|  |
| --- |
| ETML |
| Titre de test |
| [Sous-titre du document] |

|  |
| --- |
| Jeremiah Steiner  [Date] |

Table des matières

[1 Analyse préliminaire 2](#_Toc71272354)

[1.1 Introduction 2](#_Toc71272355)

[1.2 Objectifs 2](#_Toc71272356)

[1.3 Planification initiale 2](#_Toc71272357)

[2 Analyse / Conception 2](#_Toc71272358)

[2.1 Concept 2](#_Toc71272359)

[2.2 Stratégie de test 2](#_Toc71272360)

[2.3 Risque techniques 2](#_Toc71272361)

[2.4 Planification 2](#_Toc71272362)

[2.5 Dossier de conception 2](#_Toc71272363)

[3 Réalisation 2](#_Toc71272364)

[3.1 Dossier de réalisation 2](#_Toc71272365)

[3.2 Description des tests effectués 7](#_Toc71272366)

[3.3 Erreurs restantes 7](#_Toc71272367)

[3.4 Liste des documents fournis 7](#_Toc71272368)

[4 Conclusions 7](#_Toc71272369)

[4.1 Bilans 7](#_Toc71272370)

[5 Annexes 7](#_Toc71272371)

[5.1 Résumé du rapport du TPI / version succincte de la documentation 7](#_Toc71272372)

[5.2 Sources – Bibliographie 7](#_Toc71272373)

[5.3 Journal de travail 7](#_Toc71272374)

[5.4 Manuel d’Installation 7](#_Toc71272375)

[5.5 Manuel d’Utilisation 7](#_Toc71272376)

[5.6 Archives du projet 7](#_Toc71272377)

[5.7 Glossaire 7](#_Toc71272378)

# Analyse préliminaire

## Introduction

## Objectifs

## Planification initiale

# Analyse / Conception

## Concept

## Stratégie de test

## Risque techniques

## Planification

## Dossier de conception

ici

Je ne suis pas sûr de voir la différence entre cette partie et la partie réalisation

ici

# Réalisation

## Dossier de réalisation

(capture)

### Controller

La class controller contient 2 méthodes en protected, celles-ci sont nécessaires dans plusieurs des classes filles.

