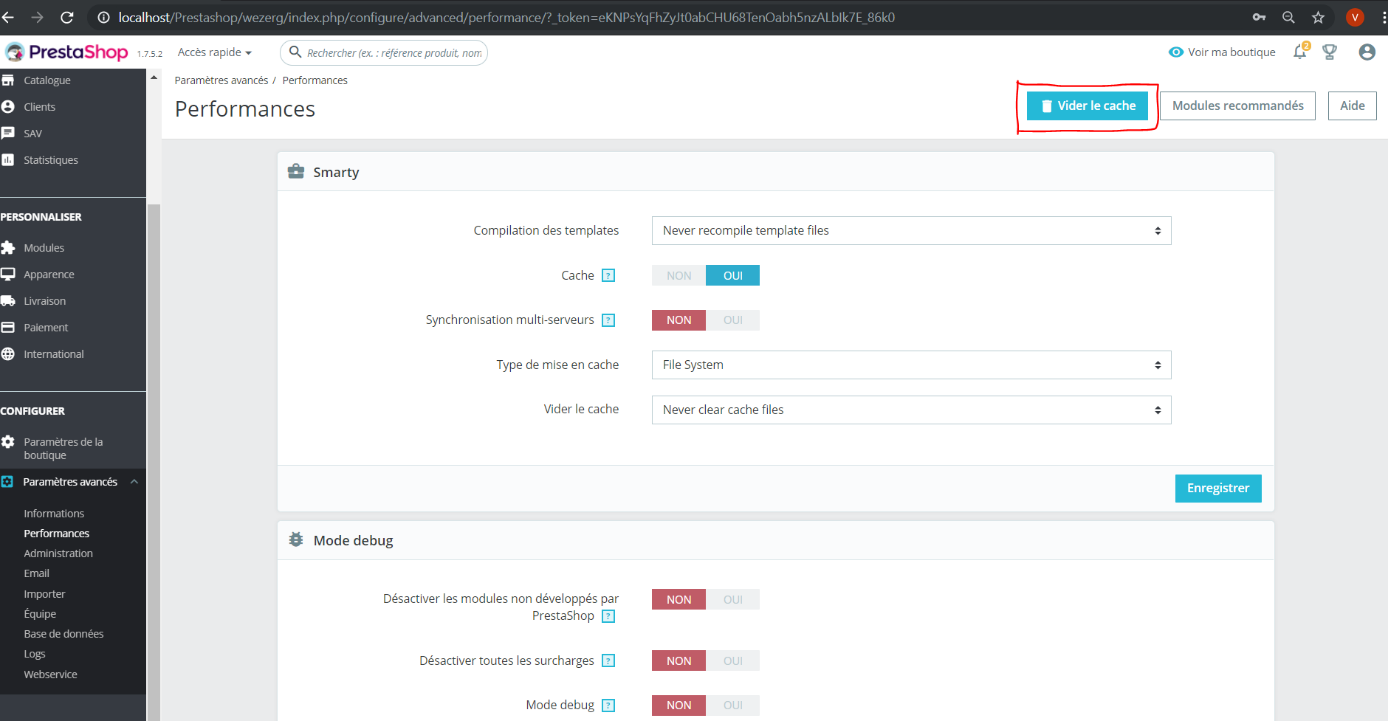
Ci-dessus on notera qu’il n’est pas obligatoire (required => false). Un des tests sera porté sur cette notion, savoir si le champ est obligatoire ou non.

Mise en place

Pour faire la mise en place du module il faut être minutieux.  
Tout d’abord, il faudra penser à vider le cache de votre site web pour que la surcharge de votre classe Customer se fasse comme il faut.   
Ce cache se trouve généralement dans le dossier app/cache/prod ou app/cache/dev sous le nom class\_index.php. Ci-dessous un Screenshot du dossier :



On peut aussi vider le cache par le back office du site en allant dans **Paramètre avancée > Performances > Vider le cache.** Ci-dessous le panneau de configuration en question :

