Documentation technique

Seconde Main

Ackermann Gawen

23.05.2019 v1.0

[1. Introduction 5](#_Toc9520099)

[2. Contenu 5](#_Toc9520100)

[2.1Sujet 5](#_Toc9520101)

[2.2 But 5](#_Toc9520102)

[2.3 Spécifications 5](#_Toc9520103)

[2.4 Environnement de développement 5](#_Toc9520104)

[2.5 Livrable 5](#_Toc9520105)

[3. Analyse fonctionnelle 6](#_Toc9520106)

[3.0 Les différents profils utilisateur 6](#_Toc9520107)

[3.0Bis Les différents plan de navigation 7](#_Toc9520108)

[3.0Bis.1 Plan de navigation d’un visiteur 7](#_Toc9520109)

[3.0Bis.2 Plan de navigation d’un utilisateur connecté 8](#_Toc9520110)

[3.0Bis.3 Plan de navigation d’un administrateur 9](#_Toc9520111)

[3.1 Navbar 10](#_Toc9520112)

[3.1.1 Visiteur 10](#_Toc9520113)

[3.1.2 Utilisateur authentifié 10](#_Toc9520114)

[3.1.3 Administrateur 10](#_Toc9520115)

[3.2 Accueil 11](#_Toc9520116)

[3.2.1 Système de filtrage 11](#_Toc9520117)

[3.2.2 Description d’un « bloc » d’annonce 12](#_Toc9520118)

[3.3 Détails d’une annonce 13](#_Toc9520119)

[3.3.1 Acheter un article 14](#_Toc9520120)

[3.4 Connexion 15](#_Toc9520121)

[3.5 Inscription 16](#_Toc9520122)

[3.6 Création d’une annonce 17](#_Toc9520123)

[3.7 Mes annonces 18](#_Toc9520124)

[3.8 Modification d’une annonce 19](#_Toc9520125)

[3.9 Gestion des annonces 20](#_Toc9520126)

[3.9.1 Changement d’état 20](#_Toc9520127)

[3.10 Gestion des utilisateurs 21](#_Toc9520128)

[4. Analyse organique 22](#_Toc9520129)

[4.1 Modèle Physique (Modèle Relationnel) 22](#_Toc9520130)

[4.1.1 Users 23](#_Toc9520131)

[4.1.2 Roles 23](#_Toc9520132)

[4.1.3 Status 23](#_Toc9520133)

[4.1.4 Ads 24](#_Toc9520134)

[4.1.5 Pictures 24](#_Toc9520135)

[4.1.6 Brands 24](#_Toc9520136)

[4.1.7 Countries 24](#_Toc9520137)

[4.1.8 Genders 24](#_Toc9520138)

[4.1.9 Models 25](#_Toc9520139)

[4.1.10 Sizes 25](#_Toc9520140)

[4.1.11 States 25](#_Toc9520141)

[4.2 Classes utilisées 26](#_Toc9520142)

[4.2.1 Class Ad 26](#_Toc9520143)

[4.2.2 Class Brand 27](#_Toc9520144)

[4.2.3 Class Country 27](#_Toc9520145)

[4.2.4 Class Gender 27](#_Toc9520146)

[4.2.5 Class Model 27](#_Toc9520147)

[4.2.6 Class Picture 27](#_Toc9520148)

[4.2.7 Class Role 27](#_Toc9520149)

[4.2.8 Class Size 27](#_Toc9520150)

[4.2.9 Class State 28](#_Toc9520151)

[4.2.10 Class User 28](#_Toc9520152)

[4.3 Manager utilisés 28](#_Toc9520153)

[4.3.1 AdManager 28](#_Toc9520154)

[4.3.1.1 getAds 28](#_Toc9520155)

[4.3.1.2 getAdById 28](#_Toc9520156)

[4.3.1.3 createAd 29](#_Toc9520157)

[4.3.1.4 getAdsFromUser 29](#_Toc9520158)

[4.3.1.5 getAdsUsersNickname 29](#_Toc9520159)

[4.3.1.6 modifyAd 29](#_Toc9520160)

[4.3.1.7 modifyAdsState 29](#_Toc9520161)

[4.3.1.8 getAdsWithFilter 29](#_Toc9520162)

[4.3.1.9 getAdsFromUserWithFilter 30](#_Toc9520163)

[4.3.2 BrandManager 30](#_Toc9520164)

[4.3.2.1 getBrandsName 30](#_Toc9520165)

[4.3.2.2 getAllBrands 30](#_Toc9520166)

[4.3.3 CountryManager 31](#_Toc9520167)

[4.3.3.1 getCountrysName 31](#_Toc9520168)

[4.3.3.2 getAllCountries 31](#_Toc9520169)

[4.3.4 GenderManager 31](#_Toc9520170)

[4.3.4.1 getGendersName 31](#_Toc9520171)

[4.3.4.2 getAllGenders 31](#_Toc9520172)

[4.3.5 ModelManager 32](#_Toc9520173)

[4.3.5.1 getModelsName 32](#_Toc9520174)

[4.3.5.2 getAllModels 32](#_Toc9520175)

[4.3.6 PictureManager 32](#_Toc9520176)

[4.3.6.1 insertPicturesForAd 32](#_Toc9520177)

[4.3.6.2 getPicturesForAnAd 32](#_Toc9520178)

[4.3.7 SizeManager 33](#_Toc9520179)

[4.3.7.1 getSizesName 33](#_Toc9520180)

[4.3.7.2 getAllSizes 33](#_Toc9520181)

[4.3.8 StateManager 33](#_Toc9520182)

[4.3.8.1 getStatesName 33](#_Toc9520183)

[4.3.8.2 getAllStates 33](#_Toc9520184)

[4.3.9 StatusManager 34](#_Toc9520185)

[4.3.9.1 getStatusName 34](#_Toc9520186)

[4.3.9.2 getAllStatus 34](#_Toc9520187)

[4.3.10 UserManager 35](#_Toc9520188)

[4.3.10.1 getUsers 35](#_Toc9520189)

[4.3.10.2 getUserByNickname 35](#_Toc9520190)

[4.3.10.2 getUserByEmail 35](#_Toc9520191)

[4.3.10.3 createUser 35](#_Toc9520192)

[4.3.10.4 UserExist 35](#_Toc9520193)

[4.3.10.5 Connection 35](#_Toc9520194)

[4.3.10.6 modifyUsersStatus 36](#_Toc9520195)

