

# Rapport de Projet

## Systèmes d'Information Web — NFE114

### TORTEVOIS Alexandre

#### 1. Livrable du projet :

Adresse du site hébergé : <http://mywishlist.tortevois.fr/>

Un script d'installation automatisée est disponible (voir §7)

Pour les tests, la base de données a été remplie avec le script ./install/fill\_bdd.php

Exemple de compte test : user\_1 / @Zerty!7

RÉVISIONS				
Version	Nature de la modification	Auteur	Vérificateur	Date
01	Édition Originale	ALT	-	04/02/2020

C:\Users\alexandre.tortevois\Desktop\mywishlist\Rapport\_Projet\_NFE114.docx

## 2. Contexte

Notre client souhaite concevoir un site Internet permettant à ces utilisateurs enregistrés de créer et partager leur(s) liste(s) d'envie (nommé ci-après WishList) dont il renseignera le nom et une description de son choix.

Celles-ci contiendront des cadeaux (nommé ci-après Gift), qu'il pourra choisir dans la liste existante ou qu'il pourra créer en renseignant le nom et la description de son choix.

Un utilisateur pourra consulter les WishList des autres utilisateurs et « réserver » les cadeaux qu'il souhaite offrir au propriétaire de la WishList. Un cadeau déjà réservé ne pourra être réservé par un second utilisateur.

Toutes les fonctionnalités du site ne seront accessibles qu'aux utilisateurs connectés.

## 3. Analyse Fonctionnelle

Pour répondre au besoin de notre client, nous devons mettre en place les fonctionnalités suivantes :

### A. Gestionnaire d'utilisateurs, « espace membre »

Les actions utilisateur possibles seront :

- Inscription (champs obligatoires : username, password, email) suivi de l'envoi d'un email contenant un lien d'activation,
- Activation du compte via le lien fourni par email
- Envoie par email d'une nouvelle clé d'activation en cas de perte, non réception, ou de clé invalide
- Envoie d'un nouveau mot de passe en cas d'oubli
- Changement du mot de passe
- Authentification sur le site (login/logout)

### B. Gestionnaire de WishList

Les actions utilisateur possibles seront :

- Créer des WishList avec un nom personnalisable.
- Rendre Partagée ou Privée ses WishList.
- Modifier le nom de ses WishList
- Supprimer ses WishList
- Accéder à la liste de toutes ses WishList
- Partager ses WishList (envoi d'un lien par email à un ami)

### C. Gestionnaire de Gift

Les actions utilisateur possibles seront :

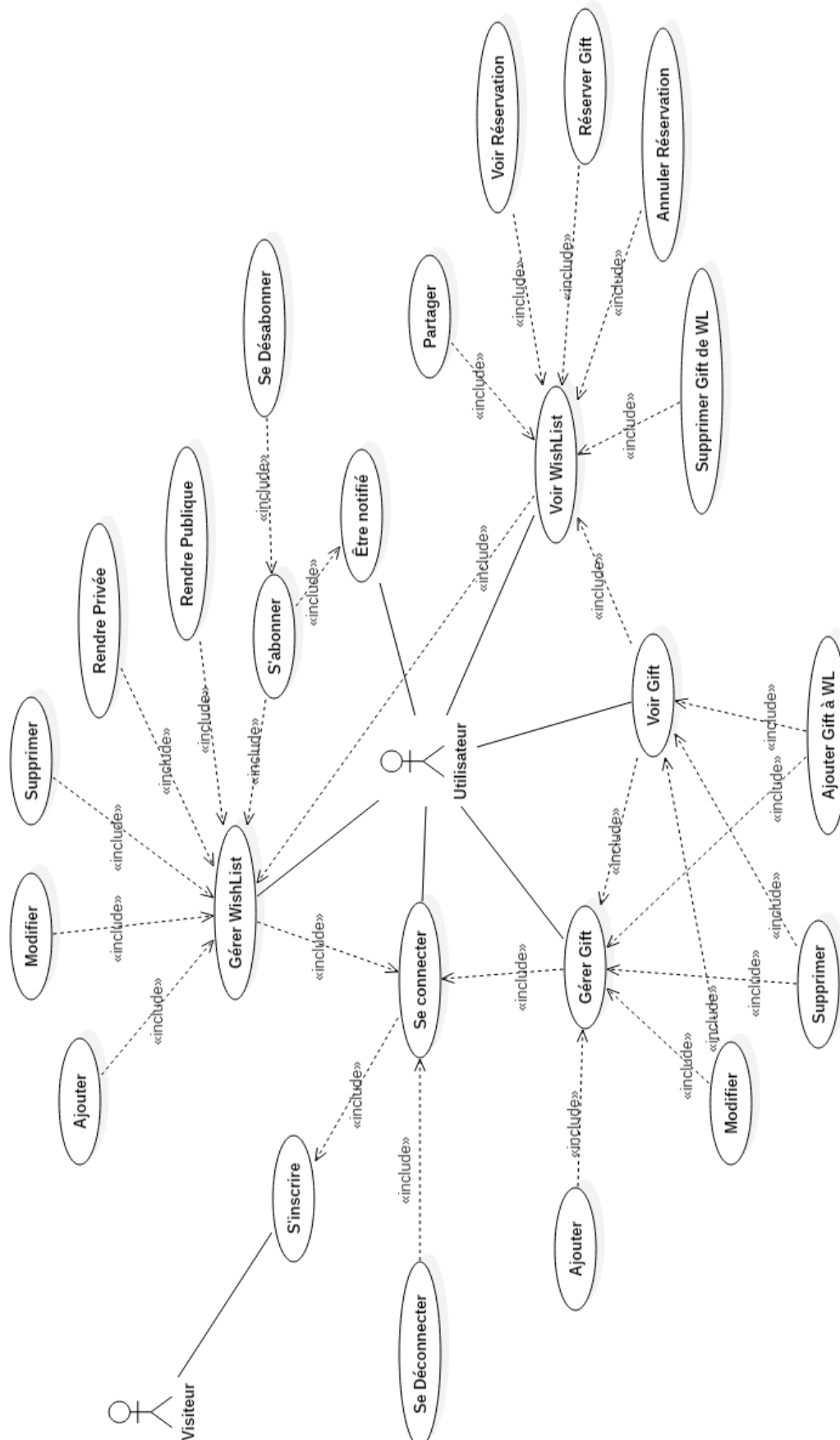
- Créer des Gift avec un nom et une description personnalisable.
- Modifier les Gift
- Supprimer les Gift – (dangereux, à valider avec le client)