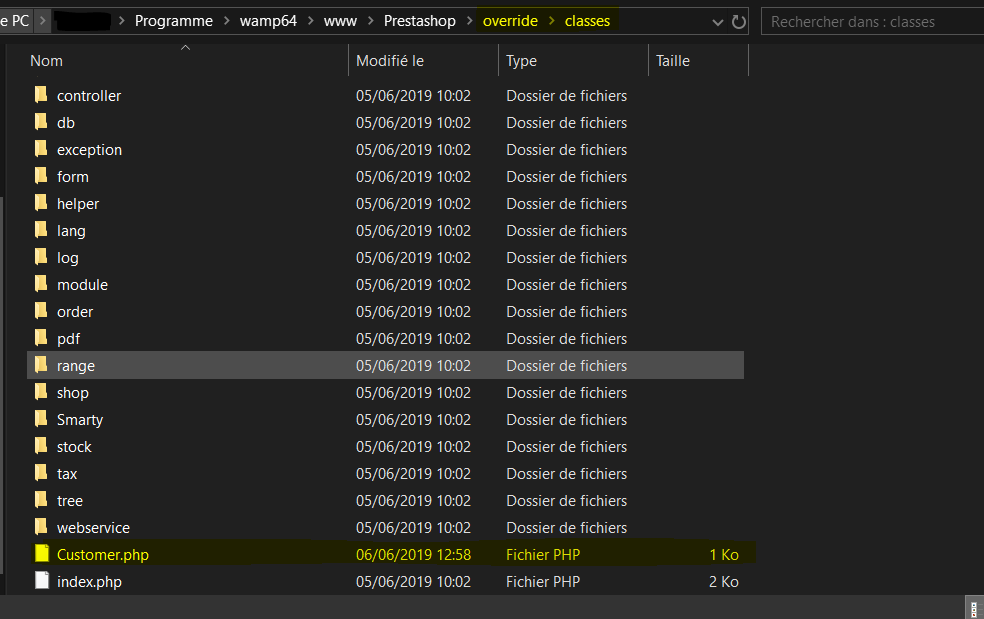


Ensuite, lorsque le cache est bien vidé, on placera notre fichier de module « FieldSupCB.php » et le fichier de la surcharge de classe « Customer.php » dans une archive zip dont le nom est celui du module.



Lorsque l’archive est créée il faut se rendre dans PrestaShop pour installer le module. Pour cela, allez dans l’onglet **Module > Modules Manager > Installer un module** et sélectionner votre archive ou faite la glisser avec un glisser-déposer de la souris. Le site PrestaShop devrait vous mettre une fenêtre vous disant que l’import a bien été effectué. Attention des erreurs peuvent survenir à cette étape, comme l’erreur 500 qui met en cause le serveur et l’hébergement.

En temps normal lors de l’installation, le module exporte de lui-même le fichier de la surcharge « Customer.php » à l’adresse **/override/classes/Customer.php.** Cependant si ce n’est pas le cas, n’hésitez pas à couper-coller le fichier dans le dossier spécifié.



A ce stade, la modification est active, relancer votre serveur pour rafraichir le site.

Des erreurs peuvent subvenir si :

* Le cache n’a pas été vidé
* Le nom du module n’est pas le même que son dossier
* Les noms des champs ne sont pas identiques
* Le nom de la surcharge n’est pas le même
* La surcharge n’est pas bien placée
* La surcharge n’est pas prise en compte
* La BDD n’a pas été bien renseigner dans le code
* Les tables n’ont pas été créer dans la BDD
* Les accès de la BDD sont erronés