[4.3.10.7 createToken 36](#_Toc9520196)

[4.3.11 SessionManager 36](#_Toc9520197)

[4.3.11.1 SetUser 36](#_Toc9520198)

[4.3.11.2 GetUser 36](#_Toc9520199)

[4.3.11.3 setNickname 36](#_Toc9520200)

[4.3.11.3 GetNickname 36](#_Toc9520201)

[4.3.11.3 GetRole 37](#_Toc9520202)

[4.3.12 MailManager 37](#_Toc9520203)

[4.3.12.1 initMailer 37](#_Toc9520204)

[4.3.12.2 sendMailForTokenValidation 37](#_Toc9520205)

[4.3.12.3 sendMailForAPurchase 37](#_Toc9520206)

[4.3.12.4 sendMailWhenUsersStatusChange 37](#_Toc9520207)

[4.3.12.4 sendMailWhenUsersAdsStateChange 37](#_Toc9520208)

[4.4 Class Database (Connexion à la base de données) 38](#_Toc9520209)

[4.4.1 Variable de classe 38](#_Toc9520210)

[4.4.2 Fonctions de la classe 38](#_Toc9520211)

[4.5 Arborescence de mon projet 38](#_Toc9520212)

[4.6 Mise en place de l'application 39](#_Toc9520213)

[5. Tests 40](#_Toc9520214)

[5.1 Scénarios de test 40](#_Toc9520215)

[5.1.1 Plan de test 40](#_Toc9520216)

[5.1.2 Scénario de test 42](#_Toc9520217)

[5.1.2.1 Scénario 1 – Accéder à l’application 42](#_Toc9520218)

[5.1.2.2 Scénario 2 – Visualisation des annonces 42](#_Toc9520219)

[5.1.2.3 Scénario 3 – Utilisations des filtres 43](#_Toc9520220)

[5.1.2.4 Scénario 4 – Achat d’un article 43](#_Toc9520221)

[5.1.2.5 Scénario 5 – Création d’un compte 44](#_Toc9520222)

[5.1.2.6 Scénario 6 – Connexion 44](#_Toc9520223)

[5.1.2.7 Scénario 7 – Création d’une annonce 45](#_Toc9520224)

[5.1.2.8 Scénario 8 – Visualisation et modification de mes annonces 46](#_Toc9520225)

[5.1.2.9 Scénario 9 – Gestion des utilisateurs 46](#_Toc9520226)

[5.1.2.10 Scénario 10 – Gestion des annonces 47](#_Toc9520227)

[5.1.2.11 Scénario 11 – Déconnexion 47](#_Toc9520228)

[6. Planning prévisionnel 48](#_Toc9520229)

[7. Planning effectif 49](#_Toc9520230)

[8. Conclusion 51](#_Toc9520231)

[9. Glossaire 52](#_Toc9520232)

[9.1 TPI 52](#_Toc9520233)

[9.2 « From Scratch » 52](#_Toc9520234)

[9.3 Javascript (JS) 52](#_Toc9520235)

[9.4 JQuery 52](#_Toc9520236)

[9.5 AJAX 52](#_Toc9520237)

[9.6 XML 52](#_Toc9520238)

[9.7 JSON 52](#_Toc9520239)

[9.8 Popup 52](#_Toc9520240)

[9.9 Template 52](#_Toc9520241)

# 1. Introduction

Pour le Travail Pratique Individuel (TPI), nous avons connu quelques changements quant aux années précédentes.   
Cette année, nous n’avions pas accès à notre énoncé jusqu’au jour précédent le début du TPI. Mon chef de projet pour le TPI est M. Aigroz. La personne m’ayant fourni le travail à réaliser est M. Court, responsable TPI.

# 2**. Contenu**

## 2.1Sujet

Création d’un site web de vente d’habits prénommé « Seconde Main ».

## 2.2 But

Réaliser une application web permettant à des visiteurs de parcourir les annonces postées sur le site et de faire des achats. Les utilisateurs peuvent acheter des articles aussi mais peuvent en créer / modifier / supprimer. L’administrateur ne peut pas faire ce dont l’utilisateur connecté peut, en revanche il peut gérer les statuts des utilisateurs et ceux des annonces aussi.

## 2.3 Spécifications

Un visiteur doit inscrire son email et/ou un mot de passe lors de l’achat d’un article.  
Un utilisateur connecté n’as pas besoin de réaliser l’étape précédente.  
La taille maximum d’un habit est XXL.  
Pour avoir accès à son compte, il faut qu’il soit validé.  
Il est obligatoire de posté une image lors de la création d’une annonce.

## 2.4 Environnement de développement

* Un PC standard école, 2 écrans
* Windows 10
* Apache / Module PHP / MySQL Server
* MySQL Workbench
* Visual Studio Code
* Suite Office

## 2.5 Livrable

* Planning
* Documentation technique
* Le résumé du TPI
* Manuel utilisateur intégré à l’application
* Journal de travail

# 3. Analyse fonctionnelle

Dans cette section je vais décrire toutes les fonctionnalités de l'application par le biais de maquettes écran.

## 3.0 Les différents profils utilisateur

L’application gère trois types de profils : le visiteur, l’utilisateur authentifié et l’administrateur.  
Chacun de ces profils donne accès à des fonctionnalités.  
Ces fonctionnalités sont décrites ci-après.

## 3.0Bis Les différents plan de navigation

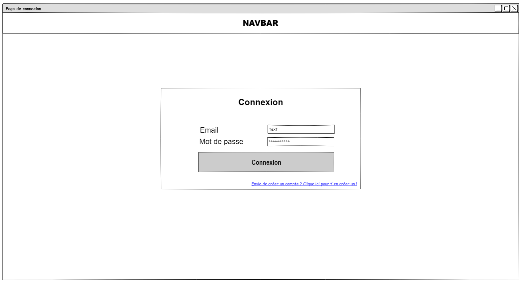
### 3.0Bis.1 Plan de navigation d’un visiteur

Le visiteur peut accéder à la page d’accueil, la page de connexion, la page d’inscription ainsi que la page de détails d’une annonce