### D. Lien WishList / Gift

Les actions utilisateur possibles seront :

- Ajouter un Gift dans une de ses WishList, s'il n'est pas déjà présent.
- Supprimer un Gift d'une de ses WishList
- Ajouter une réservation sur un Gift dans une WishList dont il n'est pas l'auteur
- Supprimer sa réservation sur un Gift

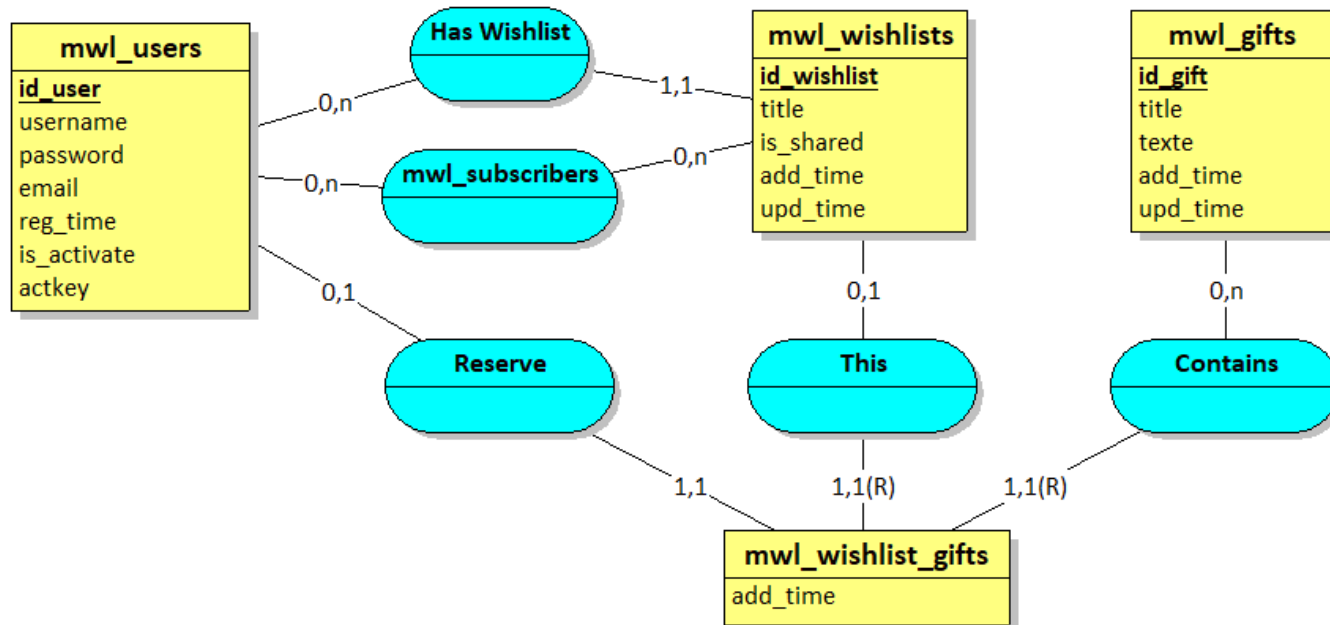
#### 4. Diagramme des Cas d'Utilisation



Un « Visiteur » ne peut que s'inscrire sur le site. Une fois son inscription validée, il devient un « Utilisateur ». Quand il se connecte sur le site, il a accès à toutes les fonctionnalités telle que décrite dans l'analyse fonctionnelle au §2.

## 5. Modèle Conceptuel de Données

À la suite de l'analyse fonctionnelle au §2, j'ai déduit le MCD suivant, dont dérivera la structure de la base de données :



## 6. Contexte technique retenu

### A. Contexte de développement

Un PC sous Windows 10 avec EasyPHP Devserver 17 et les versions de logiciels suivantes :

- Apache 2.4.25
- PHP 5.6.30
- MySQL 5.7.17
- phpMyAdmin 4.7.0

Les mises à jour d'EasyPHP sont possibles mais désormais payantes (10\$/ans).

Pour faire mes tests d'envoi d'email, j'ai utilisé hMailServer 5.6 sur lequel j'ai dû [configurer le « SMTP Relayer »](#), le port 25 étant bloqué par mon provider Free.