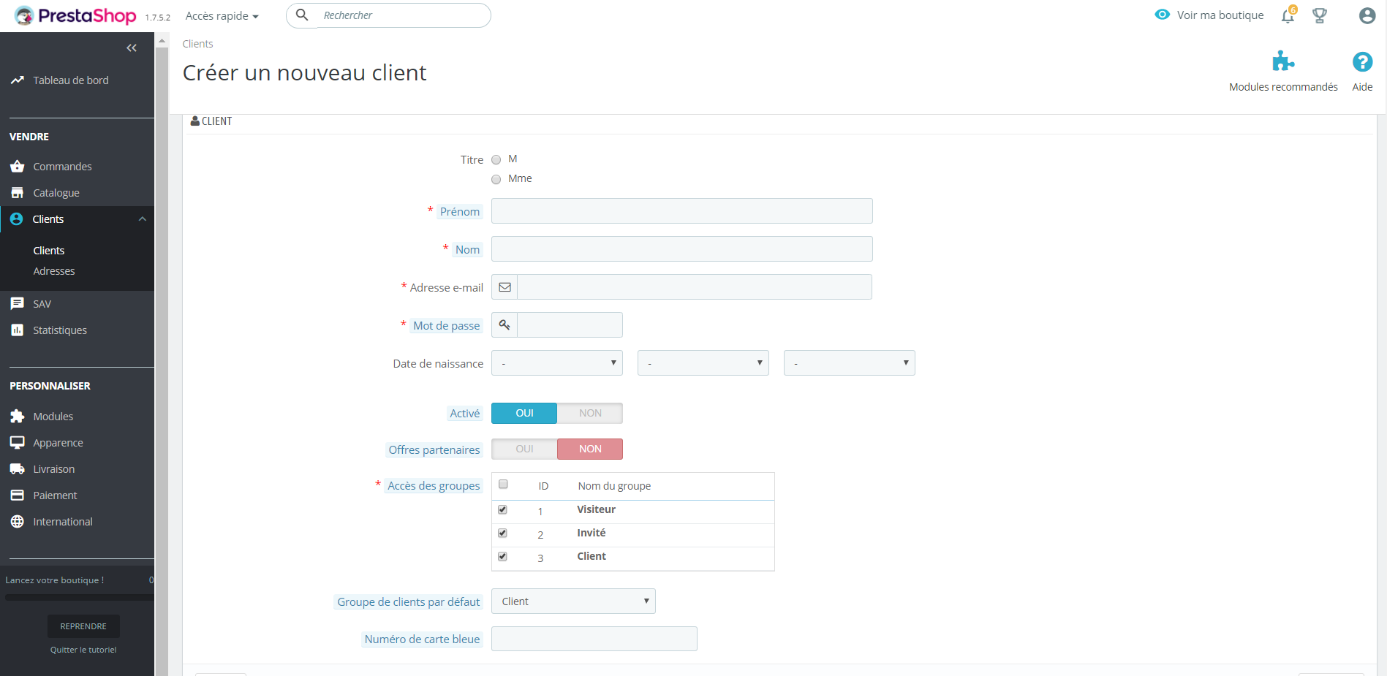
Phase de tests :

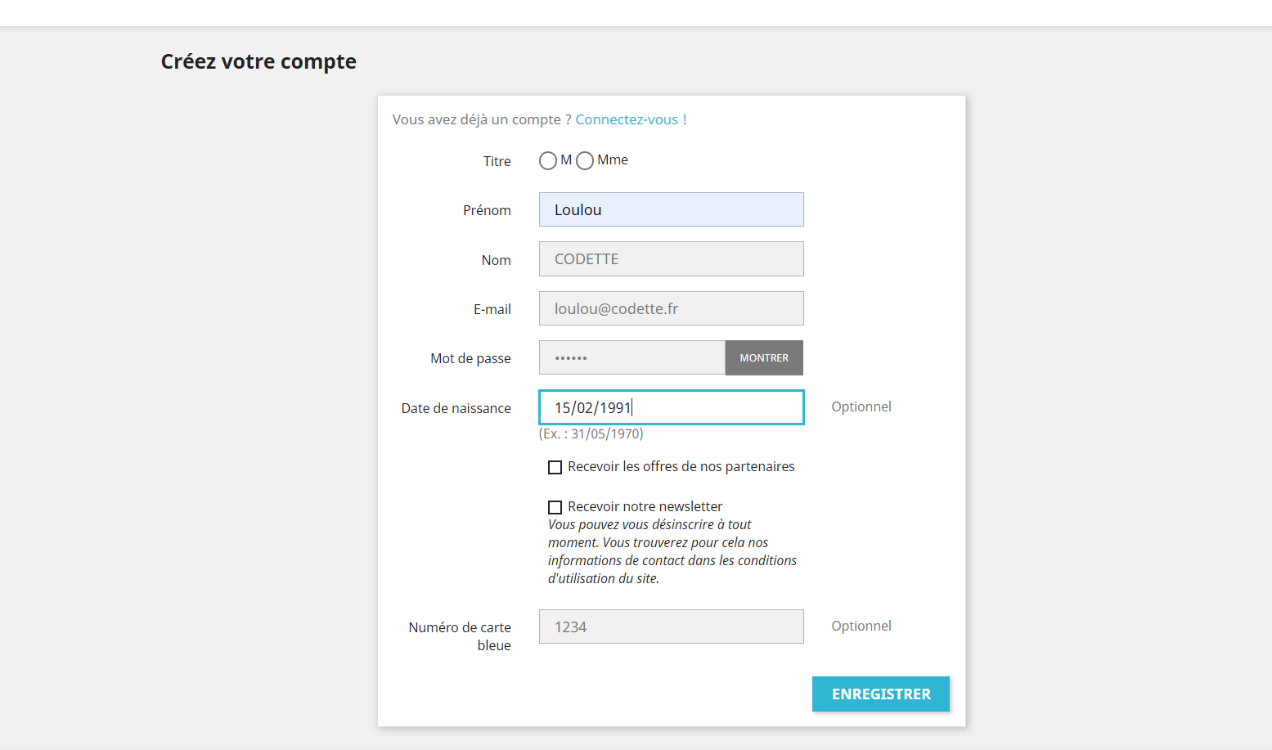
La phase de test se fait sur un site PrestaShop neuf, ce dernier est hébergé sur WampServer en local, la base de données est conservée sur MySQL Workbench 8.0. Un client générique nommé John DOE est créé instantanément lors de la création du site. Le schéma de la base de données a été créer à la main, cependant les tables ont été créer et rempli par PrestaShop de façon automatique, un préfixe « ps\_ » a été ajouté à toutes les tables de la BDD.