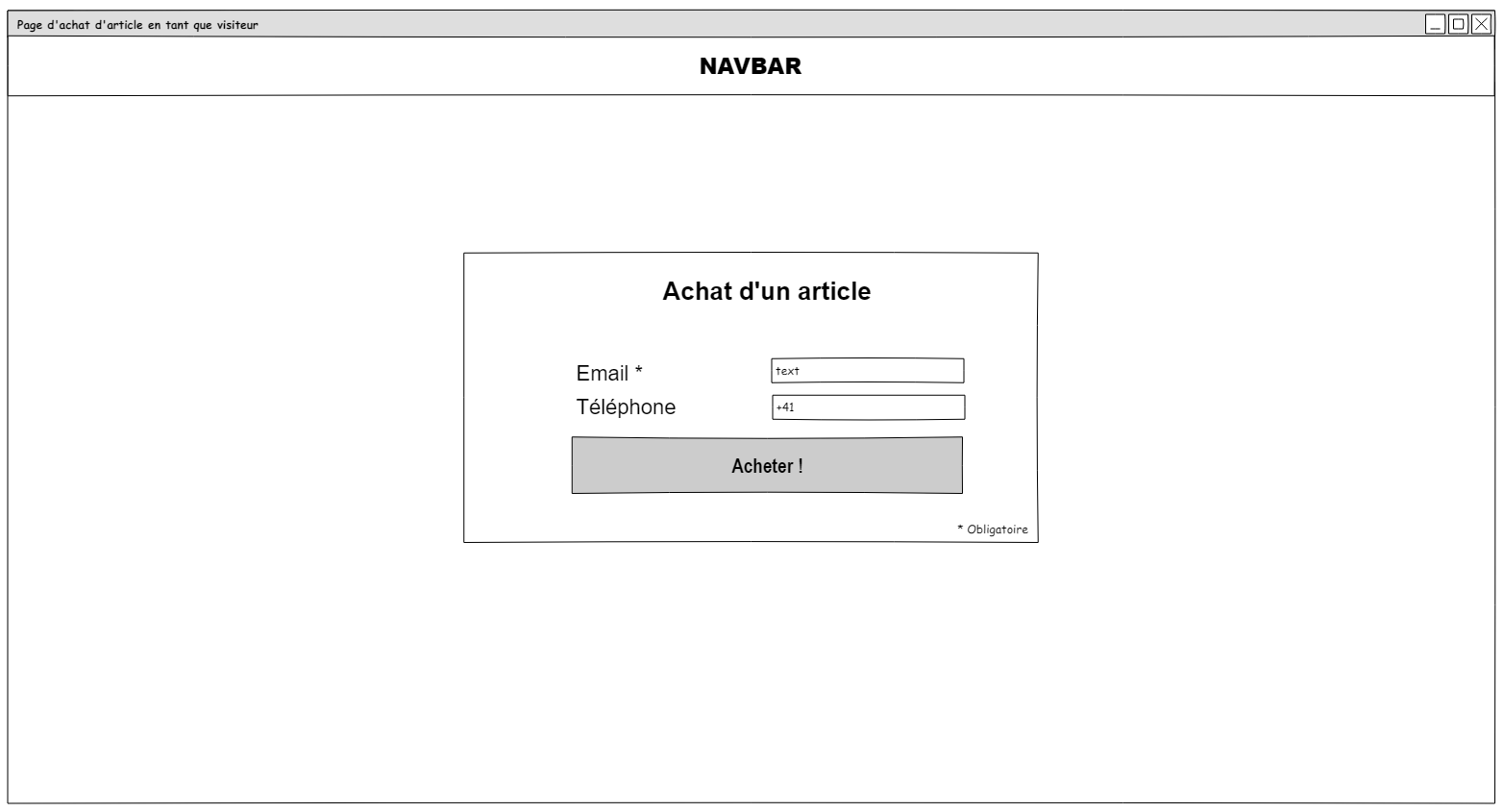
**8**

**7**

**4**



**9**



**7Bis**

### 3.0Bis.2 Plan de navigation d’un utilisateur connecté

Il a les mêmes accès que le visiteur mais en plus l’utilisateur connecté à accès à la page de création d’annonce, la page affichant toutes les annonces qu’il a posté et la page de modification.

À noter : Les pages dont l’accès est réservé à l’utilisateur connecté sont accessibles uniquement par la barre de navigation ou par redirection.







**10**

**4**

**11**



**12**

### 3.0Bis.3 Plan de navigation d’un administrateur

L’administrateur a accès à la page d’accueil, la page de détails d’une annonce, la page de gestion des utilisateurs ainsi que la page de gestion des annonces postées

À noter : Les pages dont l’accès est réservé à l’administrateur sont accessible uniquement par la barre de navigation ou par redirection.







