## Dossier de Projet

## « Développeur logiciel »



## 

## Par Yoann Mellier

Soutenance pour le titre de « Développeur Logiciel » Session 2017/2018

# Table des matiéres

REMERCIEMENTS 3

Résumé du projet en anglais 4

Tables des figures 5

Liste des compétences couvertes par le projet 7

Cahier des charges 8

Introduction 8

Contexte 8

Description de la demande 9

Les objectifs 9

Les fonctions 9

Contraintes 10

Solution proposée 10

Interface graphique 10

Technologies utilisées 14

Modèle Conceptuel de Données 16

Modèle Logique de Données 17

Persistance des données 18

Dictionnaire des données 18

Exemples de requêtes SQL 23

Exemple 1 : 23

Exemple 2 : 24

Exemple 3 : 24

Spécifications fonctionnelles 25

Profil utilisateur 25

Profil administrateur 26

Spécifications techniques 26

Cas d’utilisation (Users story mapping) 26

Réalisation 30

Architecture 30

Interface graphique 39

Rendu final version v1.1 40

Accueil 40

Détails de l’annonce 40

Formulaire de mise en contact 41

Mise en contact 41

Mise en contact 42

Gestion de projet 42

Note de fin 43

# 

# REMERCIEMENTS

Avant tout développement sur cette formation professionnelle, il apparaît opportun de commencer cette documentation par des remerciements, à ceux qui m’ont beaucoup appris au cours de cette formation, et même à ceux qui ont eu la gentillesse de faire de ce cursus un moment très profitable.

Aussi, je remercie Mr Ramdane Bourai, mon enseignant référantqui m’a formé et accompagné tout au long de cette expérience professionnelle avec beaucoup de patience et de pédagogie.

Enfin, je remercie l’ensemble des collaborateurs de *Nouas* : Mr lionel Chanson, Mr François Laurent et Vincent Olivan pour leurs conseils qu’ils ont pu me prodiguer au cours de ces huit mois.

# Résumé du projet en anglais

Food Banks fight against food waste by offering a second life to consumable but non-marketable food.

The three main players in the supply of these banks are the supermarkets, the food-processing industries and producers. Individuals represent only a small percentage of this mass of supply.

This project responds to a problem of centralization of food donations generated by individuals; outside the days of collection; wishing to participate in this commitment...

Indeed, currently we individuals when we wish to give food to collection organizations there is no process to be able to group these various volumes generated . Apart from moving individually to an association there are currently no other alternatives.

This project consists in allowing individuals to post ads describing the nature and quantity of their food donations as well as their locations, but above all a means of communication to organize a short-term volume consolidation system.

In the long term, the many volunteers already working to recover food from the three main actors will be able to graft a fourth actor even more present: individuals and thus fight even more effectively against food waste.

# Tables des figures

[Figure 1 Maquette page d'accueil 11](#_Toc515730081)

[Figure 2 Maquette espace membre 12](#_Toc515730082)

[Figure 3 Maquette formulaire annonce 13](#_Toc515730083)

[Figure 4 Maquette réponses aux annonces 14](#_Toc515730084)

[Figure 5 Modèle Conceptuel de Données 16](#_Toc515730085)

[Figure 6 Modèle Logique de Données 17](#_Toc515730086)

[Figure 7 requête SQL table membres 18](file:///C:\Users\Nouas\Desktop\Dossier%20de%20Projet.docx#_Toc515730087)

[Figure 8 Table membres 19](#_Toc515730088)

[Figure 9 Requête SQL table annonces 20](#_Toc515730089)

[Figure 10 Table membres 20](#_Toc515730090)

[Figure 11 Requête SQL contact 20](#_Toc515730091)

[Figure 12 Table contact 21](#_Toc515730092)

[Figure 13 Requête SQL catégories 21](#_Toc515730093)

[Figure 14 Table catégories 22](#_Toc515730094)

[Figure 15 Requête SQL table région 22](#_Toc515730095)

[Figure 16 Table région 23](#_Toc515730096)

[Figure 17 Requête PDO 24](#_Toc515730097)

[Figure 18 Requête SQL jointure de tables 24](file:///C:\Users\Nouas\Desktop\Dossier%20de%20Projet.docx#_Toc515730098)

[Figure 19 Fulltext 25](#_Toc515730099)

[Figure 20 User story inscription 27](#_Toc515730100)

[Figure 21 User story connexion 28](#_Toc515730101)

[Figure 22 User story post d'annonces 29](#_Toc515730102)

[Figure 23 User story Edition profil 30](#_Toc515730103)

[Figure 24 fichier bootstrap.php 31](#_Toc515730104)

[Figure 25 Fichier contenant les variables d'environnement 32](#_Toc515730105)

[Figure 26 Fichier database.php 32](#_Toc515730106)

[Figure 27 Getter class Database 33](#_Toc515730107)

[Figure 28 class Annonce héritage class Database 34](#_Toc515730108)

[Figure 29 Méthode prepareData() 35](#_Toc515730109)

[Figure 30 Méthode createnewAnnonce() 35](#_Toc515730110)

[Figure 31 Méthode listeAnnonces () 36](#_Toc515730111)

[Figure 32 Méthode UpdateAnnonce () 37](#_Toc515730112)

[Figure 33 méthode uploadFile () 38](#_Toc515730113)

[Figure 34 Méta et Links styles 39](#_Toc515730114)

[Figure 35 Rendu V 1.1 Accueil 40](#_Toc515730115)

[Figure 36 Rendu V1.1 détails annonce 40](#_Toc515730116)

[Figure 37 Rendu V1.1 formulaire prise de contact 41](#_Toc515730117)

[Figure 38 Rendu V1.1 mise en contact 41](#_Toc515730118)

[Figure 39 Rendu V1.1 espace membre 42](#_Toc515730119)

[Figure 40 Back-log pour V1.2 43](#_Toc515730120)

# Liste des compétences couvertes par le projet

1. Développer une application client-serveur
   * Maquetter une application
   * Concevoir une base de données
   * Mettre en place une base de données
   * Développer une interface utilisateur
   * Développer des composants d'accès aux données

2. Développer une application web

* + Développer des pages web en lien avec une base de données
  + Mettre en œuvre une solution de gestion de contenu ou e-commerce
  + Développer une application simple de mobilité numérique
  + Utiliser l’anglais dans son activité professionnelle en informatique

# Cahier des charges

# Introduction

# Contexte

La réalisation de ce projet est effectuée dans le cadre d’une formation pour obtenir un « Titre professionnel « Développeur logiciel ».



Cette formation est effectué avec Nouas, organisme de Conseil, formation coopération internationale intervenant en France en Allemagne et sur l’ensemble du bassin méditerranéen. Nouas construit, propose et anime des formations sur mesure, labellisée « Grande Ecole du Numérique ».

Les Banques Alimentaires luttent contre le gaspillage alimentaire en offrant une seconde vie à des denrées consommables mais non commercialisables.

Les trois principaux acteurs d’approvisionnement de ces banques sont la grande distribution, les industries agro-alimentaires et les producteurs. Les particuliers ne représentent qu’un pourcentage minime de cette masse d’approvisionnement.

Ce projet répond à une problématique de centralisation des dons alimentaires générés par les particuliers ; en dehors des journées de collecte ; souhaitant participer à cet engagement.

En effet, actuellement nous particuliers lorsque nous désirons donner de l’alimentation à des organismes de collectes il n’existe aucun procédé pour pouvoir regrouper ces différents volumes générés. Hormis le fait de se déplacer individuellement en direction d’une association il n’y a pas d’autres alternatives actuellement.

Ainsi ce projet consiste à permettre aux particuliers de poster des annonces décrivant la nature, la quantité de leurs dons alimentaires ainsi que leurs localisations mais surtout un moyen de communication pour organiser un système de regroupement de volumes à court terme.

Sur le long terme les nombreux bénévoles œuvrant déjà pour récupérer les denrées issus des trois principaux acteurs pourront greffer un quatrième acteur encore plus présent : les particuliers et ainsi lutter encore plus efficacement contre le gaspillage alimentaire.

# Description de la demande

# Les objectifs

L’objectif du projet est d’organiser la récupération des dons alimentaires, en effet le post d’annonces ciblées par catégories et géo-localisation permettront d’améliorer la logistique au sein des particuliers. Ainsi une collecte générée par les dons de particuliers pourra se greffer sur le circuit utilisé par les professionnels.