Ci-dessous plusieurs phases de test pour s’assurer du bon fonctionnement du module de code.

* **1er Test : Présence des champs OK**

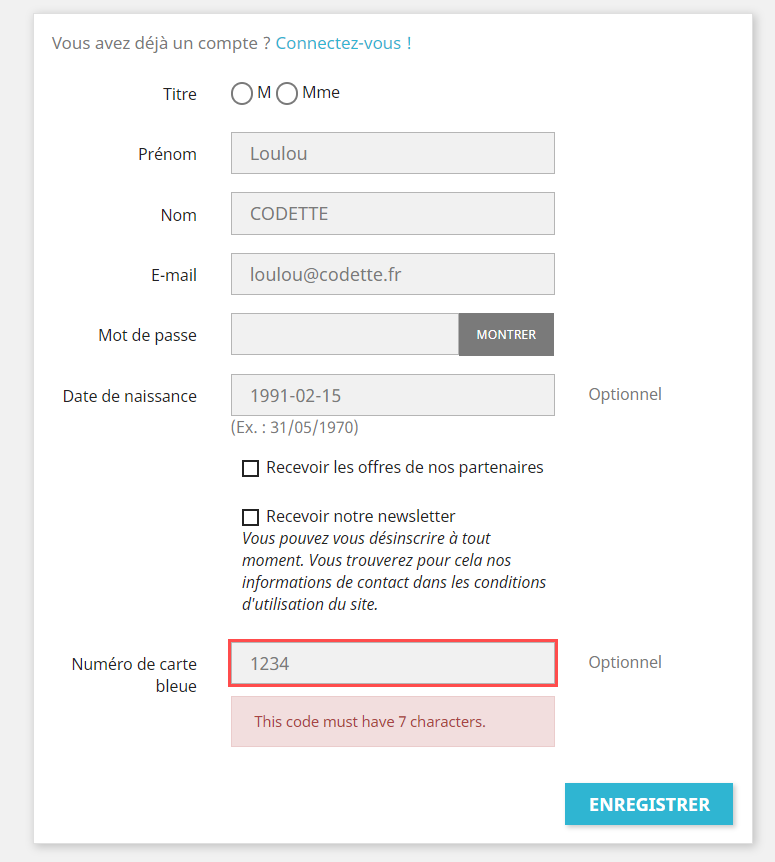
Ci-dessous un Screenshot de la création de compte client dans l’administration PrestaShop. On remarque tout en bas le champs « Numéro de carte bleue ».

* 

Ci-dessous un Screenshot de la création de compte client dans le front office. Tous comme le back office, le champs « Numéro de carte bleue » apparait en bas du formulaire.

* **2ème Test : Validité des caractères saisis en front-office OK**

Ci-dessous un Screenshot de la vérification effective du nombre de caractère, on peut voir dans cet exemple que le client a voulu rentrer un numéro a 4 caractères, le système a refusé sa saisie en lui montrant le message « This code must have 7 characters » en rouge.



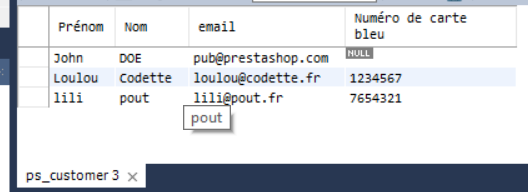
L’objectif étant de rajouter des codes banques composé uniquement de chiffre, un contrôle supplémentaire peut être rajouter sur la nature des caractères saisie.