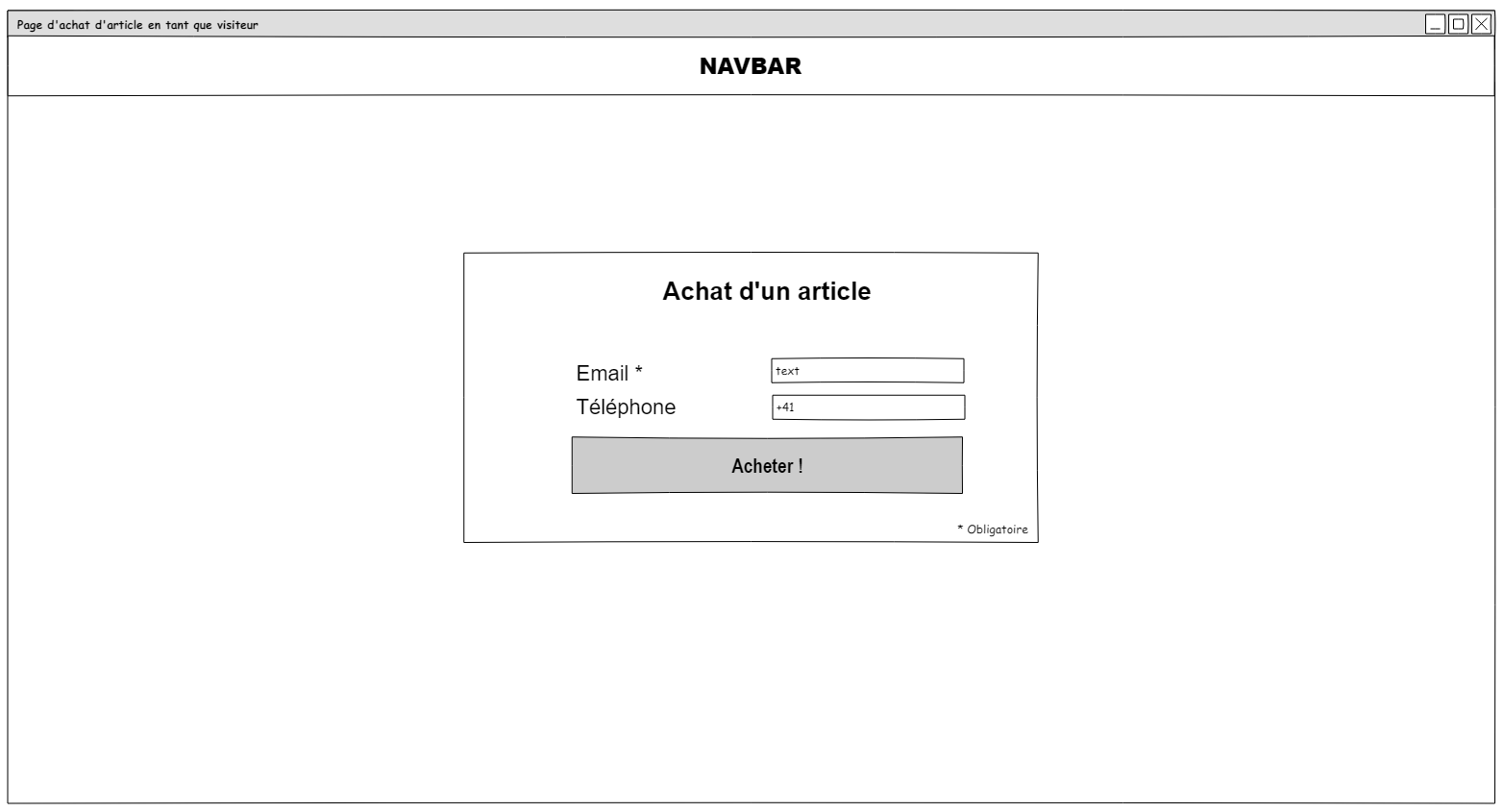
**15**

**14**

**4**



**7**



**7Bis**

## 3.1 Navbar

En fonction du profil, la barre de navigation intègre des fonctions liées à son rôle.

### 3.1.1 Visiteur

**A**

**B**

**1**



1. Bouton pour se diriger vers l’accueil du site. (Voir fig. 4)

### 3.1.2 Utilisateur authentifié

**D**

**C**

**B**

**A**

**2**



1. Bouton redirigeant vers l’accueil du site. (Voir fig. 4)
2. Bouton redirigeant sur la page permettant de créer une annonce (Voir fig. 10)
3. Bouton redirigeant sur la page répertoriant toutes les annonces postées par l’utilisateur connecté (Voir fig. 11)
4. Bouton permettant la déconnexion de l’utilisateur

### 3.1.3 Administrateur

**A**

**D**

**C**

**B**

**3**



1. Bouton redirigeant vers l’accueil du site. (Voir fig. 4)
2. Bouton redirigeant sur la page permettant de gérer les utilisateurs (Voir fig. 15)
3. Bouton redirigeant sur la page de gestions des annonces répertoriant toutes les annonces postées par les utilisateurs connectés (Voir fig. 13)
4. Bouton permettant la déconnexion de l’utilisateur

## 3.2 Accueil

La page principale, accessible aux visiteurs, répertorie les dernières annonces publiées par les utilisateurs. L’utilisateur a la possibilité de filtrer les annonces. Il accède à une annonce en cliquant sur l’image.

**4**



**F**

**D**

### 3.2.1 Système de filtrage

**B**

**A**



**5**

**E**

**C**

1. Plage de prix
2. Le modèle (Exemple : T-Shirt, Pantalon, Short, Pull-Over, etc…)
3. La taille de l’habit sous la forme de lettre tels que « L », « M », …, « XXL »
4. La marque du produit, par exemple « Nike », « Puma », « H&M », etc…
5. Détermine si le produit est pour femme, homme ou unisexe
6. L’état de l’annonce (Il existe « à vendre », « vendu », « en cours de vente », « suspendre », « supprimer »)

### 3.2.2 Description d’un « bloc » d’annonce

Voici ce à quoi ressemble un « bloc » d’annonce. Lors du clic sur l’image de l’annonce, l’utilisateur est redirigé sur la page de description d’un article. (Voir fig. 7)

**6**

**D**

**C**

**B**

**A**



1. L’image représentant le produit
2. Le titre du produit
3. La marque du produit
4. Le prix du produit

## 3.3 Détails d’une annonce

La page montrant les détails d’une annonce.

**K**

**J**

**I**

**H**

**G**

**F**

**E**

**D**

**C**

**B**

**A**

**7**



1. L’image représentant le produit
2. Le titre du produit
3. La description du produit
4. La marque du produit
5. La taille du produit (Les tailles vont de XS à XXL)
6. Le modèle du produit (Il existe toute sortes de modèle tels que : « T-Shirt », « Sweatshirt », « Chemise », etc…)
7. Le type de produit (Il existe : Femme, Homme et Unisexe)
8. Le pays du vendeur
9. Bouton d’achat du produit (S’il s’agit d’un visiteur, il sera redirigé sur une page de confirmation avec son email, voir fig. 8)
10. Le prix du produit
11. Date de mise en vente