#### Protected

##### Gestion des images

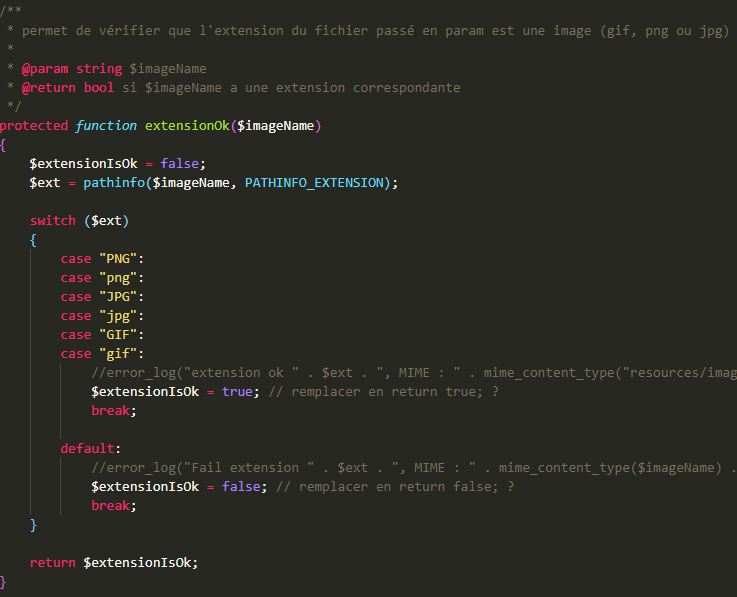


Figure 1 extensionOk, vérification que le fichier a bien une extension d'image

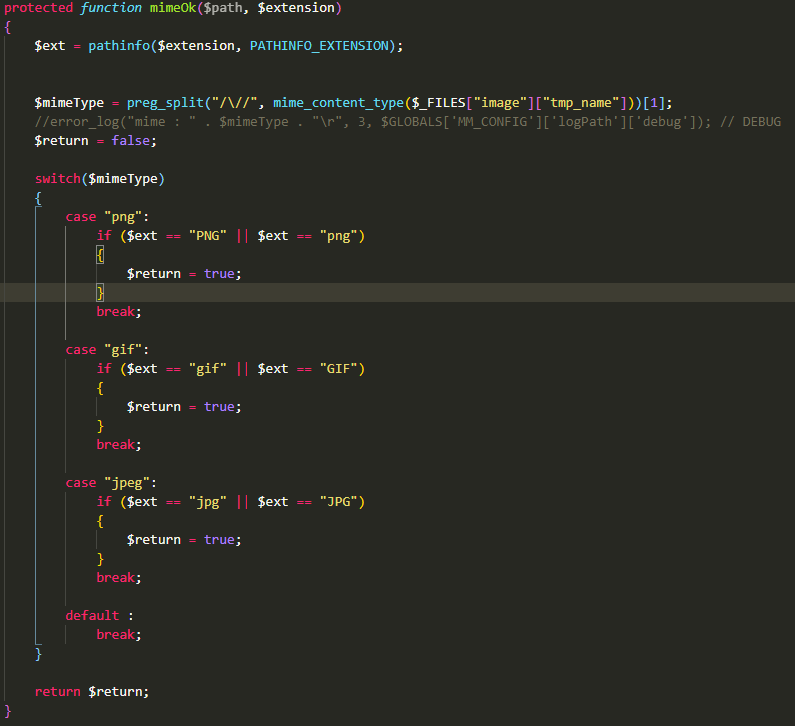


Figure 2 mimeOk, vérification que le mime type est bien le même que l'extension

### Upload d’image de profile

#### Introduction

Pour réaliser l’upload d’image de profile il a été nécessaire d’exécuter plusieurs recherches. Premièrement sur le moyen d’enregistrer l’image, en suite d’effectuer une vérification de son type Mime et de son extension, puis sur le moyen de la convertir en jpeg afin qu’elle ne prenne pas trop de place, puis pour finir d’effacer l’ancienne image (toujours dans une optique d’occuper le moins d’espace possible).

#### Réalisation

##### Création de l’utilisateur

Une image par défaut est assignée aux nouveaux utilisateurs, le nom de cette image inscrit dans la base de donnée, l’image toujours présente dans les dossiers du site, sur l’image ci-dessous l’on peut voir « useImage » qui a comme valeur par défaut le nom « defaultUserPicture.png ».

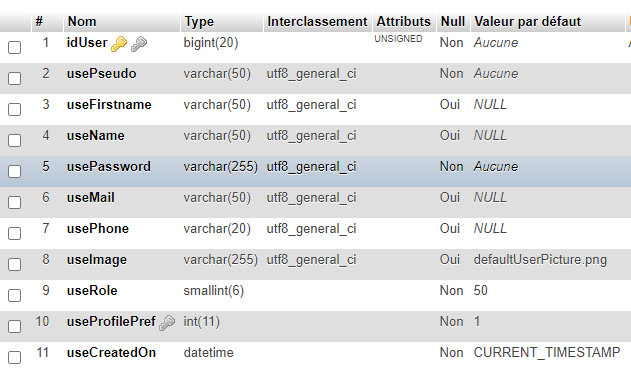


Figure 3 table t\_user de la database, useImage.

L’endroit où cette image est stockée est le suivant : >> ressources >> image >> Users.

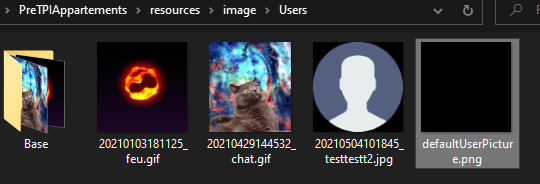


Figure 4 stockage de l'image de profile

##### Création du form

Il est maintenant nécessaire de récupérer l’image que l’utilisateur souhaite utiliser, Pour ce faire le formulaire doit avoir method=’’post’’ et enctype=’’multipart/form-data’’.

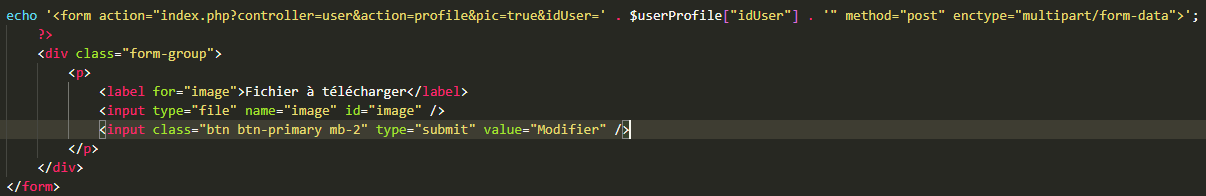


Figure 5 form de modification de l'image du profile

##### Vision utilisateur

Ici l’on peut voir le formulaire vu depuis un utilisateur standard, on y trouve l’image actuelle, le bouton de sélection de fichier et le bouton de validation de modification.