### B. Contexte de production

J'ai retenu OVH chez qui je suis historiquement client depuis 2007. J'ai un hébergement mutualisé « plan Perso » et le domaine **tortevois.fr** depuis 2012, hébergé dans le datacenter de Paris.

Les versions logicielles proposées par OVH sont les suivantes :

- PHP 5.6.38
- MySQL 5.5.60
- phpMyAdmin 4.7.0

OVH propose des versions plus récentes de MySQL seulement pour les nouveaux clients, dont les bases sont installées dans le datacenter de Gravelines. Une migration pour les anciens clients est prévue courant 2019.

### C. Langages utilisés

Le projet a été développé avec l'utilisation de PHP, HTML5, CSS3 et Javascript. Pour des raisons pratique, j'ai également utilisé le framework jQuery, pour le « toggle password ».

Les mêmes fonctionnalités auraient pu être écrites directement en javascript ; mais cette version aurait été plus lourde à écrire avec entre autre des :

```
document.getElementById(idDeMonElement).className = « maClasse »
```

## 7. Mise en œuvre/installation

1. Créer préalablement une base de données MySQL, à l'aide de phpMyAdmin. (pour l'exemple : *mywishlist*)
2. Modifier le constructeur de la classe *My\_SQL* en remplaçant les variables de connexion dans *./class/class.mysql.php*

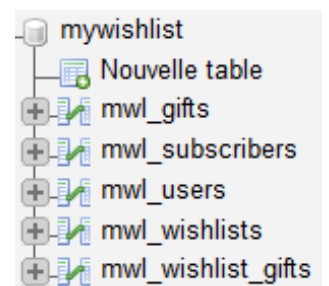
```
C:\Users\alexandre.tortevois\Desktop\mywishlist\class\class.mysql.php - Notepad++
Fichier Édition Recherche Affichage Encodage Langage Paramètres Outils Macro Exécution Modules d'extension Documents ?
class.mysql.php
12  /*
13  * Ouvre la connexion, gère suivant mode : CLI / localhost / serveur distant
14  */
15  public function __construct()
16  {
17      if ( preg_match('/[\\\/]/', $_SERVER['PHP_SELF']) ) // CLI : chemin absolu
18      {
19          $this->host = "localhost";        // serveur
20          $this->user = "root";            // nom d'utilisateur
21          $this->pass = "";               // mot de passe
22          $this->dbb = "mywishlist";      // nom de la base "locale"
23      }
24      else if( preg_match( '127.0.0.1', $_SERVER['HTTP_HOST']) ) // localhost
25      {
26          $this->host = "localhost";        // serveur
27          $this->user = "root";            // nom d'utilisateur
28          $this->pass = "";               // mot de passe
29          $this->dbb = "mywishlist";      // nom de la base "locale"
30      }
31      else // serveur distant
32      {
33          $this->host = "tortevois.mysql.db"; // serveur
34          $this->user = "";                // nom d'utilisateur
35          $this->pass = "";                // mot de passe
36          $this->dbb = "";                 // nom de la base
37      }
38
39      parent::__construct( $this->host, $this->user, $this->pass, $this->dbb );
```

3. Lancer le script d'installation situé dans *./install/install.php*.

La création des tables est automatisée, confirmer l'initialisation (popup de validation).

```
127.0.0.1/edsa-mywishlist/install/in X +
127.0.0.1/edsa-mywishlist/install/install.php?confirm=ok
Création de la table `mw1_gifts` ... OK
Création de la table `mw1_subscribers` ... OK
Création de la table `mw1_users` ... OK
Création de la table `mw1_wishlists` ... OK
Création de la table `mw1_wishlist_gifts` ... OK
Création des index de la table `mw1_gifts` ... OK
Création des index de la table `mw1_subscribers` ... OK
Création des index de la table `mw1_users` ... OK
Création des index de la table `mw1_wishlists` ... OK
Création des index de la table `mw1_wishlist_gifts` ... OK
Modification du champ `add_time` de la table `mw1_gifts` ... OK
Modification du champ `add_time` de la table `mw1_wishlists` ... OK
```

Résultat dans phpMyAdmin :



Nota : le même script permet également de faire une RAZ de la base de données.

## 8. Difficultés/problèmes rencontrés

### A. La version de MySQL sur le serveur OVH

Comme l'indique la [documentation MySQL](#) plusieurs colonnes de type `TIMESTAMP` et/ou `DATETIME` peuvent être initialisées et mise à jour automatiquement depuis MySQL 5.6.5. Étant en version 5.5.6 sur les serveurs mutualisés d'OVH, j'ai contacté le support client (voir Annexe A) et n'ayant pas eu une réponse favorable pour migrer sur une version de MySQL plus récente (sans remettre la main au porte-monnaie), pour pallier à ce manque sur le serveur de production j'ai dû implémenter la mise à jour des champs directement dans mes requêtes SQL de création ou de mise à jour :

```
reg_time = NOW(), add_time = NOW(), upd_time = NOW()
```

### B. La manipulation des REGEX :

C'est un des points toujours très délicat pour chaque projet : bien contrôler les données des formulaires avant insertion / modification en base de données.

Dans le cadre de ce projet, je me suis posé la question d'utiliser ou non les méta-séquences (comme par exemple `\w`, `\d` ...) où les catégories Unicode puisque j'ai choisi de travailler en UTF-8 (comme `\p{L}`, `\p{N}` ...).