# Les fonctions

L’interface graphique doit posséder un système de création et de gestion d’annonces par l’utilisateur. Un formulaire de recherche d’annonces pouvant cibler par différent critères entré par l’utilisateur.

Création d’un espace membre avec prise de contact entre membres pour une organisation de la logistique.

Le projet doit inclure une adaptation au différent format d’affichage (écran ordinateur, tablette, Smartphone …).

# Contraintes

La réalisation de ce projet doit être faite dans un délai équivalent à la durée de la formation.

La nécessité d’assimiler de nouvelles technologies pour rendre fonctionnelles des spécifications demandées.

# Solution proposée

# Interface graphique

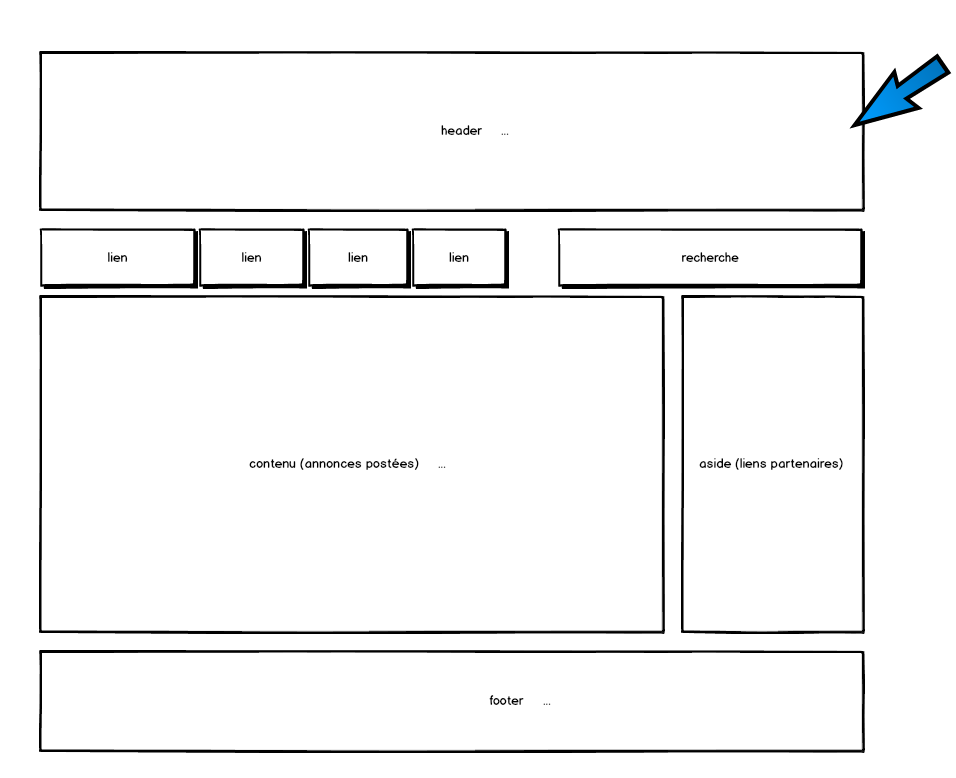


Figure Maquette page d'accueil

L’interface utilisateurs affiche les dernières annonces postées en format miniature cliquable pour voir l’annonce en détail et une éventuelle prise de contact.

Une barre de navigation permettant de se déplacer au sein du site et une barre de recherche intégrés pour filtrer les annonces selon les besoins de l’utilisateur.

Une liste de liens redirigeant vers les divers partenaires.

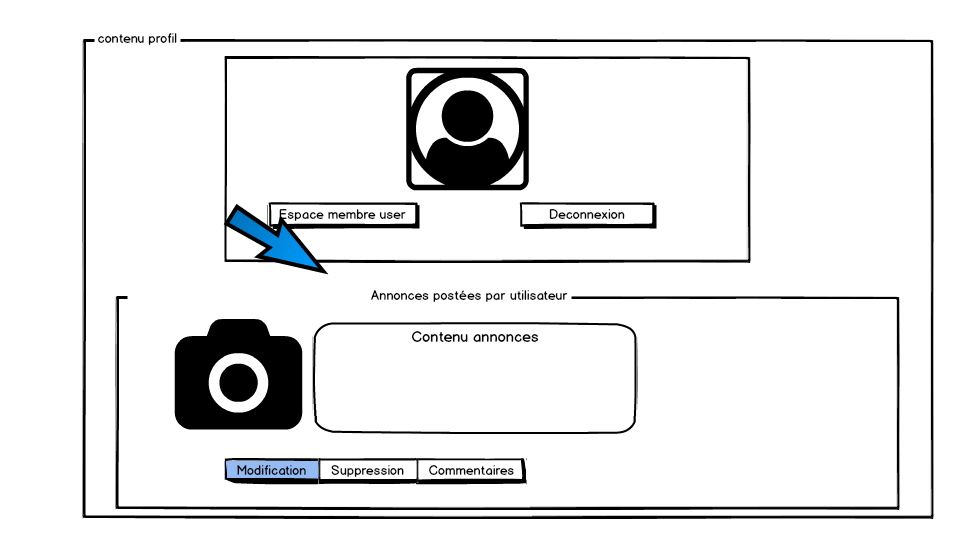


Figure Maquette espace membre

L’espace membre de l’utilisateur permet de visualiser son profil ainsi que la possibilité d’éditer les différentes informations liées au compte.

Ensuite s’affiche les différentes annonces postées par l’utilisateur classées par ordre chronologique .

Chaque annonces contiennent trois boutons actifs qui offrent la possibilité de :

-modifié les informations de l’annonce sélectionnée.

-supprimé l’annonce sélectionnée.

- et enfin ; grâce à un tag apparaissant sur le bouton ‘réponse’ ; l’utilisateur est informé du nombre de réponses qu’il a généré et peux les visualiser, pour en amont contacter les personnes intéressés par son annonce.

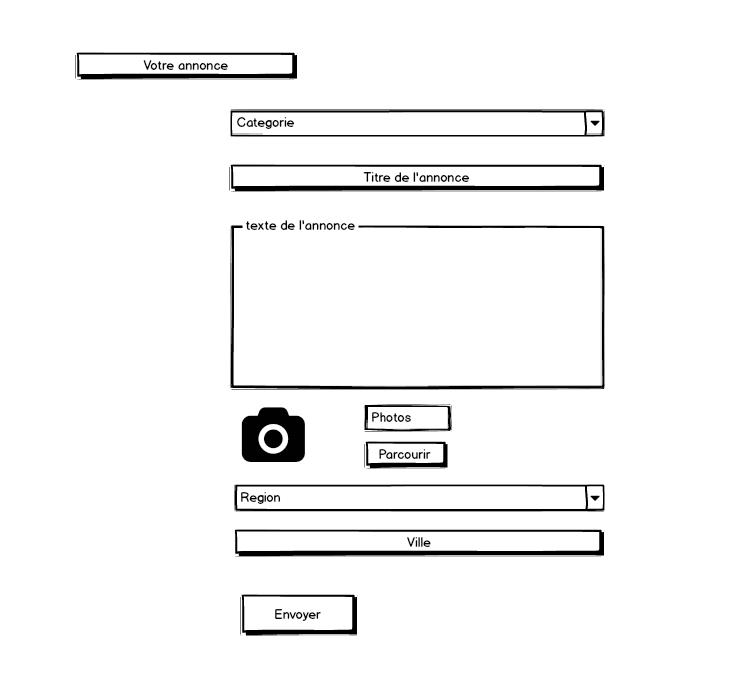


Figure Maquette formulaire annonce

L’interface pour poster des annonces se compose de six champs classés dans cet ordre :

* La catégorie du produit (ex : produit sucré, salé etc ….).
* Le titre de l’annonce.
* Le texte de l’annonce.
* La possibilité d’insérer une photo du produit (non obligatoire car photo par défaut existante).
* La région ou se situe l’utilisateur.
* Et enfin la ville de l’utilisateur.

Tous ces champs permettent d’avoir un descriptif complet du produit proposé ainsi qu’une géo-localisation précise. Cela permet aussi d’indexer toutes ces informations dans la Base De Donnée pour rendre la barre de recherche ; accessible tous le long de la navigation ; la plus performante possible.

En effet, au sein de la barre de recherche il est possible pour les utilisateurs de rechercher par catégorie de produit, titre, contenu ou encore par lieux (ville, région ….).