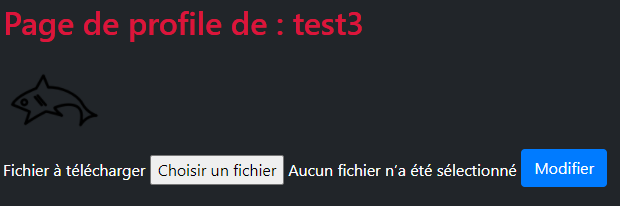


Figure 6 form de modification d'image de profile (avant)

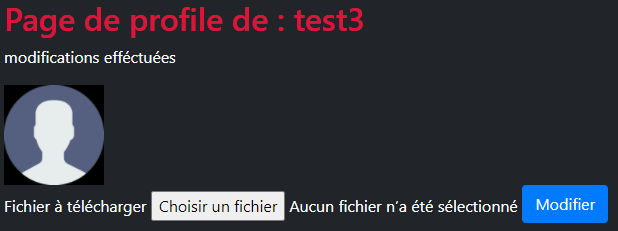


Figure 7 form de modification d'image de profile (après)

##### Côté serveur

En premier il y a la vérification du fichier (extension + mime type, voir section : [Controller](#_Controller)), puis s’il est conforme on entre dans la condition.

On y enregistre le nom de l’image ainsi que sa taille nécessaire pour le traitement des png.

Arrive ensuite le switch permettant de gérer la conversion des différents formats vers le jpeg.

Dans le cas des png, il a été nécessaire d’ajouter une limite de taille que l’on peut voir dans le case png et PNG.

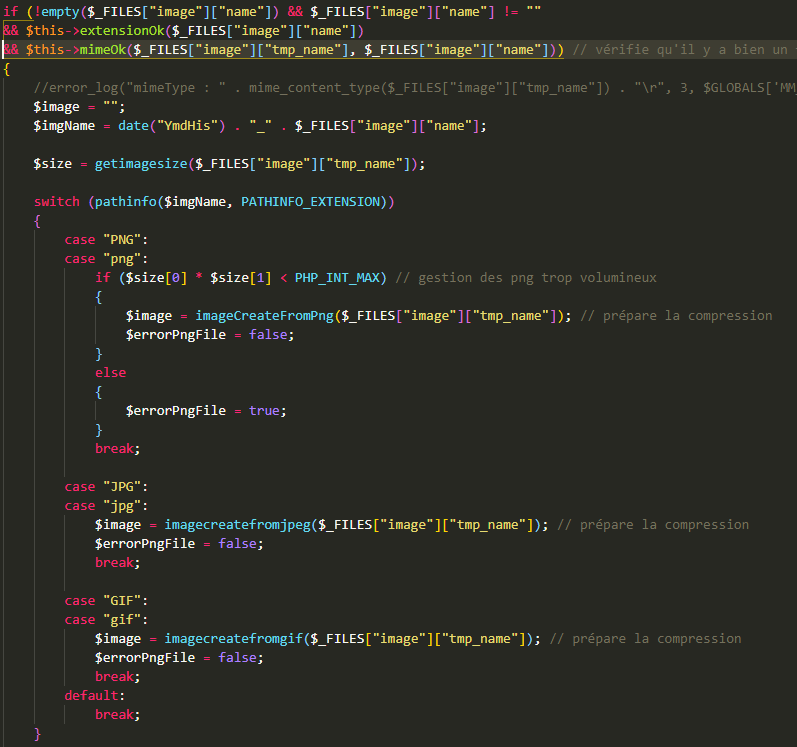


Figure 8 enregistrement de l'image de profile part 1/3

Ensuite si l’image a pu être correctement convertie (pas de png trop volumineux), on commence alors par détruire la précédente image stockée sur le serveur (unlink). Puis l’on crée le nouveau nom de l’image en modifiant l’extension. Ensuite imagejpeg compresse et enregistre l’image à l’endroit désiré. On finit par modifier les valeurs useImage de l’utilisateur courant afin d’éviter des erreur php sur la page de profile



Figure 9 enregistrement de l'image de profile part 2/3

Il est maintenant nécessaire d’apporter les modifications à la database. Après vérification des erreurs possibles On met à jour la base de donnée, puis on réinitialise la variable $userProfile qui contient les informations nécessaires pour afficher la page de profile de l’utilisateur.