On peut donc imaginer brider l’utilisateur aux simples caractères numériques allant de 0 à 9 pour éviter d’avoir un code banque avec le bon nombre de caractères, (7) mais de mauvais caractères saisis, exemple : 096R234.

* **3ème Test : Lien avec la base de données OK**

Après la création des deux comptes, à la fois dans le front office et dans le back office, nous avons procédé à une simple requête dans la base de données comme suit :

**select firstname as Prénom, lastname as Nom, email, numero\_carte\_bleue as "Numéro de carte bleu"  
from ps\_customer;**



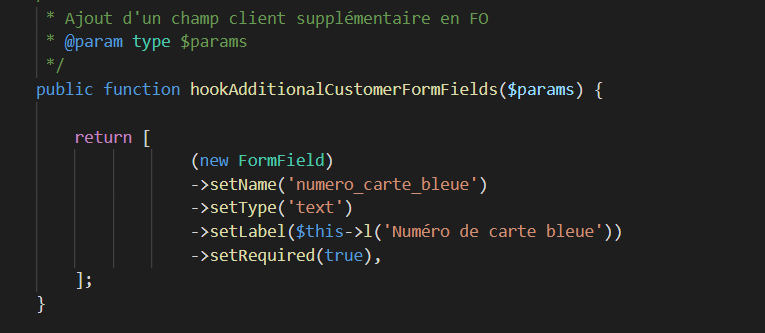
On retrouve donc nos 3 comptes :

* 1er compte : Compte générique créer par Prestashop, il ne possède pas de numéro de carte bleue
* 2ème compte : Compte créer dans le back office, il possède bien son numéro de carte bleue
* 3ème compte : Compte créer dans le front office, il possède aussi son numéro de carte bleue
* **4ème Test : Obligation de saisir son code**

Comme écrit dans l’aide de l’encadré n°6 en page 5, le site n’oblige pas l’utilisateur à rentrer son code.

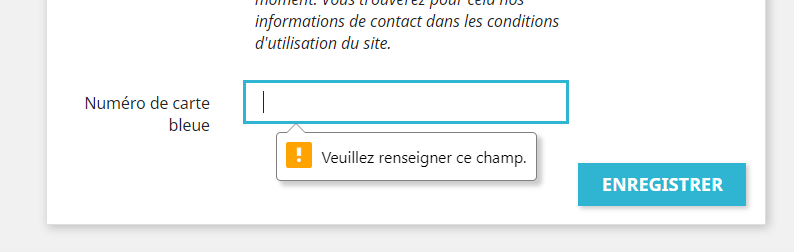


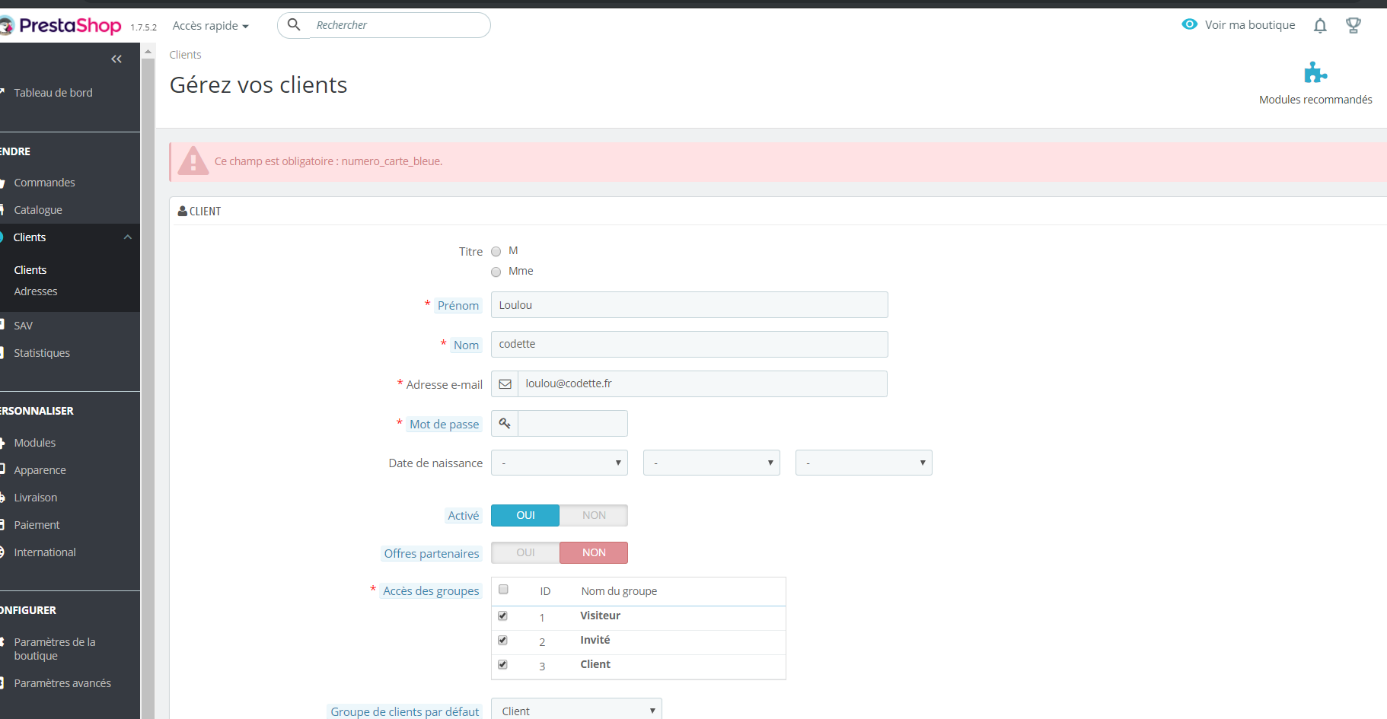
Après plusieurs test, une solution a été trouvée, nous avons modifié la fonction « HookAdditionalCustomerFields »