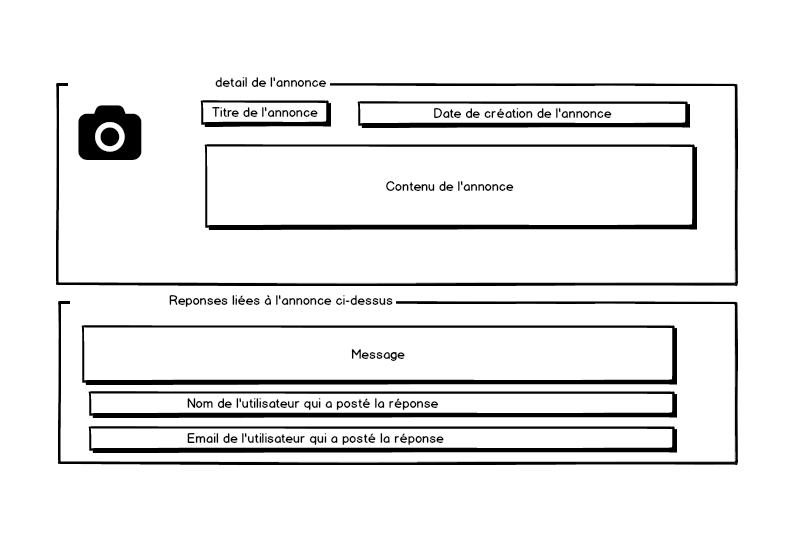


Figure Maquette réponses aux annonces

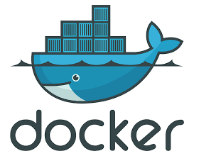
L’interface qui affiche le détail des annonces permet de visionner son annonce postée en vue de la modifier si elle ne correspond pas à l’attente de l’utilisateur.

Ensuite ; si réponses il y a ; un descriptif des réponses liées à l’annonce apparait et permet de constater quel utilisateur est intéressé et des informations de contact sont affichées pour pouvoir mettre les différents acteurs en relation.

## Technologies utilisées

# Modèle Conceptuel de Données

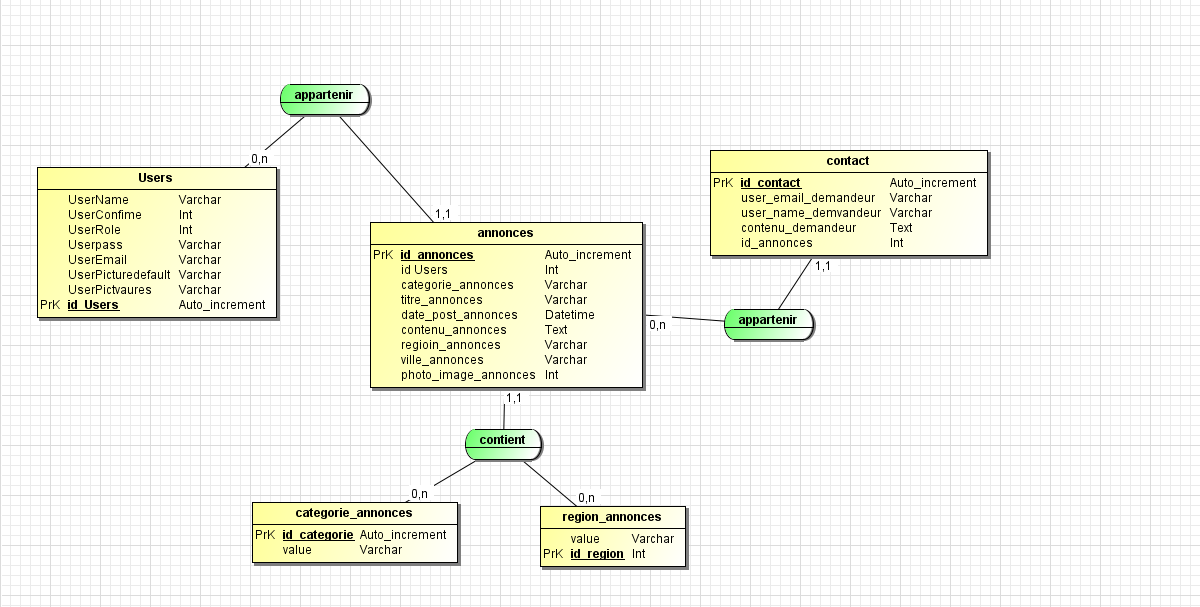


Figure Modèle Conceptuel de Données

# Modèle Logique de Données

# BDD_MCD.PNG

Figure Modèle Logique de Données

# Persistance des données

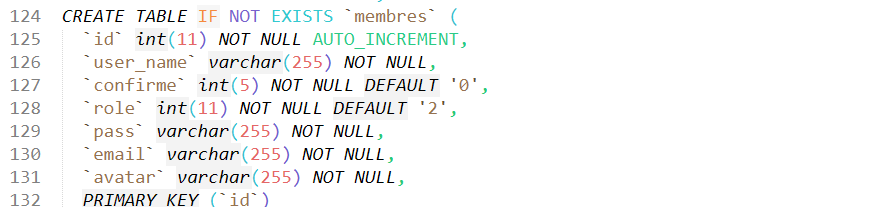
# Dictionnaire des données

* Table membres

La table membres se constitue de sept champs :

Figure requête SQL table membres

* Un champ ‘id’ contenant la clé primaire de type INT auto-incrémenté.
* Un champ ‘rôle’ de type INT nécessaire pour définir les profils utilisateurs (administrateur, modérateurs ou utilisateurs).
* Un champ ‘confirme’ de type INT nécessaire pour confirmer un membre au sein du site.
* Un champ ‘user\_name’de type VARCHAR (255).
* Un champ ’email’ de type VARCHAR (255).
* Un champ ‘pass’de type VARCHAR (255) qui en amont a été haché en utilisant password\_hash() afin de ne pas stockée le mot de passe utilisateur en clair.
* Un champ ‘avatar’ de type VARCHAR(255) utile lors de la récupération de l’image uploadée par l’utilisateur.



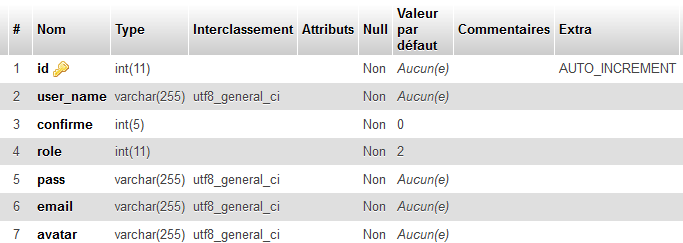


Figure Table membres

* Table annonces

La table annonces se constitue de neuf champs :

* Un champ ‘id’ contenant la clé primaire de type INT auto-incrémenté.
* Un champ ‘membre\_id’ de type INT contenant la clé étrangère nécessaire à la jointure des tables « annonces et membres ».
* Un champ ‘categorie’ de type VARCHAR (255) contenant des entrées provenant de la table « categories ».
* Un champ ‘titre’ type VARCHAR (255).
* Un champ ’date\_post\_message’ de type DATETIME.
* Un champ ‘contenu’de type TEXT.
* Un champ ‘region’ de type VARCHAR(255) contenant des entrées provenant de la table « region ».
* Un champ ‘ville’ de type VARCHAR(255) contenant des entrées provenant de la table « region ».
* Un champ ‘photo’ de type VARCHAR(255) utile lors de la récuperaton de l’image uploadée par l’utilisateur.

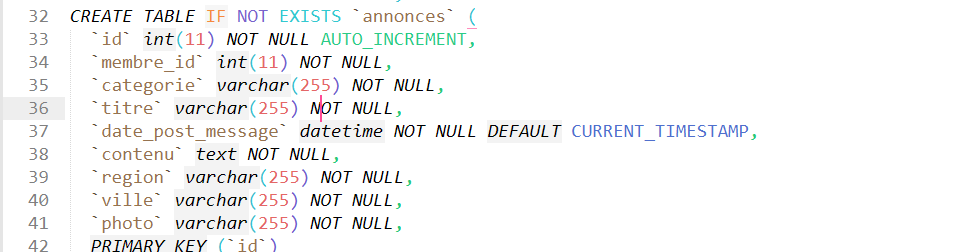


Figure Requête SQL table annonces

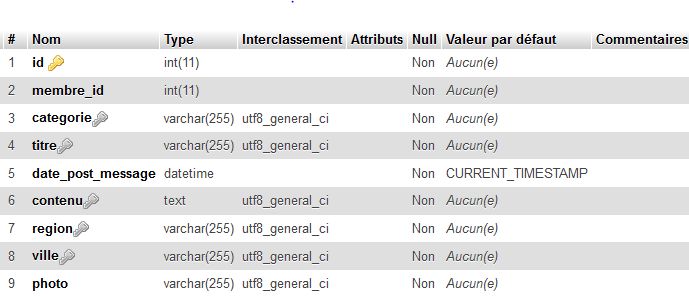


Figure Table membres

* Table contact

La table contact se constitue de cinq champs :

* Un champ ‘id’ contenant la clé primaire de type INT auto-incrémenté.
* Un champ ‘id\_annonces’de type INT contenant la clé étrangère. nécessaire à la jointure des tables « annonces et contacts ».
* Un champ ‘user\_name’ de type VARCHAR (50).
* Un champ ‘email’de type VARCHAR (50).
* Un champ ‘reponse’de type MEDIUM TEXT .