### 3.3.1 Acheter un article

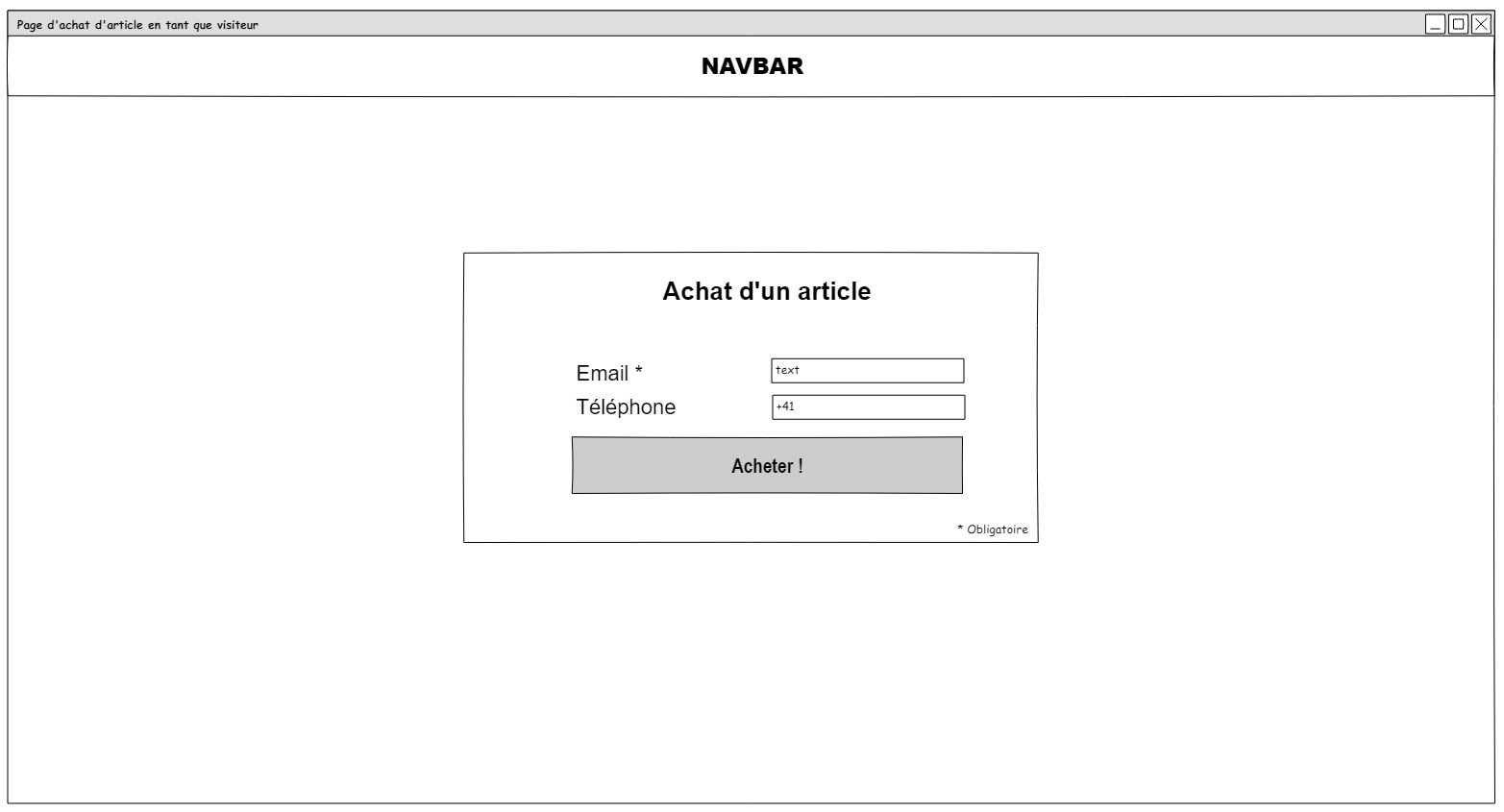
Page confirmant l’achat d’un article pour un visiteur.

**7Bis**

**C**

**B**

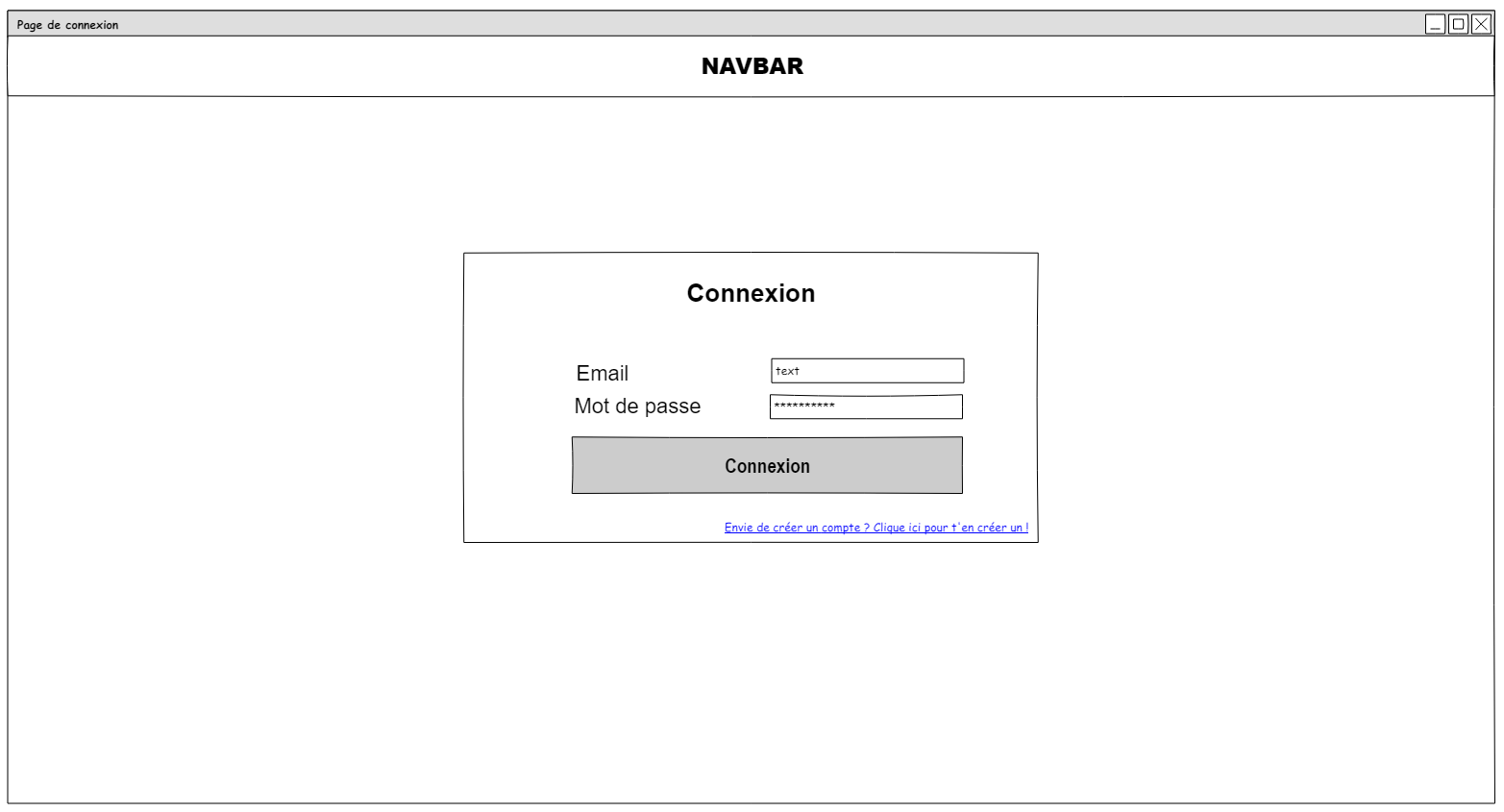
**A**



1. Email à fournir pour acheter un article (Obligatoire). Si l’utilisateur est connecté, son email est inscrit d’office
2. N° de téléphone
3. Bouton de validation d’achat

## 3.4 Connexion

Page permettant à l’utilisateur de se connecter. L’utilisateur doit spécifier son adresse email et son mot de passe.