J'ai implémenté un script pour réaliser différents essais sur l'ensemble des caractères disponibles en Unicode (dans `./unit_test/test_unicode.php`). Finalement, j'ai choisi pour une meilleure lisibilité de mon code et être certain de ce qui est validé par mes REGEX de rester avec les classes de caractères déclarées « standard ».

Pour renforcer la sécurité des utilisateurs, j'ai choisi de forcer des mots de passe avec au moins :

- 1 majuscule,
- 1 chiffre
- 1 caractère spécial parmi la liste : `_&~#{ }()[ ]|^@%*+|-\/=$<>!?.,;:`

J'ai pas mal tâtonné avant de trouver la bonne REGEX pour bloquer la validation du formulaire puis valider cette entrée dans ma fonction `$form->check_vars()`.

Après recherche, [ce post](#) m'a bien aidé à comprendre comment mieux construire mes REGEX.

Deux scripts de tests unitaires sur la `class.form` m'ont également aidé dans l'ajustement de mes REGEX : voir dans `./unit_test/class.form/` → `check_vars.php` et `check_password.php`. Ces scripts sont disponibles à l'exécution par le navigateur (mais ne génère pas une « vraie » page HTML, c'est juste de l'affichage pour du test), ou en mode CLI (voir Annexe B pour la mise en œuvre).

Nota : Depuis HTML5, la vérification des *input* peut se faire directement côté client par le navigateur en utilisant les attributs *required* et *pattern*. Il n'est donc plus nécessaire d'utiliser du Javascript pour réaliser cette fonctionnalité. Cependant, par sécurité, il est indispensable de garder les vérifications côté serveur.

### C. Les styles CSS

J'ai réalisé le développement de la feuille de style sous l'éditeur proposé par Mozilla Firefox.

Il me reste cependant quelques bugs que je n'ai pas réussi à corriger dans le temps imparti pour ce projet.

#### Sous Chrome :

- « L'œil » pour la fonction « toggle password » est décalé d'1px en haut sur le formulaire de modification du mot de passe :

The screenshot shows three input fields for a password form. The first two are labeled 'Password' and the third is labeled 'Confirmer'. Each field has a small eye icon to its right for toggling visibility. The eye icon on the first 'Password' field is slightly higher than the others, indicating a CSS misalignment.

#### Sous Edge :

- La propriété ***background-image: linear-gradient*** est mal interprété :



▲ Titre ▼	▲ Username ▼	Action
-----------	--------------	--------

## D. Le temps

Pour ce projet, le facteur temps a été la plus grosse contrainte pour moi. Un temps préalable de réflexion pour anticiper l'intégration de fonctionnalités futures « sans tout casser » aurait été nécessaire pour mieux mener ce projet. Une passe *Design Pattern*, que j'étudie en parallèle avec l'unité NFP121, aurait permis d'améliorer la structure et l'organisation de mon code et de limiter les difficultés pour intégrer les nouvelles fonctionnalités.

Pour éviter une redondance de copie de code et améliorer sa maintenabilité, il serait nécessaire de développer avant toute nouvelle implémentation de fonctionnalités :

- une classe de gestion des erreurs, évitant la copie des fonctions *set\_error*, *get\_error*, *has\_error*, *merge\_error* dans toutes les classes et l'utilisation pouvant s'avérer hasardeuse des *\$GLOBALS* dans la classe *page*.  
Mon idée à la base était de gérer des codes erreurs sous forme « binaire » afin d'être capable à partir d'une seule variable numérique de lister toutes les erreurs survenues dans l'exécution du code. Le code erreur pouvant permettre de vérifier le bon fonctionnement des fonctions grâce à un/des scripts de tests unitaires externes.  
Il faudrait réfléchir à un système suffisamment généraliste mais tout en conservant la précision permettant de savoir d'où provient une erreur, peut-être en regardant les possibilités amenées avec l'utilisation des « exceptions ».  
Nota : il y a certainement des frameworks qui gèrent déjà tout ça très bien, un temps de recherches et tests préliminaires serait nécessaire avant de faire un choix et de poursuivre le développement.
- une classe pour l'envoi des mails, évitant la copie de la fonction *send\_mail()* dans les classe *user/wishlist/gift*. Cela ne demanderait pas un gros travail de mise en œuvre et ne vas pas « tout casser » dans le code existant.
- une classe permettant de gérer les notifications en implémentant *SpISubject* et *SpIObserver* comme dans cet [exemple d'utilisation de l'observateur-observer](#). Il faudrait réfléchir s'il est nécessaire d'ajouter des codes de notifications pour personnaliser celle-ci en fonction de l'événement survenu (ajout, mise à jour, suppression, réservation, ... sur un Gift, sur une WishList ...)
- finaliser quelques fonctionnalités de bases. J'ai laissé quelques « TODO » dans le code dont je n'ai pas eu le temps de finaliser l'implémentation. Cependant, cela ne gêne en rien le fonctionnement du site dans le cadre d'une utilisation « normale », car ce sont des optimisations ou des améliorations à ajouter.

```
Find result - 23 hits
Search "TODO : " (23 hits in 6 files)
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\class\class.form.php (2 hits)
Line 367: // TODO : ajouter ici la gestion des longueurs / ajouter tous les champs ... reprendre les tests unitaires !
Line 523: // TODO : ajouter la $page number pour le retour
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\class\class.gift.php (7 hits)
Line 24: // TODO : améliorer cette initialisation "brute"
Line 60: // TODO : check if exist
Line 288: if( $row->id_user == $this->id_session || ($row->id_user != $this->id_session && $row->is_shared == 1) ) // TODO : to include in SQL
Line 436: // TODO : if( $this->notify() )
Line 462: // TODO : if( $this->notify() )
Line 505: // TODO : ajouter des codes notifications
Line 565: private function sql_check_id_wishlist() // this function is copied from class.wishlist ... TODO : should be optimized !
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\class\class.page.php (3 hits)
Line 23: // TODO : DANGER $GLOBALS !!
Line 95: // TODO : if( isset($row['actions']['select']) ) ??
Line 439: // TODO : à optimiser pour ne pas parcourir tout le buffer ...
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\class\class.user.php (1 hit)
Line 107: // TODO : des cookies (pour 4h!)
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\class\class.wishlist.php (9 hits)
Line 20: // TODO : améliorer cette initialisation "brute"
Line 56: // TODO : check if exist
Line 115: // TODO : if( $this->notify() )
Line 145: // TODO : if( $this->notify() )
Line 221: // TODO : $this->notify()
Line 242: // TODO : $this->notify()
Line 266: // TODO : check id_user ?
Line 287: // TODO : check id_user ?
Line 320: // TODO : ajouter des code aux notifications
C:\Users\alexandre.tortevois\ARECO\OneDrive - ARECO\CNAM\mywishlist\user.php (1 hit)
Line 163: if( !isset($_POST['submit']) || $user->has_error() ) // TODO : be careful isset($user) ??
```