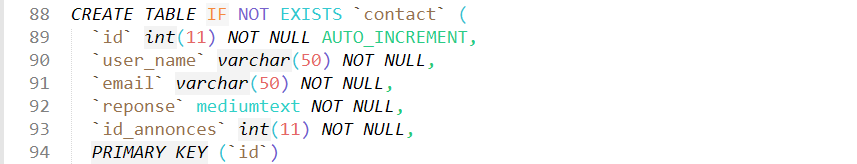


Figure Requête SQL contact

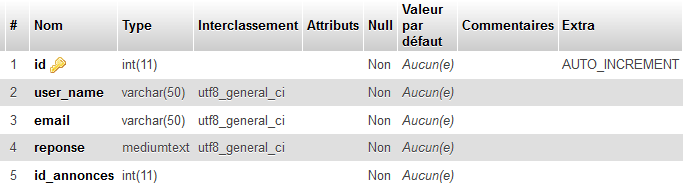


Figure Table contact

* Table catégories

La table catégories se constitue de deux champs :

* Un champ ‘id’ contenant la clé primaire de type INT auto-incrémenté.
* Un champ ‘value’de type VARCHAR (50).

L’utilisation de cette table est nécessaire pour générer un select présent dans le formulaire de post d’annonces.

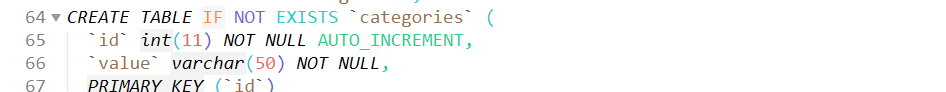


Figure Requête SQL catégories

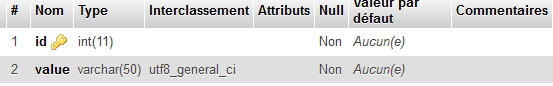


Figure Table catégories

* Table région

La table région se constitue de deux champs :

* Un champ ‘id’ contenant la clé primaire de type INT auto-incrémenté.
* Un champ ‘value’de type VARCHAR (50).

L’utilisation de cette table est nécessaire pour générer un select présent dans le formulaire de post d’annonces.

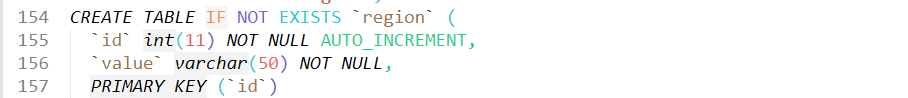


Figure Requête SQL table région

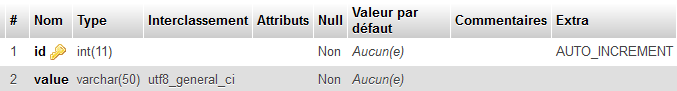


Figure Table région

# Exemples de requêtes SQL

## Exemple 1 :

La base de données est accessible en utilisant PDO. PDO est un pilote qui implémente l'interface de PHP Data Objects (PDO) pour autoriser l'accès de PHP aux bases de données de MySQL. L’utilisation de try et catch permets de lever une erreur et de l’attraper pour l’affichage, cependant une condition a été ajoutée en fonction de l’environnement où se trouve le projet. En effet si la variable d’environnement [‘DEBUG’] a été initialisée à TRUE cela signifie que le projet n’est pas en production et donc on peut afficher les erreurs.

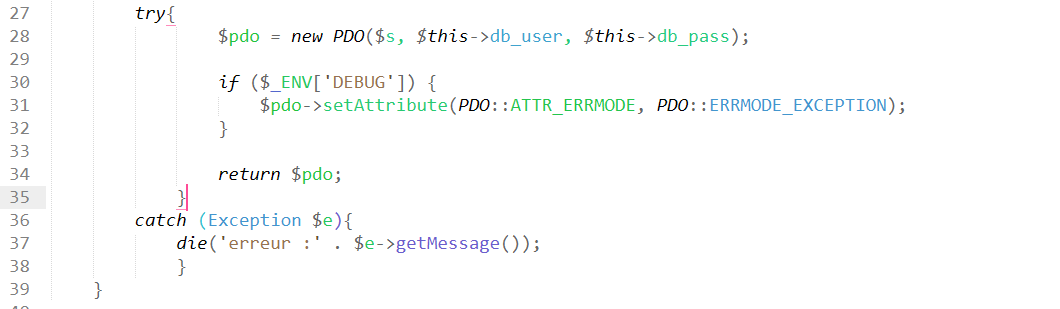


Figure Requête PDO

Plus en détail chapitre *Réalisation/fichier databse.*php (pages 31, 32*).*

## 

## Exemple 2 :

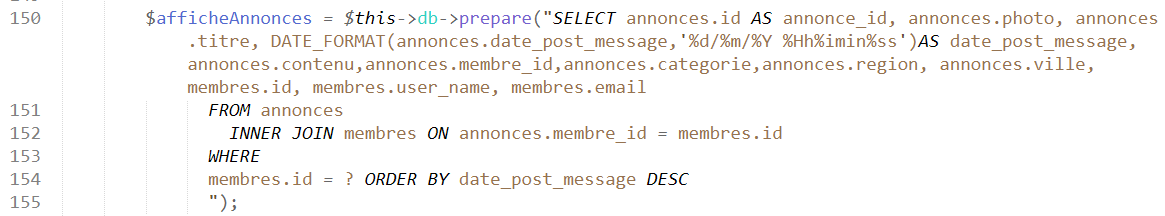
Cette requête préparée attend comme clé étrangère « membres.id » qui doit correspondre à « annonces\_membres\_id ». Elle est nécessaire lors de l’affichage d’un POST détaillé ainsi qu’un complément d’information sur le créateur.

Ensuite un execute(array($membre\_id) est appliqué pour comparer la valeur du paramètre d’entrée avec la valeur de « membres.id » reçu lors de la requête.

Pour l’affichage un FETCH ALL() sera exécuté et stockée dans une variable .

Un allias « «AS » utilisé sur deux parametres (‘annonce.id et annonce.date\_post\_message’) permets d’avoir une meilleure lisibilité. Un travail de formatage a aussi été effectué pour un affichage type européens.

Figure Requête SQL jointure de tables



## Exemple 3 :

Un index FULLTEXT est utilisé pour faire des recherches de manière puissante et rapide sur un texte dont le type est CHAR, VARCHAR ou TEXT.

Une requête SQL a été exécuté dans PHP MY ADMIN permettant d’indexer les champs « categorie, titre, contenu, region et ville » pour améliorer la recherche au sein du contenu des multiples champs et entrées.

Dans le *SELECT* on récupère tous les champs des tables « membres » et « annonces » si la clé étrangère est semblable mais aussi chaque champ qui "MATCH" avec le mot-clé contenu dans une variable protégée par « htmlspecialchars ». Ainsi chaque entrée de chaque champ où il y a *match* sera retournée à l’affichage.

La barre de recherche étant présente sur l’ensemble du site via la « NAVBAR » ,il est possible de formuler une recherche à n’importe quel moment de la navigation.

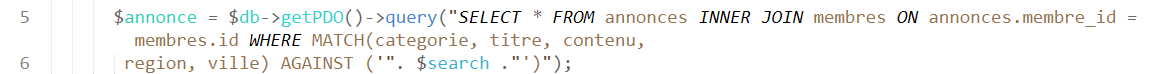


Figure Fulltext

# Spécifications fonctionnelles

## Profil utilisateur

L’utilisateur peut créer un compte utilisateurs en entrant son nom d’utilisateur, son email, son mot de passe personnel. Ensuite il peut accéder à un espace membre personnel où il lui est possible d’éditer ses données personnelles saisies lors de l’inscription et aussi d’ajouter un avatar.