**D**

**8**

**C**

**A**

**B**

1. Email de l’utilisateur
2. Mot de passe de l’utilisateur
3. Bouton de connexion
4. Lien redirigeant sur la page de création de compte

## 3.5 Inscription

Page d’inscription pour les utilisateurs.

**H**

**G**

**F**

**E**

**A**

**B**

**C**

**D**

**9**



1. Pseudonyme de l’utilisateur
2. Nom de famille de l’utilisateur
3. Prénom de l’utilisateur
4. Pays de résidence de l’utilisateur
5. N° de téléphone de l’utilisateur
6. Email de l’utilisateur
7. Mot de passe de l’utilisateur
8. Bouton de validation d’inscription

## 3.6 Création d’une annonce

Page permettant à l’utilisateur connecté de créer une annonce.

**10**

**I**

**B**

**F**

**G**

**E**

**A**

**C**

**H**

**D**



1. Titre de l’annonce
2. Marque du produit
3. Modèle du produit
4. Taille du produit
5. Type du produit
6. État de l’annonce
7. Prix de vente
8. Bouton permettant de charger une ou plusieurs image(s)
9. Bouton de création de l’annonce

## 3.7 Mes annonces

Page affichant à l’utilisateur toutes les annonces qu’il a posté. Comme pour les autres pages affichant des annonces, le système de tri est présent (Pour plus d’informations, voir Fig. 5).  
 Un bouton de modification est disponible, lors du clic sur ce dernier, l’utilisateur sera redirigé sur la page de modification de l’annonce. (Voir fig. 12)

**A**

**11**



1. Bouton de modification de l’annonce

## 3.8 Modification d’une annonce

Page de modification des informations pour l’annonce désirée.

**A**

**B**

**12**



(Pour avoir des informations complémentaires pour chaque champ, voir Fig. 10)

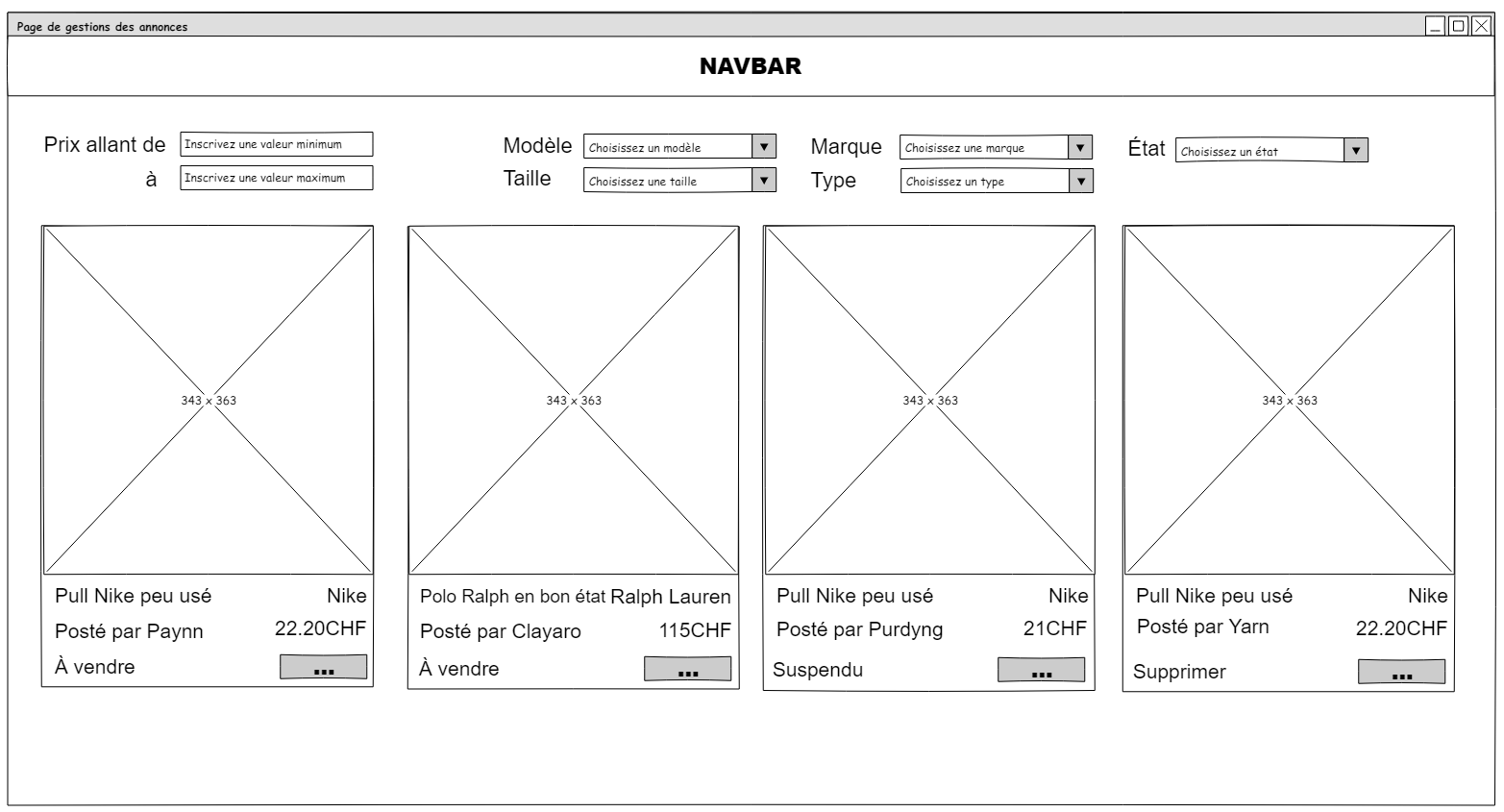
1. Toutes les images liées à l’annonce
2. Bouton de suppression d’une image

## 3.9 Gestion des annonces

Page d’administration permettant à l’administrateur de changer l’état des annonces publiées.