Nous avons rajouté la ligne « SetRequired(true) » directement dans le module pour obliger le renseignement du code.

Ci-dessous un Screenshot du front office, on peut voir que le site oblige bien le client à renseigner l’information.



Ci-dessous un Screenshot du back office ou on voit que le champ est obligatoire.

Cependant, il manque l’étoile rouge précisant qu’il est obligatoire.

Problèmes prévisionnels :

**Le 1er problème** est l’information souhaité :

* Nous faut-il seulement le code affiliation de carte bleue du commerçant ?
* Faut-il aussi récupérer son code banque ?
* Ou récupérer son code American Express (Amex) ?
* Ou bien récupérer son code de Vente à distance (VAD) ?
* Ou encore récupérer son numéro de sans contact ?

**Le 2ème problème** découle du premier : va-t-on obliger un commerçant à rentrer 5 informations différente lorsqu’il créer son compte ? C’est peut-être fastidieux même si cela n’est fait qu’une fois.

Cela va-t-il rebuter le client de s’inscrire et ainsi rater de vendre par la surcharge d’information ?

**Le 3ème problème** découle aussi du premier. Est-ce que tous les champs cités dans la première problématique seront obligatoire ?

Si tel est le cas, comment un commerçant ne possédant pas de sans contact ni de VAD pourra-t-il s’inscrire ?

**Le 4ème problème** survient à l’achat. Comment va-t-on gérer les commerçants qui possèdent tel ou tel option ? Si un commerçant ne possède pas de code Amex, il ne pourra pas acheter un TPE initialisé pour AMEX.

Dans ce cas le bouton d’achat sera-t-il grisé ?   
Le page d’achat du TPE ne s’affichera pas ?

PHASE 2

Mise en situation :

L’objectif était de mettre à disposition du client des champs supplémentaire à la création de compte pour l’initialisation des TPE.

Des champs ont été créer pour les informations suivantes :