Par exemple, dans ma classe *form*, certaines de mes REGEX sont définie dans la fonction *get\_pattern* car elles sont « partagées » avec la vérification post-soumission du formulaire rendue possible en HTML5, quand d'autres REGEX sont définies directement dans la fonction *check\_vars*, parce qu'initialement elles étaient toutes dans *check\_vars*). Il faudrait uniformiser et basculer toute la définition des REGEX dans la fonction *get\_pattern*.

Idem pour les contrôles de longueur autorisée sur les champs, il faudrait plutôt les définir dans une seule fonction, pour éviter les bugs si on change une valeur à un seul endroit dans le code.

Cela demande de revoir toute la programmation de *check\_vars*.

Pour le projet, j'avais également prévu ces améliorations/fonctionnalités, mais le manque de temps m'a fait renoncer à leur intégration :

- implémenter l'export des WishList en PDF ; j'ai déjà pu utiliser la bibliothèque FPDF dans le cadre de mon travail pour générer des rapports de validation de machine (voir dossier « areco »).
- utiliser la classe PDO à la place de mysqli pour mieux se prémunir des injections SQL.
- ajouter le système de [recaptcha](#) de Google pour « limiter » les inscriptions des robots.
- ajouter une sécurisation de la globale \$\_SESSION qui est utilisée dans certaines requêtes SQL et rajouter les cookies en suivant les préconisations de [cet article](#).
- lier la table « Gift » avec l'API Amazon en suivant [cet article](#). Cela éviterai des doublons d'article puisqu'on pourrait stocker l'ItemId Amazon et le titre (et peut être d'autres infos en se servant de notre BDD comme d'un cache pour éviter des requêtes chez Amazon). Cela impliquerait implicitement l'ajout des « catégories » sur les Gifts. Au préalable, il faudrait valider si cela est possible avec un hébergement mutualisé OVH.
- réfléchir à l'implémentation d'une fonction « recherche » pour les gift, les wishlist, les username ...
- voir s'il serait intéressant de définir un système de droits pour les actions des utilisateurs, notamment pour l'édition voir la suppression des Gifts.
- regarder le fonctionnement et la mise en place de tests unitaires avec PHPUnit ; certainement plus poussé que les tests que j'ai pu réaliser.
- ajouter d'autres idées qui viendraient au fil de l'implémentation des fonctionnalités ...

## 9. Conclusion

---

À la vue de la masse de travail importante à produire, je me suis lancé très (et certainement trop) rapidement dans la programmation. Ce projet m'a fait prendre conscience de l'importance de la phase d'étude et de définition du projet qui doit être le « socle » de base de tout projet (informatique ou non).

On ne réfléchit jamais assez à toutes les possibilités d'évolutions futures et comment elles pourraient être implémentées, de façon à produire des briques de code de bases qui soient le plus évolutives et adaptables possibles par la suite. Ce temps de réflexion n'est pas du temps « perdu » mais évite une dérive exponentielle des coûts de développement (temps, énergie, argent) au fur et à mesure de l'avancement ou de l'implémentation de nouvelles fonctionnalités dans le projet.

Si les fondations sont bancales, toute l'implémentation du logiciel produite sera complexe voir instable et cela restera très difficile à « rattraper » et coûtera énormément plus que ce qu'aurait coûté le temps de la réflexion au départ avec le risque de remettre en cause totalement certaines « briques » et de devoir refaire le travail plusieurs fois !

Malgré le manque de temps et les améliorations encore possibles, je reste satisfait du code produit en partant de « zéro ». Je compte réutiliser ce travail, pour partager des listes d'envie avec la famille et les amis, mon beau-frère et ma belle-sœur fêtant leur 25ans de mariage cet été.