**A**

**13**



1. Bouton de changement d’état d’une annonce (voir fig. 14)

### 3.9.1 Changement d’état

Lors du clic sur le bouton « … », un menu déroulant apparaît laissant le choix à l’administrateur de l’état de l’annonce.

**A**

**14**



1. Menu déroulant affichant les états, lors du clic sur l’un des états, ce dernier deviendra l’état actuel de l’annonce.

## 3.10 Gestion des utilisateurs

Page d’administration permettant à l’administrateur de changer le statut d’un utilisateur inscrit. Il peut changer leur statut en choisissant celui voulu dans la liste déroulante.

**15**

**B**

**A**

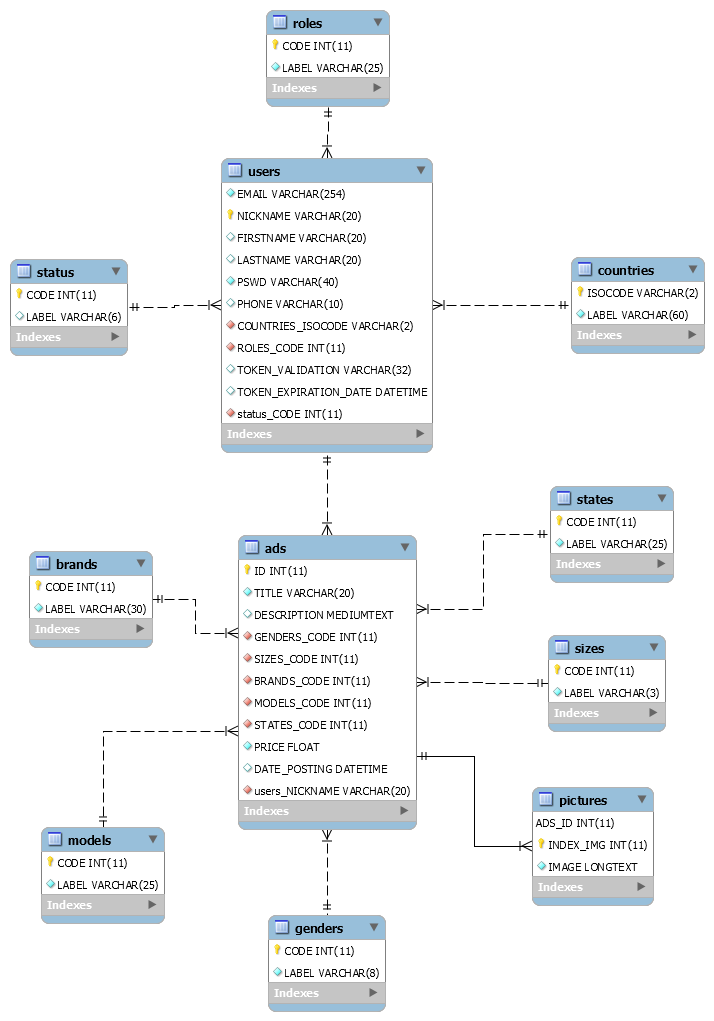


1. Pseudonyme de l’utilisateur
2. Statut de l’utilisateur

# 4. Analyse organique

## 4.1 Modèle Physique (Modèle Relationnel)

**16**



### 4.1.1 Users

Cette table contient la liste des utilisateurs inscrit sur « Seconde Main »

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| EMAIL | L’email de l’utilisateur | VARCHAR(254) |  |
| NICKNAME | Le pseudonyme de la personne détenant l'annonce | VARCHAR(20) | **X** |
| FIRSTNAME | Le prénom de l’utilisateur | VARCHAR(20) |  |
| LASTNAME | Le nom de famille de l’utilisateur | VARCHAR(20) |  |
| PSWD | Le mot de passe de l’utilisateur | VARCHAR(40) |  |
| PHONE | Le N° de téléphone de l’utilisateur | VARCHAR(10) |  |
| COUNTRIES\_ISOCODE | Le code du pays | VARCHAR(2) |  |
| ROLES\_CODE | Le code du rôle de l’utilisateur | INT(11) |  |
| TOKEN\_VALIDATION | La clé d’activation envoyé par mail à l’utilisateur lors de la création de compte | VARCHAR(32) |  |
| TOKEN\_EXPIRATION\_DATE | La date de péremption de la clé d’activation | DATETIME |  |
| status\_CODE | Le code du statut de l’utilisateur | INT(11) |  |

### 4.1.2 Roles

Cette table répertorie les différents rôles possibles pour un utilisateur inscrit

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID du rôle | INT(11) | **X** |
| LABEL | Le libellé du rôle | VARCHAR(25) |  |

### 4.1.3 Status

Cette table répertorie les différents statuts possibles pour un utilisateur inscrit

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID du statut | INT(11) | **X** |
| LABEL | Le libellé du statut | VARCHAR(6) |  |

### 4.1.4 Ads

Cette table contient la liste des annonces postées sur « Seconde Main »

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| ID | L’ID de l’annonce | INT(11) | **X** |
| TITLE | Le titre de l’annonce | VARCHAR(20) |  |
| DESCRIPTION | La description du produit | MEDIUMTEXT |  |
| GENDERS\_CODE | Le code du type du produit | INT(11) |  |
| SIZES\_CODE | Le code de la taille du produit | INT(11) |  |
| BRANDS\_CODE | Le code de la marque du produit | INT(11) |  |
| MODELS\_CODE | Le code du modèle du produit | INT(11) |  |
| STATES\_CODE | Le code de l'état du produit | INT(11) |  |
| PRICE | Le prix du produit | FLOAT |  |
| DATE\_POSTING | La date de mise en ligne de l'annonce | DATETIME |  |
| users\_NICKNAME | Le pseudonyme de la personne détenant l'annonce | VARCHAR(20) |  |

### 4.1.5 Pictures

Cette table contient la liste des images encodées en base64 affiliées à une annonce

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| ADS\_ID | L’ID de l’annonce | INT(11) | **X** |
| INDEX\_IMG | L’ordre d’ajout d’image | VARCHAR(20) | **X** |
| IMAGE | L’image encodée en base64 | LONGTEXT |  |