Ensuite il peut poster une annonce via un formulaire où il sera nécessaire d’entrer plusieurs informations indispensables à l’insertion dans la base de donnée et pour l’indexation utile à la recherche. Une fois cela réalisé l’annonce apparaitra au sein de son espace membre lui offrant ainsi la possibilité de modifier cette même annonce ou encore la supprimée si elle n’a plus lieu d’être. Enfin un système de tag permet à l’utilisateur d’être informé de la présence de réponses liés à ses annonces. En cliquant sur un lien « Réponses » un tableau lui est présenté contenant les réponses d’autres utilisateurs ainsi que leurs coordonnées respectives.

Les annonces générées apparaitront sur la page d’accueil sous la forme de miniatures ordonnées par ordre décroissant dans le temps. Sur les miniatures un lien est accessible pour afficher les annonces plus en détails mais aussi un formulaire de mise en contact entres utilisateurs.

Au sein de la barre de navigation un champ de recherche permet à l’utilisateur de trouver des annonces selon de nombreux critères tel que a nature du produit ou encore sa géo-localisation.

Pour conclure en bas de page de nombreux liens sont disponibles tel que les différents partenaires, la prise de contact avec l’administrateur du site ou encore les Conditions Générales d’Utilisations des données personnelles.

# Profil administrateur

L’administrateur possède les mêmes fonctions que l’utilisateur. Cependant en accédant à l’espace Admin il peut valider un membre via un lien agissant directement dans la base de données par le biais d’un GET. Il peut aussi supprimer un membre, dans le cas où ce membre ne respectait pas les règles du site.

# Spécifications techniques

## Cas d’utilisation (Users story mapping)

* Inscription d’un utilisateur

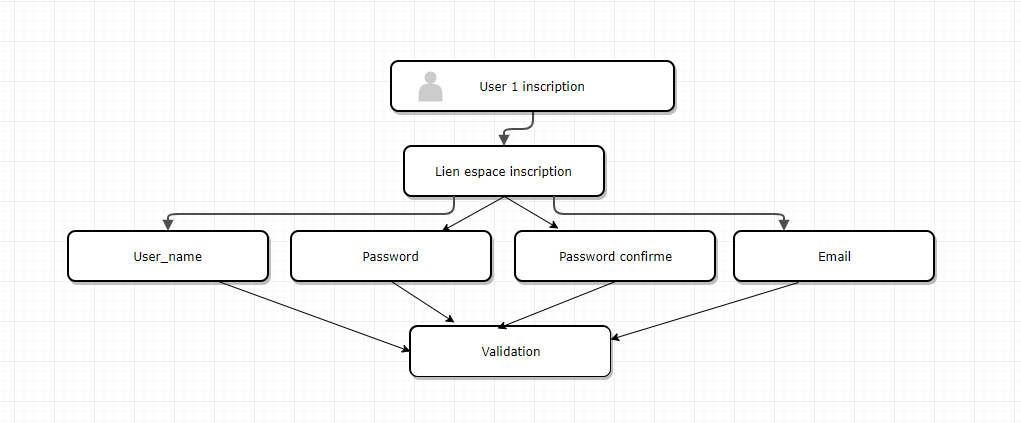


Figure User story inscription

* Connexion d’un utilisateur

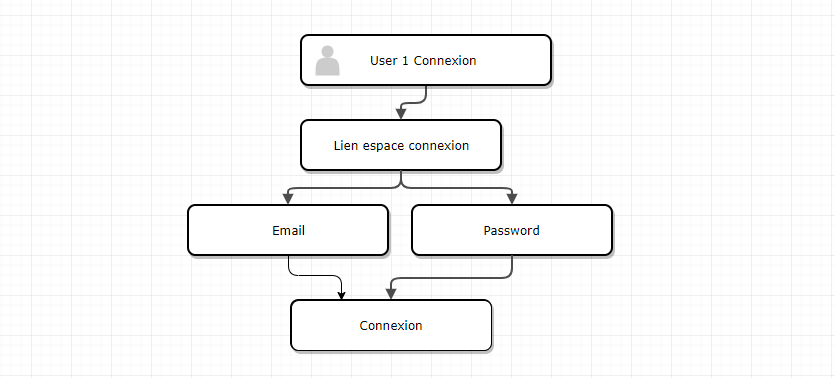


Figure User story connexion

* Post d’une annonce par un utilisateur

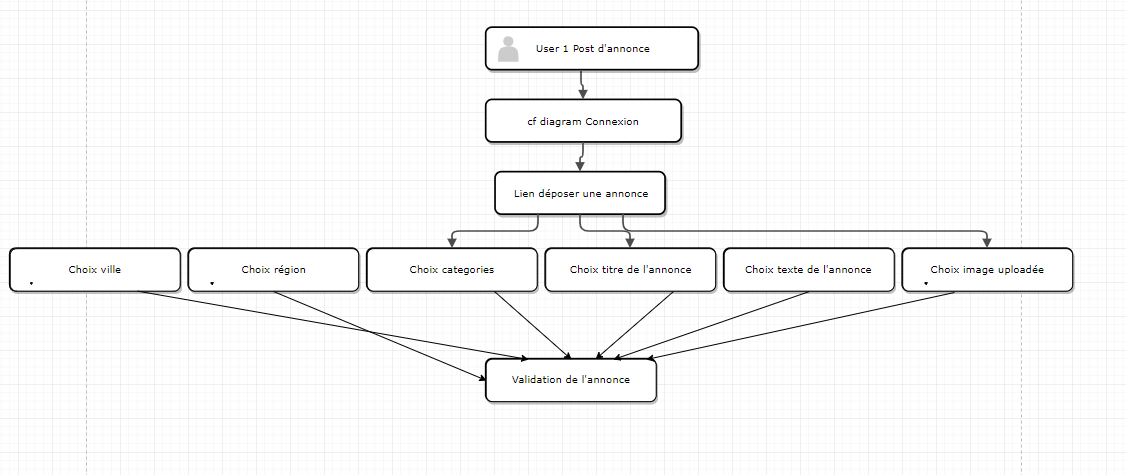


Figure User story post d'annonces

* Edition du profil utilisateur

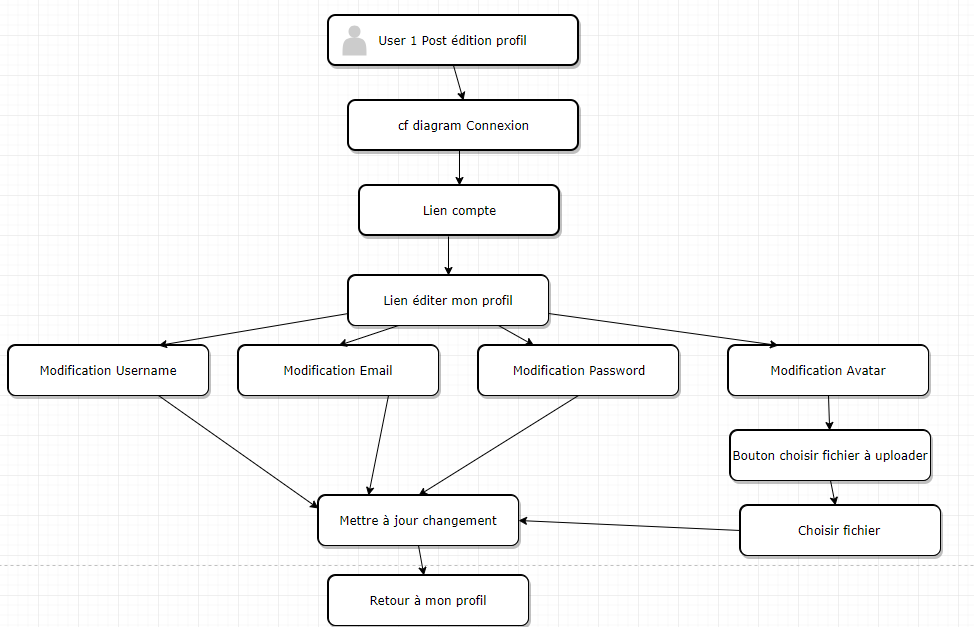


Figure User story Edition profil

# Réalisation

## Architecture

En début de formation ayant appris seulement les bases des langages de programmation j’ai commencé le présent projet sans adopter réellement de Design Pattern.