## 10. Logithèque

---

Pour le code : [Notepad++](#)

Diagramme UML : [StarUML](#)

MCD : [Looping](#)

## Annexes

### 11.Échange avec le support Client OVH

**Sujet du ticket :**

Base de données

**Ouvert depuis le :**

23/12/2018 09:34

**Service concerné :**

tortevois.fr

**Mise à jour le :**

13/01/2019 08:30

**Type de ticket**

Requête générique

Réponse du 26/12/2018 08:53

**De : Support OVH**

Bonjour Monsieur TORTEVOIS,

Merci pour votre retour.

Vous avez créé une base de données sur un ancien hébergement installé dans le data center à Paris.

L'achat d'un nouvel hébergement sera installé dans le datacenter de Gravelines qui vous permet de créer une base de données 5.6 mysql par défaut.

Si vous souhaitez créer des bases de données 5.6 mysql sur l'hébergement tortevois.fr, vous pouvez commander un sql privé qui vous permet de choisir la version de vos bases de données lors de sa création.

Pour commander un sql privé, veuillez suivre cette démarche :

- À partir de votre espace client, Commander, Bases de données

Pour plus d'informations sur le sql privé, depuis ce lien :

<https://www.ovh.com/tn/hebergement-web/options-sql.xml>

Je reste à votre disposition pour toute demande complémentaire.

Cordialement,

—

Wassim B.

Technicien Support IT - Web

Réponse du 08/01/2019 08:06

**De : Support OVH**

Bonjour Monsieur TORTEVOIS,

Veuillez nous excuser pour la réponse tardive.

Je comprends parfaitement que cette situation est inconfortable pour vous.

Je suis navré de ne pas pouvoir vous aider.

La seule solution dans votre cas, c'est de commander un sql privé afin de choisir le mysql 5.6.

Je reste à votre disposition pour toute demande complémentaire.

Cordialement,

—

Wassim B.

Technicien Support IT - Web

## 12.Utiliser PHP en ligne de commande sous Windows (Mode CLI)

- Installer « ansicon »

Ce plugin fournit une reconnaissance de séquence d'échappement ANSI pour les programmes de console Windows et permet la gestion des couleurs dans la console.

Télécharger la dernière release disponible : <http://ansicon.adoxa.vze.com/>

Dézipper à la racine de votre disque C:

Lancer une invite de commande (cmd.exe) en mode administrateur (CTRL+SHIFT+Entrée)

Aller dans « ansicon\x64 » avec la commande cd.

Exécuter « ansicon -i ».

Se reporter au fichier sequences.txt pour les codes couleurs.

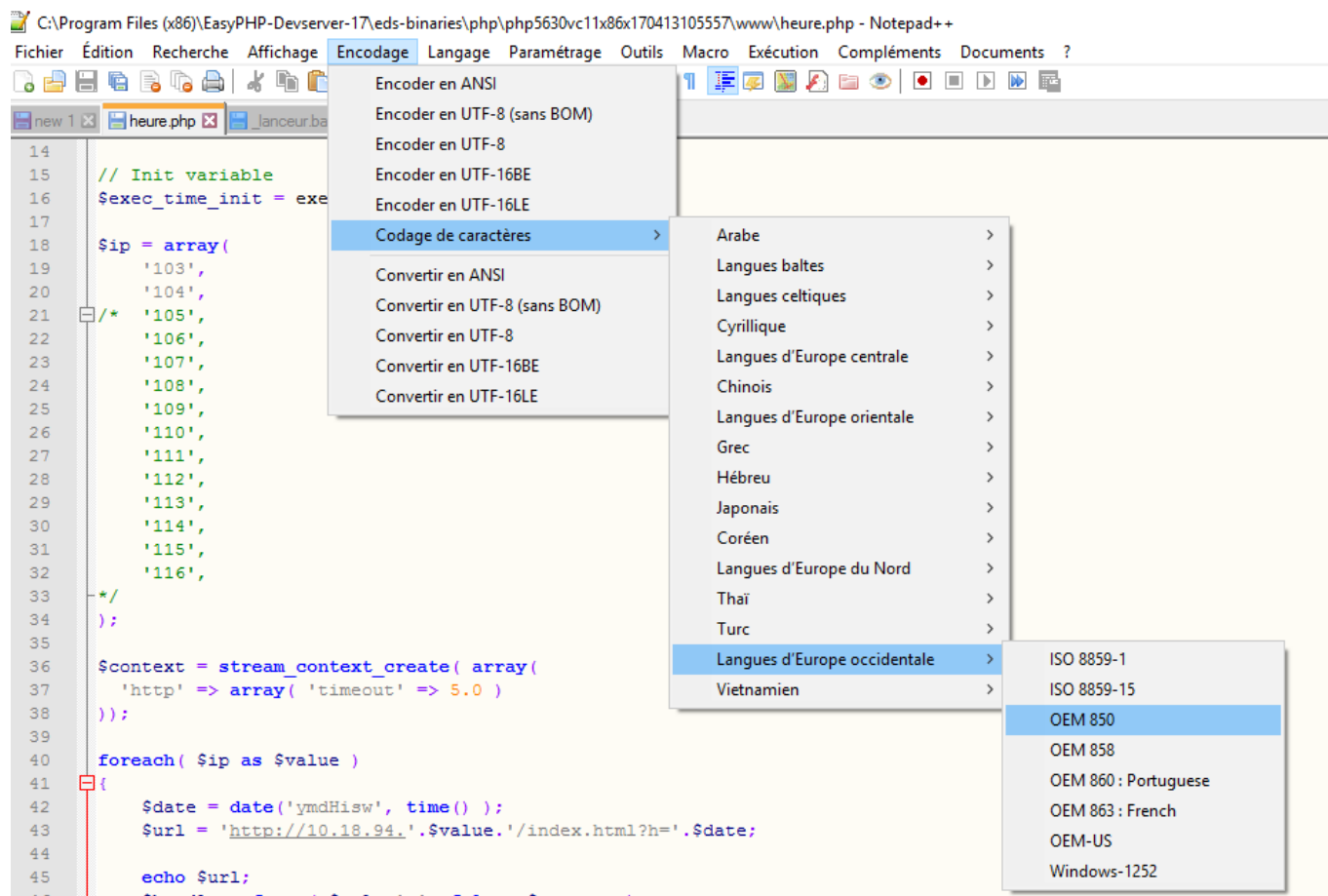
```
Administrator: Invite de commandes
Microsoft Windows [version 10.0.17134.407]
(c) 2018 Microsoft Corporation. Tous droits réservés.

C:\WINDOWS\system32>cd c:/ansicon/x64

c:\ansicon\x64>ansicon -i
```