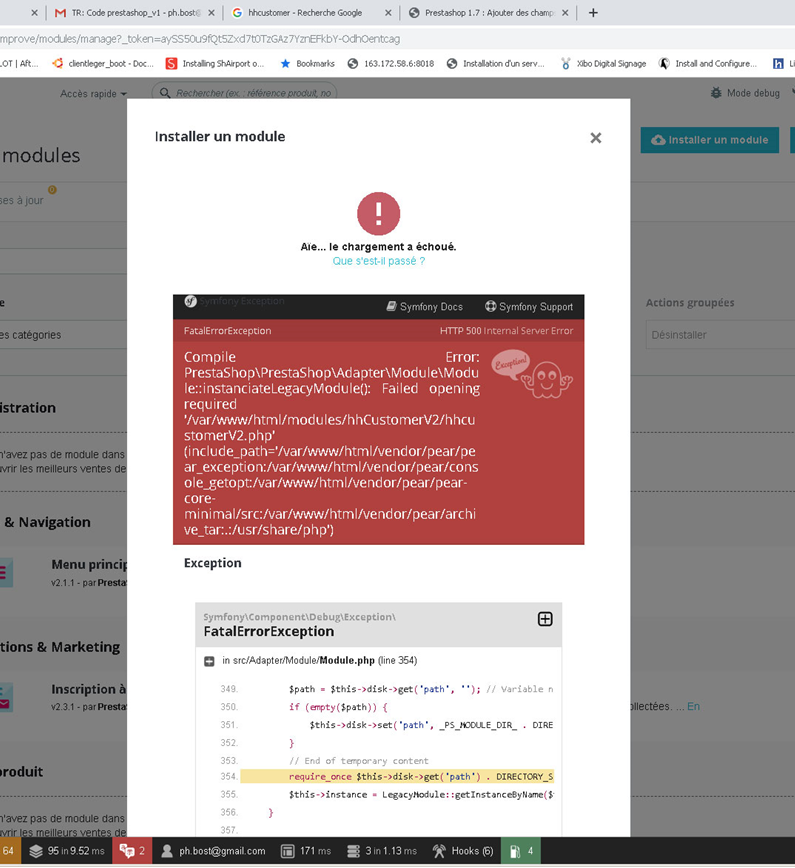
* Code Banque OBLIGATOIRE
* Numéro d’identification OBLIGATOIRE
* Numéro de commerçant OPTIONNEL
* Numéro American Express (AMEX) OPTIONNEL
* Numéro de vente à distance OPTIONNEL

La décision de fixer ces champs ci-dessus résout le premier problème prévisionnel que nous avions soulevez sur la questions de l’information demandé.

Le second et troisième problème sont eux aussi résolus puisque l’on demande effectivement 5 informations au clients mais seulement 2 d’entre elles sont obligatoire à la création du compte.

Le 4ème problème sera résolu par un encadré destiné aux clients pour les informer de la nécessité de communiquer leur numéro.

M.A.J du 26/06/2019 :

Un problème est survenu à l’installation du module :

La résolution du problème s’est effectuée par un correctif sur les nom des fichiers du module.

Le code est identique à celui décrit en phase 1 a la seule différence que tous ce qui affairait au champs « numero\_carte\_bleue » est désormais remplacé par les champs ci-dessus. Ainsi pour exemple, la fonction qui permet l’ajout du champs en front-office se transforme comme suit :

public function hookAdditionalCustomerFormFields($params) {

return [

(new FormField)

->setName('numero\_carte\_bleue')

->setType('text')

->setLabel($this->l('Numéro de carte bleue'))

->setRequired(true),

];

}

public function hookAdditionalCustomerFormFields($params) {

return [

(new FormField)

->setName('code\_banque')

->setType('text')

->setLabel($this->l('Code banque'))

->setRequired(true),

(new FormField)

->setName('numero\_identification')

->setType('text')

->setLabel($this->l('Numéro d\'identification'))

->setRequired(true),

(new FormField)

->setName('numero\_commercant')

->setType('text')

->setLabel($this->l('Numéro de commercant')),

(new FormField)

->setName('numero\_american\_express')

->setType('text')

->setLabel($this->l('Numéro American Express (AMEX)')),

(new FormField)

->setName('numero\_vente\_a\_distance')