Ensuite ayant pris connaissance des Design Pattern je me suis tourné vers une architecture de style MVC avec une vue, un contrôleur et un modèle malheureusement par manque de temps ce n’est pas encore tout a fait optimal.

### Fichier bootstrap.php

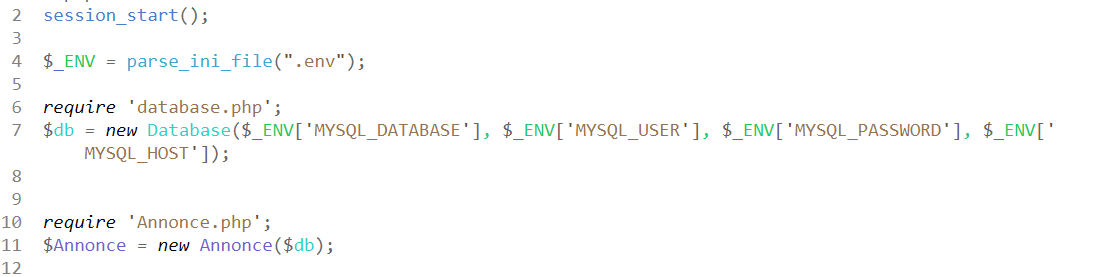


Figure fichier bootstrap.php

Le fichier « bootstrap.php » est le premier fichier chargé par le site, il sert de contrôleur entre le modèle et la vue.

Lors de son exécution il y a une session START () qui démarre une session pour ultérieurement stocké les données utilisateurs générées lors d’une inscription ou d’une connexion.

Ensuite une variable $\_ENV récupère les variables d’environnement via la commande parse\_ini\_file() stockées dans un fichier .env.

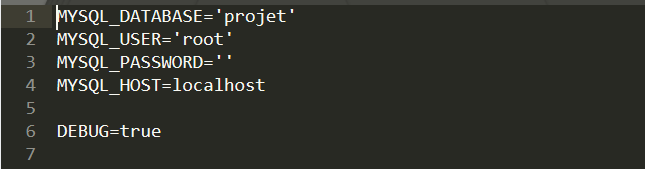
* 

Figure Fichier contenant les variables d'environnement

Ce fichier contenant des données sensibles n’est jamais versionné (Git Hub, Git Lab) en utilisant un fichier .gitignore qui créer des exclusions lors de push Git.

Pour pouvoir alors utiliser le présent projet en production il est alors nécessaire de créer un fichier environnement vierge qu’il faudra remplir selon l’environnement où il sera.

Ensuite instanciation des deux class, en premier « Database » avec en paramètres utilisation des variables d’environnement et en deuxième « Annonce » pour générer l’affichage des miniatures présentes sur la page d’accueil avec en paramètre l’instance de « Database ».

### La class Database

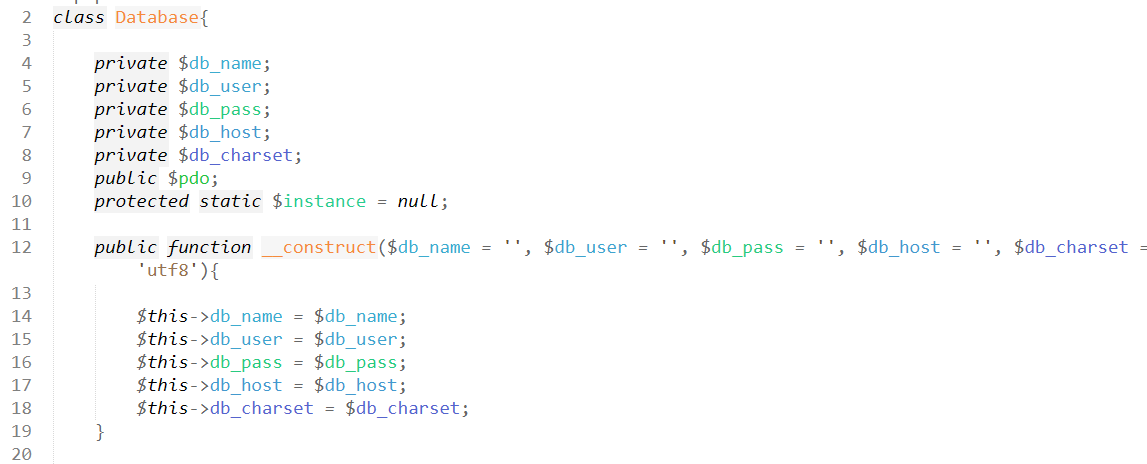


Figure Fichier database.php

La class « Database » permets de déterminer les paramètres de connexion à MySQL.

Les propriétés sont mises en private lors de leurs déclarations pour que la portée ne soit que dans la class elle-même.

Ensuite, la fonction constructeur se charge de définir la valeur à stocker dans

* $db\_name
* $db\_user
* $db\_pass
* $db\_host
* Et $db\_charset

Dès que le constructeur va être exécuté, les propriétés seront stockées et cette fonction va nous renvoyer le contenu des propriétés qui était déclarées en private.

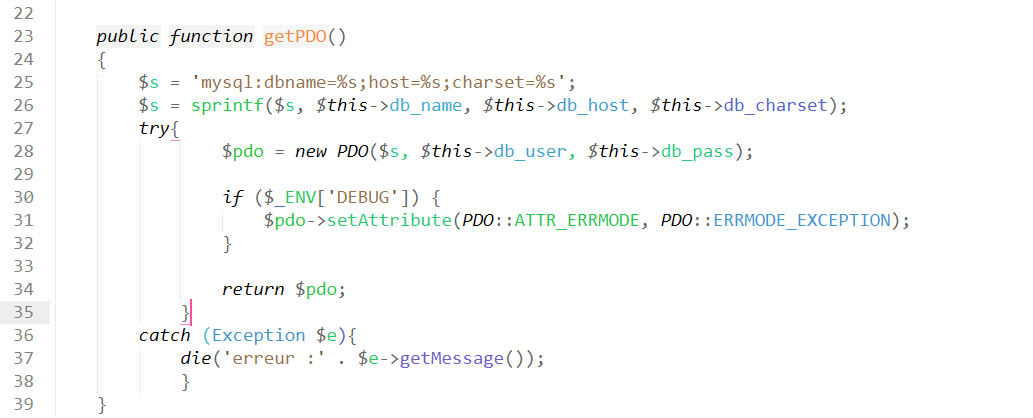


Figure Getter class Database

Finalement, un getter nommé getPDO() est créer pour pouvoir récupérer les propriétés de l’objet hors de la class « Database » . La fonction getPDO() est mise en public pour pouvoir y accéder hors de la class  « Database » , à l’intérieur une variable $s stocke une chaine de caractères en premier lieu. Ensuite application de la fonction sprintf() qui permet de retourner une chaîne formatée . Le fait d’utiliser cette fonction permet de gagner en lisibilité par rapport à une concaténation. On stocke l’instance de la class PDO dans une variable $pdo, c’est cette même variable qui sera retourné par la méthode.

Une condition est présente pour vérifier si le projet est en production ou en local via une variable d’environnement citée plus haut.

En effet si la variable d’environnement [‘DEBUG’] a été initialisée à TRUE cela signifie que le projet n’est pas en production et donc on peut afficher les erreurs.

### La class Annonce

La class « Annonce » est utile pour le CRUD (Create, Read, Update, Delete) concernant toutes les annonces postées par les utilisateurs.

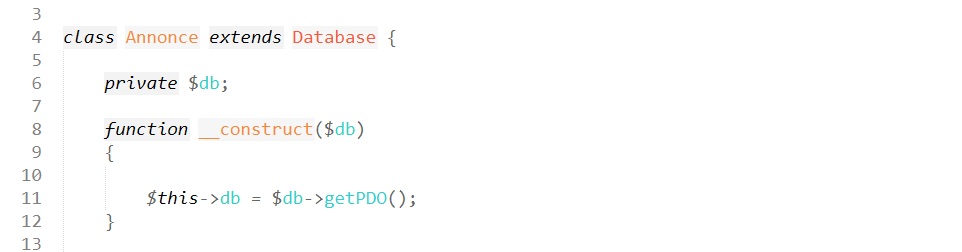


Figure class Annonce héritage class Database

La class « Annonce » hérite de la classe parent « Database » lui permettent ainsi de se connecter à la Base de Données. Une propriété passer en private est alors crée $db ensuite via la méthode \_\_construct($db) qui attend en paramètre la propriété $db. Ensuite à l’intérieur de la méthode on n’attribue à la propriété $db l’instance de la class « Database » qui appelle la méthode getPDO().