- Gestion des caractères accentués

Le mode console ne gère pas les accents sauf si on encode les fichiers en OEM850.



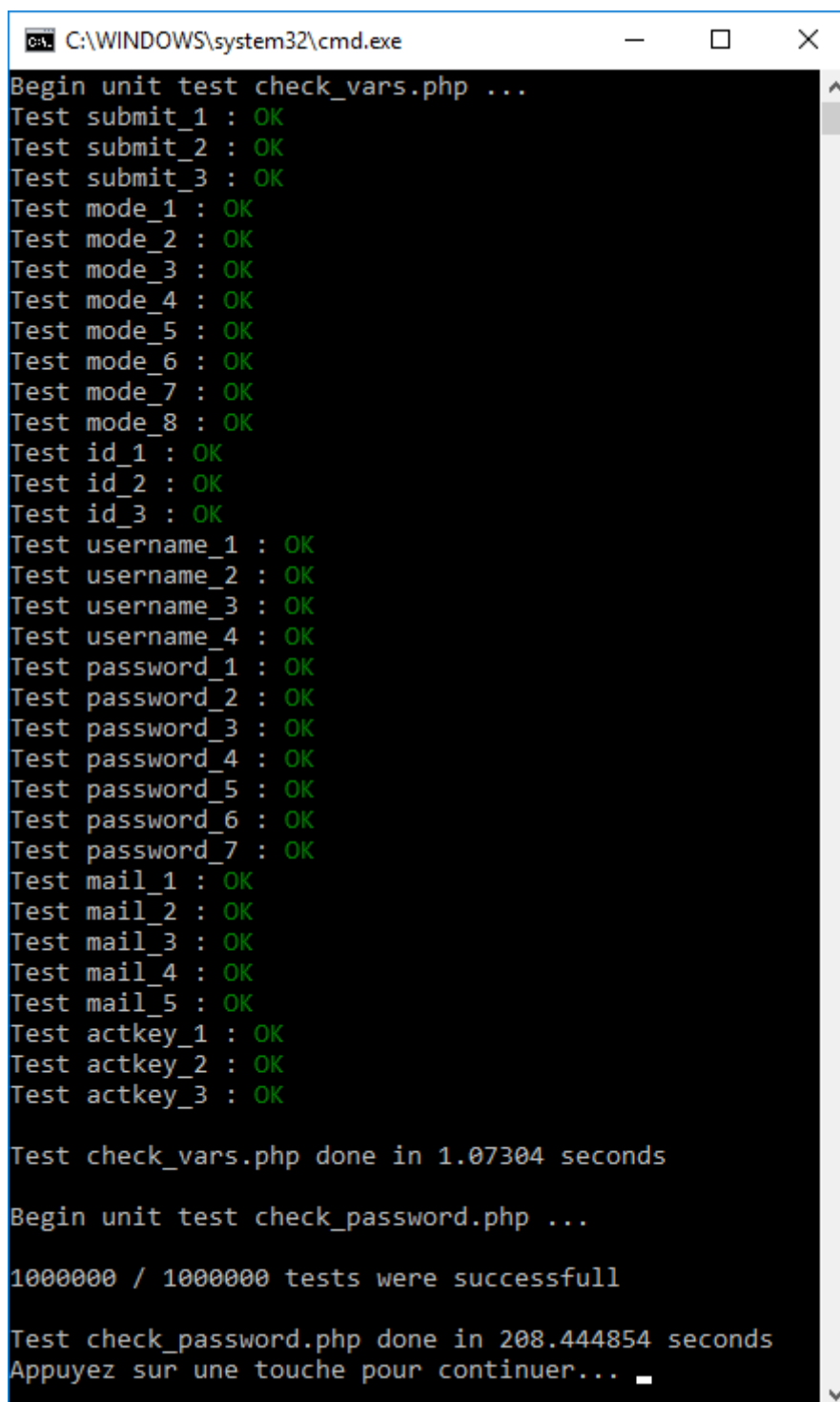
Pour pallier à cette modification d'encodage, j'ai implémenté la fonction *strip\_accented* qui enlève les caractères accentués à l'affichage (dans ./functions.php).