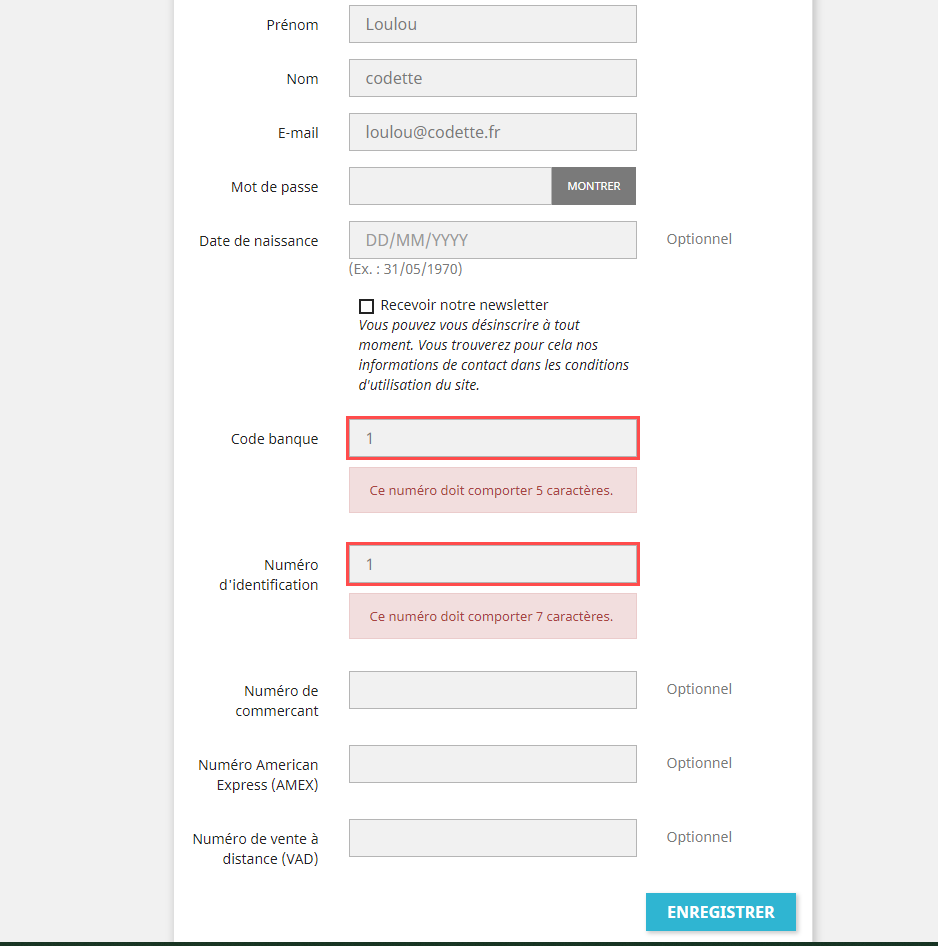
->setType('text')

->setLabel($this->l('Numéro de vente à distance (VAD)')),

];

}

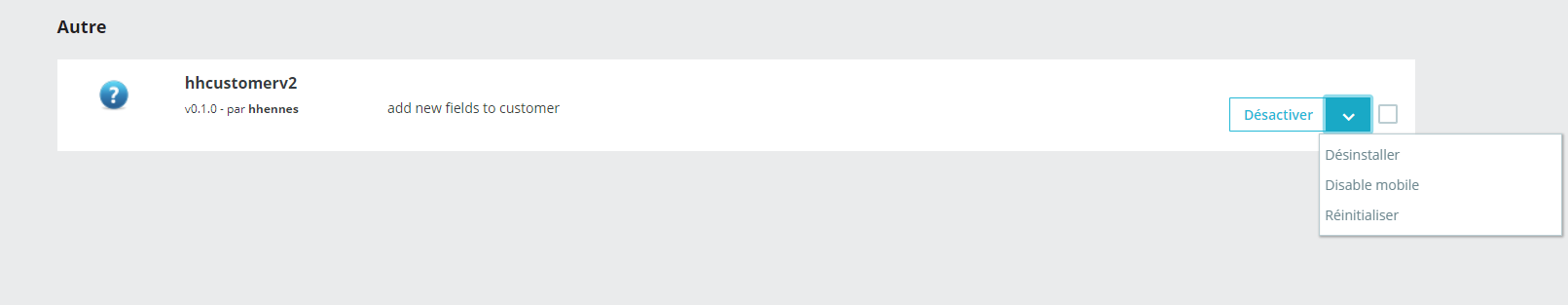
La démarche au niveau de code reste donc la même, on l’adapte au nombre de champs à créer.

Comme convenu le champs est bien obligatoire dans front-office comme dans le back-office. Dans l’idée de vouloir sécuriser un peu plus la véracité de l’information, nous avons mis un contrôle de caractère. Il est bien actif pour l’interface client :

Cependant ce n’est pas le cas du côté de l’administration PrestaShop, le code ne bride pas l’administrateur au nombre de caractères sur son interface.

Désinstallation :

Pour procéder à la désinstallation du module il est OBLIGATOIRE de passer par l’administration de PrestaShop.



En effet, le module n’est pas prévu pour supporter la modification manuelle d’une des tables concerné, ni pour la modification ou la suppression manuelle du dossier du module, ou bien encore pour la modification du code directement.

Si l’un de ces évènement venais à se produire, la désinstallation du module ne serait alors plus possible, les champs nouvellement créer ne serait plus actif ou bien la site ne fonctionnerait plus.

Une désinstallation propre depuis PrestaShop supprimera les champs de la base de données, ainsi que le dossier du module. Il ne restera qu’à supprimer la surcharge de classe « Customer ».