Figure Méthode prepareData()

Ensuite la première méthode nommée prepareData($request) qui attend comme paramètres $request représentant les POST issus du formulaire d’annonces. Pour éviter que les entrées en Base de Donnée ne contiennent du code HTML il est nécessaire d’utiliser la fonction htmlspecialchars (). Enfin la méthode retournera $request pour des utilisations ultérieures.



Figure Méthode createnewAnnonce()

La méthode createNewAnnonce($request) attend en paramétres $request . Une première propriété est alors créer $membre\_id et se voit attribuer comme valeur l’ID du membre désirant créer l’annonce via la super global $\_SESSION [‘membre\_id].

La deuxième propriété créée $request contiendra comme valeur l’ensemble des valeurs contenu dans $request au sein de la méthode $prepareData ().

Une condition est mise en place pour tester que l’ensemble des champs ont bien été remplie , si tel n’est pas le cas alors un message d’erreur est retourné via la super globale $\_SESSION[’erreur’] et la valeur false .

Si la condition est vraie alors une requête SQL insère chaque entrée dans les champs correspondant dans la table annonces et retourne la dernière ID générée pour le traitement de l’image uploadée.

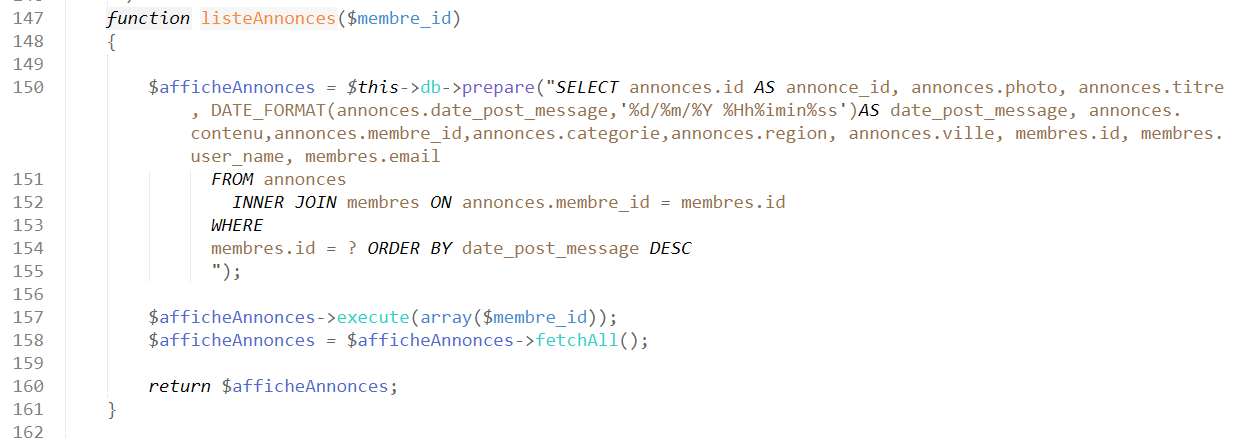


Figure Méthode listeAnnonces ()

La méthode listeAnnonces ($membre\_id) est nécessaire à l’affichage des annonces postées par l’utilisateur. Elle attend comme paramètre la clé étrangère $membre\_id nécessaire à la jointure entre les tables « annonces et membres ».

Une requête SQL préparée est alors exécutée pour pouvoir afficher l’ensemble des annonces via un fetchAll () classée par date de post. Enfin la méthode retourne la propriété $afficheAnnonces contenant le résultat du fetchAll ().



Figure Méthode UpdateAnnonce ()

La méthode updateAnnonce($request) est nécessaire à la modification des annonces postées par l’utilisateurs. En effet au sein de son espace membre l’utilisateur a la possibilité de modifier l’ensemble de ses annonces.

Cette méthode est très proche du mode de fonctionnement de la méthode createNewAnnonce($request) . Elle attend en paramètre $request et les mêmes propriétés sont créée ainsi que les valeurs retournées sont identiques. La différence majeure se trouve dans la requête qui est un UPDATE au lieu d’un INSERT.



Figure méthode uploadFile ()

La méthode uploadFile ($files, $last\_annonce\_id) est nécessaire pour le traitement des images uploader par l’utilisateur. Elle attend deux paramètres :

* $files qui est un tableau bidimensionnel contenant toutes les caractéristiques des images uploadées tel que le nom, la taille, l’extension etc …
* $last\_annonce\_id qui est la dernière ID générée lors de l’insertion d’une annonce sur la base de Donnée. Cette ID est issus des retours des méthodes updateAnnonce($request) et createNewAnnonce($request) et est nécessaire pour l’attribution du noms des images.

La propriété $tailleMax est utile pour déterminer une taille maximale lors d’une condition. La propriété $extansionValides est utile pour déterminer les extensions valides lors de l’upload d’images. Un traitement est effectué sur les extensions afin qu’elles soient en minuscules. Une propriété $chemin est créée pour définir le chemin. Enfin le nom de l’image ainsi traitée sera déterminé par la valeur de $last\_annonce\_id , d’un point et de l’extension contenu dans $extensionUpload.

## Interface graphique

L’utilisation du style de BOOTSTRAP permis un gain de temps non négligeable mais surtout la possibilité d’avoir un design facilement modifiable et responsive nativement. Cependant pour pouvoir personnalisé le site il a été nécessaire de créer mes propres feuilles de style CSS et prendre le dessus sur les feuilles de style de BOOTSTRAP.

Voici les META et les LINKS nécessaire au design du site.



Figure Méta et Links styles

* La ligne 7 est nécessaire à l’affichage sur un support de taille smartphone , elle permet aussi de zoomer .
* La ligne 8 est une autre version du site comprenant une utilisation du framework BULMA.
* La ligne 9 est la feuille de style nécessaire pour prendre l’ascendant sur les feuilles de style BOOTSTRAP.
* Les lignes 10 et 11 utilisent un CDN pour pouvoir charger les styles de BOOTSTRAP.
* Les lignes 13, 14 et 15 sont nécessaires au traitement coté client de nombreux effets de style et utilisent les technologies JAVASCRIPT et JQUERY.
* La dernière ligne quand à elle est utile pour charger des icones présentent dans le site via FONT AWESOME .COM.