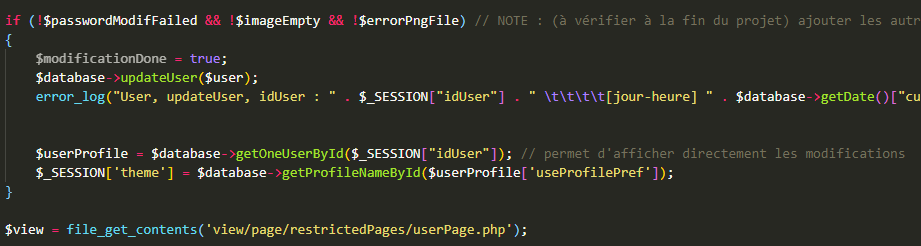


Figure 10 enregistrement de l'image de profile part 3/3

## Description des tests effectués

## Erreurs restantes

## Liste des documents fournis

# Conclusions

## Bilans

### Bilan planification

### Bilan personnel

### Bilan global

# Annexes

## Résumé du rapport du TPI / version succincte de la documentation

### Situation de départ

Créer un site de vente/location de biens immobilier (par nature), où des utilisateurs peuvent se connecter avec leur compte afin de créer une liste personnelle. La notation desdits biens doit être possible. Le pourvoyeur de ceux-ci peut les retirer du marché, les masquer ou les modifier.

Les Biens, les utilisateurs ont plusieurs caractéristiques modifiables.

### Mise en œuvre

Pour résoudre la situation initiale il est nécessaire de créer une base de données, avec des tables pour les utilisateurs, les biens, les biens de la liste personnelle, les catégories, la notion de rating et les profiles.

Le site utilise du javascript, php, css et html à l’aide de bootstrap.

Visual studio code a donc été choisi comme éditeur de code.

Une architecture MVC a été sélectionnée pour la construction du site, les contrôleurs gérant l’accès aux pages et permettant l’affichage des données de la DB grâce au modèle

### Résultats

Le site est atteignable et les accès sont sécurisés (mot de passe hashé, pas de TLS/SSL). La liste personnelle est opérationnelle, elle permet de noter un biens (système d’upvote). Les biens sont lister dans un tableau, une page de détail est disponible pour avoir davantage de détails sur les biens.

Une page personnelle permet d’ajouter des biens et de les gérer, ainsi que de modifier son profile et préférences. Une page d’accueil a été créée et contient un carrousel contenant trois biens, le dernier ajout au site, le bien avec le prix le plus bas et celui qui a le plus d’upvote.

## Sources – Bibliographie

### Mime Type

#### Stack overflow : [mime](https://stackoverflow.com/questions/2295897/problem-with-imagecreatefrompng-php) ? il faut mettre les liens en clair ? (si on imprime)

#### Fonctions :

1. [getimagesize()](https://www.php.net/manual/fr/function.getimagesize.php)
2. [imagecreatefrompng()](https://www.php.net/manual/fr/function.imagecreatefrompng.php)
3. [mime\_content\_type()](https://www.php.net/manual/en/function.mime-content-type.php)

## Journal de travail

## Manuel d’Installation

## Manuel d’Utilisation

## Archives du projet

## Glossaire

|  |  |  |
| --- | --- | --- |
| **Mots ou phrase** | **(Chapitres)** | **Description** |
| DB | .. | Base de donnée |
| MVC | .. | Model View Controller, architecture logicielle séparant les interfaces utilisateurs(V) des données(M), et également séparer de la logique de traitement des informations(C). |
| Protected | [A](#_Protected) | Visibilité de propriétés ou de méthodes, protected signifie une visibilité réservée aux membres de la classe ou à ceux de ses classes filles. |
| Type MIME | [A](#_Gestion_des_images), [B](#_Côté_serveur) | Type de médias, permet d’identifier le format de données. Exemples : image/png, text/css, [etc…](https://fr.wikipedia.org/wiki/Type_de_m%C3%A9dias) |
|  |  |  |
|  |  |  |