- Créer un « lanceur » de script

Il est possible de créer un fichier « **.bat** » pour automatiser le lancement de l'exécution d'un script PHP.  
La structure du fichier est la suivante :

```
@echo OFF
"lien exécutable php.exe" "lien script.php" "éventuels arguments"
pause();
```

Exemple de scripts de test unitaire lancer en mode console :



```
C:\WINDOWS\system32\cmd.exe

Begin unit test check_vars.php ...
Test submit_1 : OK
Test submit_2 : OK
Test submit_3 : OK
Test mode_1 : OK
Test mode_2 : OK
Test mode_3 : OK
Test mode_4 : OK
Test mode_5 : OK
Test mode_6 : OK
Test mode_7 : OK
Test mode_8 : OK
Test id_1 : OK
Test id_2 : OK
Test id_3 : OK
Test username_1 : OK
Test username_2 : OK
Test username_3 : OK
Test username_4 : OK
Test password_1 : OK
Test password_2 : OK
Test password_3 : OK
Test password_4 : OK
Test password_5 : OK
Test password_6 : OK
Test password_7 : OK
Test mail_1 : OK
Test mail_2 : OK
Test mail_3 : OK
Test mail_4 : OK
Test mail_5 : OK
Test actkey_1 : OK
Test actkey_2 : OK
Test actkey_3 : OK

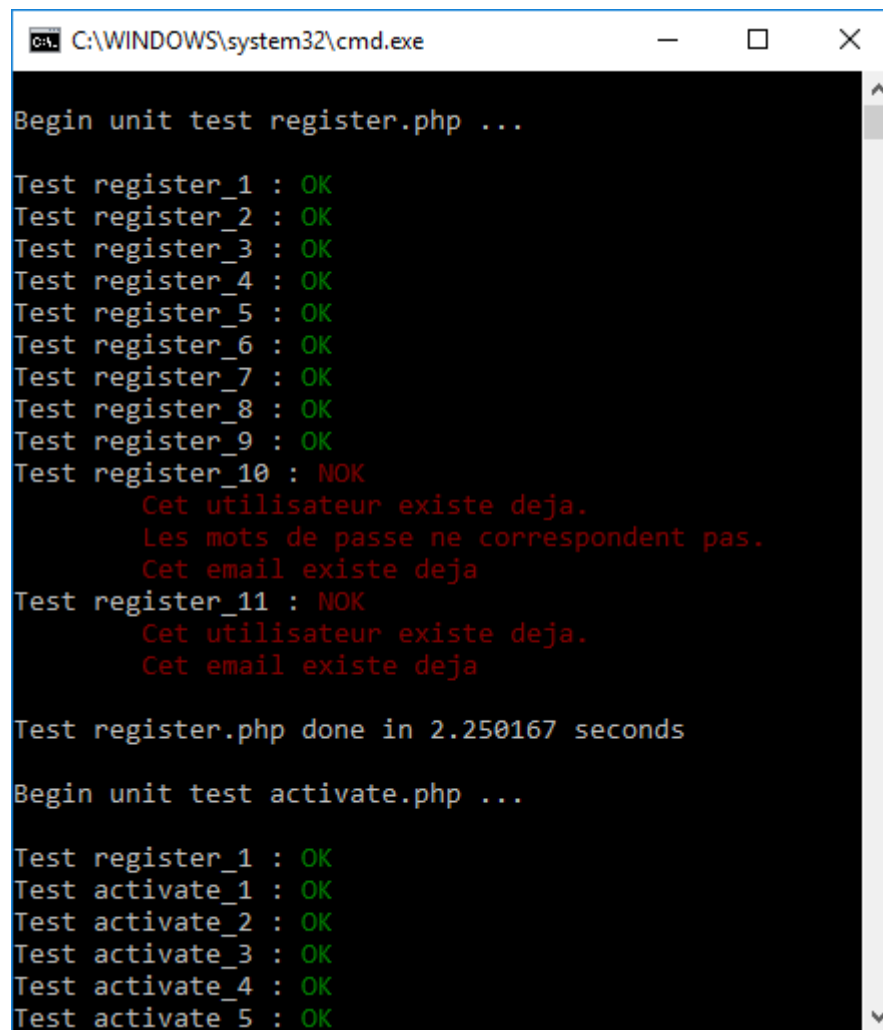
Test check_vars.php done in 1.07304 seconds

Begin unit test check_password.php ...

1000000 / 1000000 tests were successfull

Test check_password.php done in 208.444854 seconds
Appuyez sur une touche pour continuer...
```





```
C:\WINDOWS\system32\cmd.exe

Begin unit test register.php ...

Test register_1 : OK
Test register_2 : OK
Test register_3 : OK
Test register_4 : OK
Test register_5 : OK
Test register_6 : OK
Test register_7 : OK
Test register_8 : OK
Test register_9 : OK
Test register_10 : NOK
    Cet utilisateur existe deja.
    Les mots de passe ne correspondent pas.
    Cet email existe deja
Test register_11 : NOK
    Cet utilisateur existe deja.
    Cet email existe deja

Test register.php done in 2.250167 seconds

Begin unit test activate.php ...

Test register_1 : OK
Test activate_1 : OK
Test activate_2 : OK
Test activate_3 : OK
Test activate_4 : OK
Test activate_5 : OK
```