## Rendu final version v1.1

## Accueil

## Capture18.PNG

Figure Rendu V 1.1 Accueil

## Détails de l’annonce



Figure Rendu V1.1 détails annonce

## Formulaire de mise en contact

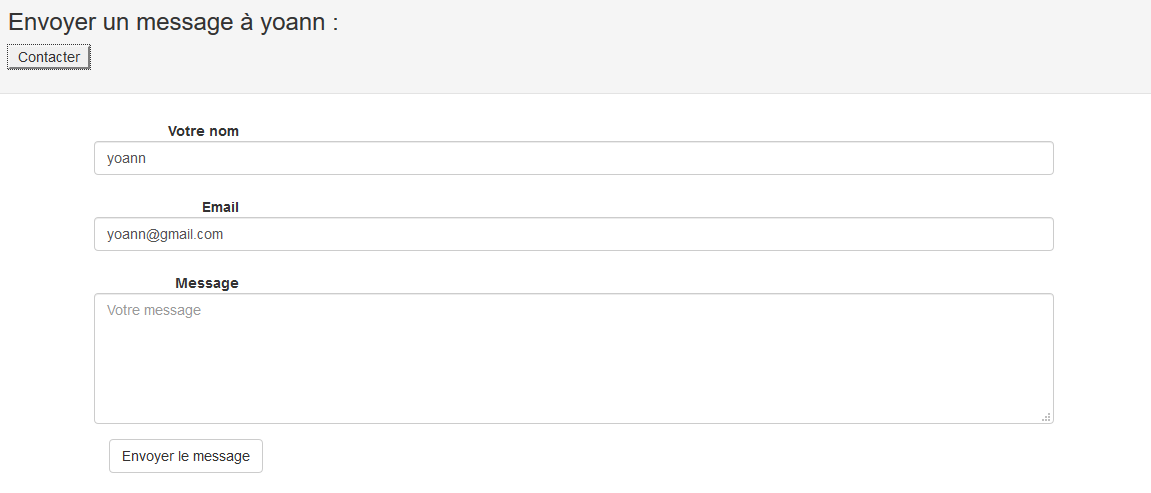


Figure Rendu V1.1 formulaire prise de contact

## Mise en contact

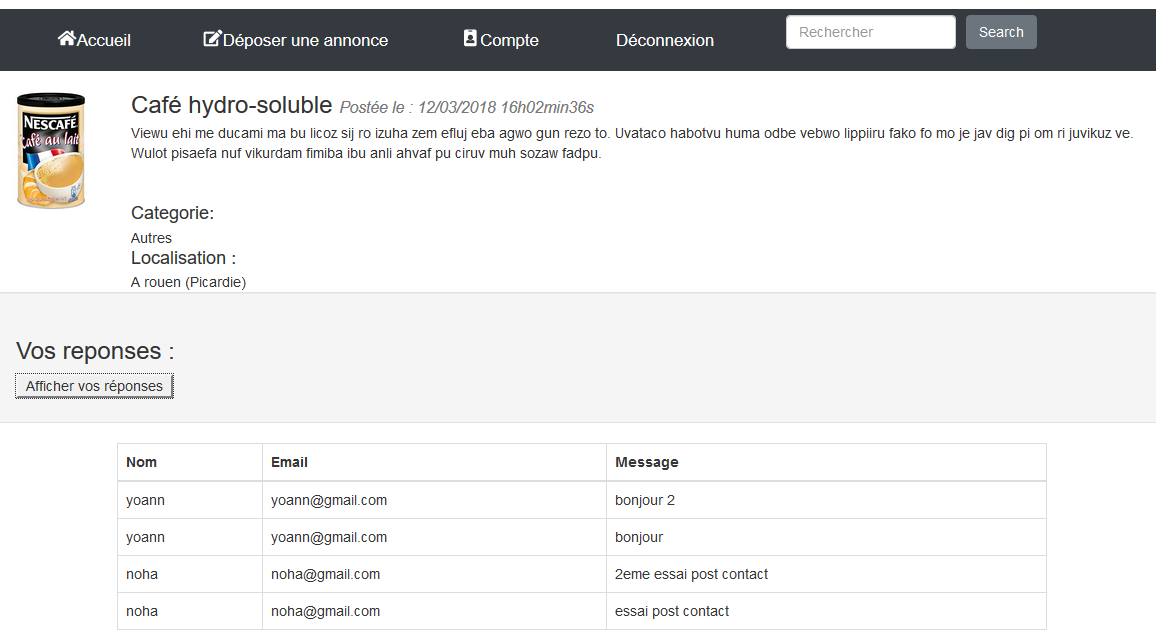


Figure Rendu V1.1 mise en contact

## Mise en contact

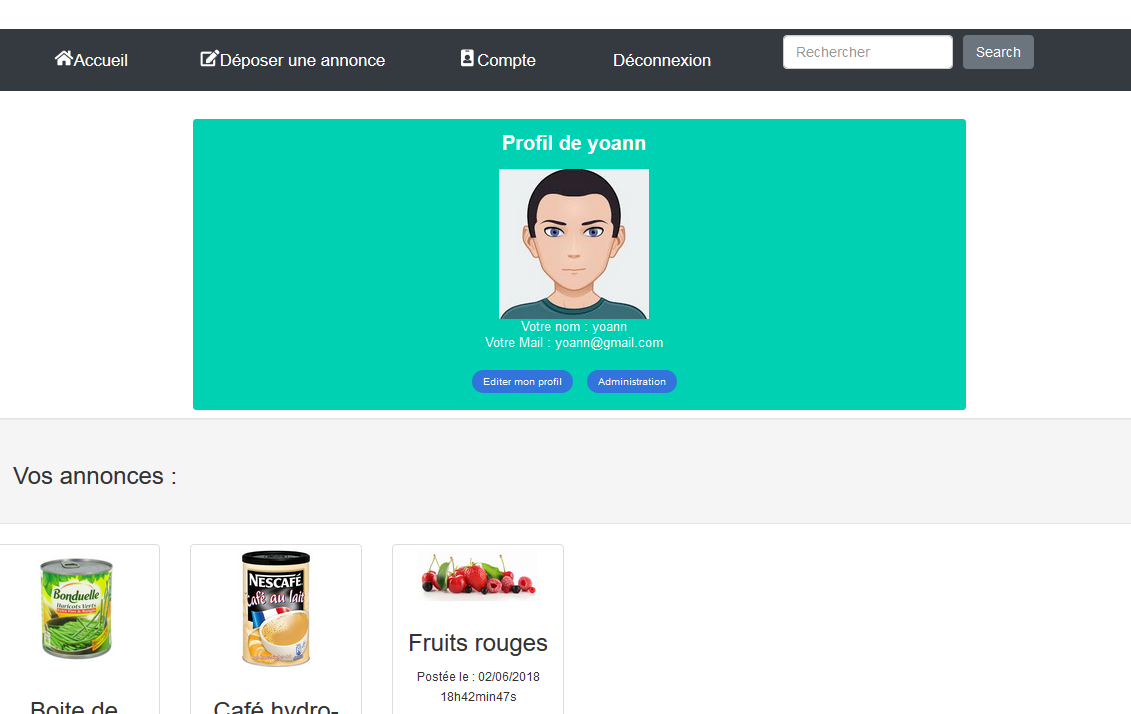


Figure Rendu V1.1 espace membre

# Gestion de projet

Le présent projet est à la version V1.1 dans une vision long terme le design sera retravaillé et de nombreuses fonctionnalités seront ajoutées. Un aperçu du BACK-LOG permet de voir les taches restantes à réaliser pour la version V.1.2

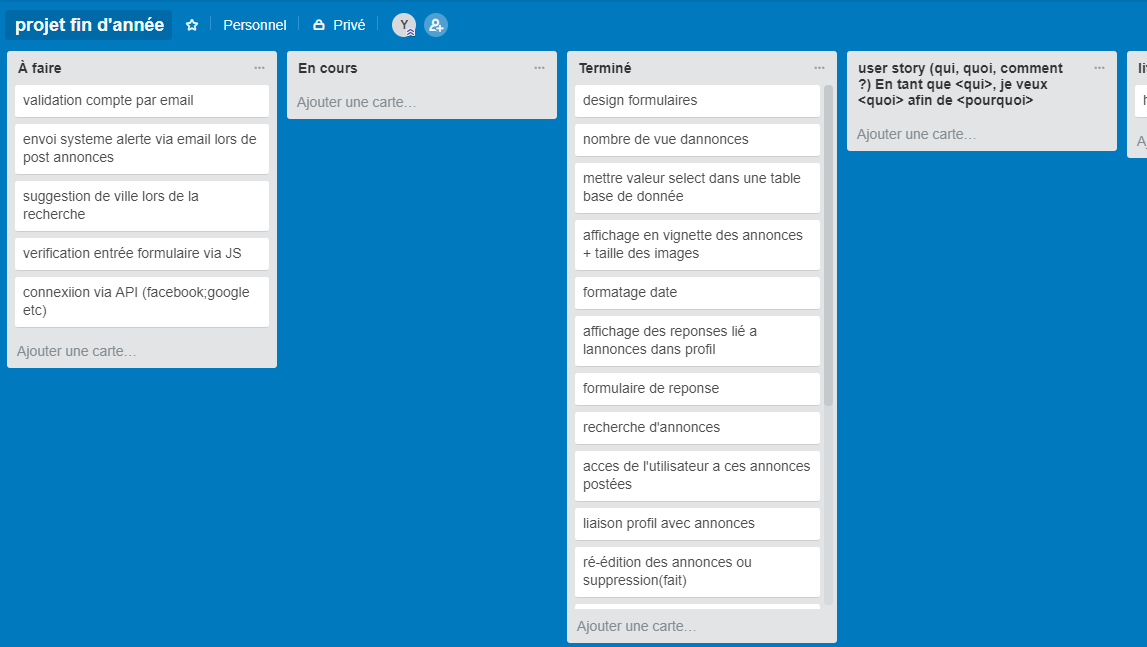


Figure Back-log pour V1.2

# Note de fin

Ce document a été rédigé dans le cadre d’une formation pour l’obtention du titre professionnel « Développeur logiciel ».

J’espère avoir été clair dans les explications et la rédaction de ce document et aussi vous avoir transmis le plaisir que j’ai eu à concevoir ce projet. Cordialement.

Yoann MELLIER