### 4.1.6 Brands

Cette table répertorie les différentes marques existantes

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID de la marque | INT(11) | **X** |
| LABEL | Le libellé de la marque | VARCHAR(25) |  |

### 4.1.7 Countries

Cette table répertorie les différents pays existants

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| ISOCODE | Le code ISO du pays | VARCHAR(2) | **X** |
| LABEL | Le libellé du pays | VARCHAR(25) |  |

### 4.1.8 Genders

Cette table répertorie les différents genres existant

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID du type d’habit | INT(11) | **X** |
| LABEL | Le libellé du type d’habit | VARCHAR(25) |  |

### 4.1.9 Models

Cette table répertorie les différents modèles existants

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID du modèle d’habit | INT(11) | **X** |
| LABEL | Le libellé du modèle | VARCHAR(25) |  |

### 4.1.10 Sizes

Cette table répertorie les différentes tailles existantes

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID de la taille d’habit | INT(11) | **X** |
| LABEL | Le libellé de la taille d’habit | VARCHAR(25) |  |

### 4.1.11 States

Cette table répertorie les différents états existants

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | PK |
| CODE | L’ID de l’état | INT(11) | **X** |
| LABEL | Le libellé de l’état | VARCHAR(25) |  |

## 4.2 Classes utilisées

Vous trouverez dans cette section les classes utilisées. Ces classes servent de conteneurs, ces dernières sont remplies avec les informations récupérées de la base de données.

### 4.2.1 Class Ad

Cette classe contient les informations sur une Annonce

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| id | L'id de l'annonce | INT |
| nickname | Le pseudonyme de la personne détenant l'annonce | STRING |
| title | Le titre de l'annonce | STRING |
| description | La description du produit | STRING |
| gender | Le code du type du produit | INT |
| size | Le code de la taille du produit | INT |
| brand | Le code de la marque du produit | INT |
| model | Le code du modèle du produit | INT |
| state | Le code de l'état du produit | INT |
| price | Le prix du produit | FLOAT |
| postingDate | La date de mise en ligne de l'annonce | DATETIME |

### 4.2.2 Class Brand

Cette classe contient les informations des différentes marques.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code de la marque | INT |
| label | Le libellé de la marque | STRING |

### 4.2.3 Class Country

Cette classe contient les informations des différents pays.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| isocode | Le code ISO du pays | INT |
| label | Le libellé du pays | STRING |

### 4.2.4 Class Gender

Cette classe contient les informations des différents types.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code du type | INT |
| label | Le libellé du type | STRING |

### 4.2.5 Class Model

Cette classe contient les informations des différents modèles d’habits.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code du modèle | INT |
| label | Le libellé du modèle | STRING |

### 4.2.6 Class Picture

Cette classe contient les informations sur une Image.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| idAd | L'id de l'annonce à laquelle l'image est reliée | STRING |
| index | L'ordre d'ajout de l'image | INT |
| img | L'image encodée en base64 | STRING |

### 4.2.7 Class Role

Cette classe contient les informations des différents rôles.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code du rôle | INT |
| label | Le libellé du rôle | STRING |

### 4.2.8 Class Size

Cette classe contient les informations des différentes tailles d'habits.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code de la taille | INT |
| label | Le libellé de la taille | STRING |

### 4.2.9 Class State

Cette classe contient les informations des différents états.

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| code | Le code de l'état | INT |
| label | Le libellé de l'état | STRING |

### 4.2.10 Class User

Cette classe contient les informations sur une personne (utilisateur) inscrite sur Seconde Main

|  |  |  |
| --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE |
| email | L'email de la personne | STRING |
| nickname | Le pseudo de la personne | STRING |
| firstname | Le prénom de la personne | STRING |
| lastname | Le nom de famille de la personne | STRING |
| phone | Le N° de téléphone de la personne | STRING |
| country | Le code du pays de résidence de la personne | INT |
| role | Le code du rôle de la personne | INT |
| pswd | Le mot de passe de la personne | STRING |
| status | Le statut de l’utilisateur | INT |

## 4.3 Manager utilisés

Les managers sont des classes manipulant les objets conteneurs ainsi que les données de la base de données.

### 4.3.1 AdManager

Cette classe contient les fonctions traitant ou récupérant des données concernant une annonce

#### 4.3.1.1 getAds

Fonction récupérant les informations de toutes les annonces

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau d’Ad | Toutes les annonces |

#### 4.3.1.2 getAdById

Fonction récupérant les informations d'une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | Object Ad | Les informations de l'annonce |

#### 4.3.1.3 createAd

Fonction créant une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| Ad | Object Ad |  |
| pictures | STRING[] |  |
|  | Boolean | True si OK, False si problème d'insertion |

#### 4.3.1.4 getAdsFromUser

Fonction récupérant les annonces postées par l'utilisateur connecté

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| nickname | STRING |  |
|  | Tableau d’Ad | Les informations de toutes les annonces de l'utilisateur |

#### 4.3.1.5 getAdsUsersNickname

Fonction récupérant le pseudonyme du propriétaire de l'annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| idAd | INT |  |
|  | STRING | Le pseudonyme du propriétaire de l'annonce |

#### 4.3.1.6 modifyAd

Fonction modifiant une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| Ad | Object Ad |  |
|  | Boolean | True si OK, False si problème de modification |

#### 4.3.1.7 modifyAdsState

Fonction modifiant l’état d’une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| Ad | Object Ad |  |
|  | Boolean | True si OK, False si problème de modification |

#### 4.3.1.8 getAdsWithFilter

Fonction récupérant les annonces selon le filtre appliqué

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | VALEUR RETOURNÉE |
| minPrice | Le prix minimum du produit | INT |  |
| maxPrice | Le prix maximum du produit | INT |  |
| brand | La marque du produit | INT |  |
| model | Le modèle du produit | INT |  |
| size | La taille du produit | INT |  |
| type | Le type du produit | INT |  |
| state | L'état de l'annonce | INT |  |
|  |  | Ad[] | Toutes les annonces existantes |

#### 4.3.1.9 getAdsFromUserWithFilter

Fonction récupérant les annonces selon le filtre appliqué du propriétaire de l'annonce

|  |  |  |  |
| --- | --- | --- | --- |
| VARIABLES | DESCRIPTION | TYPE | VALEUR RETOURNÉE |
| minPrice | Le prix minimum du produit | INT |  |
| maxPrice | Le prix maximum du produit | INT |  |
| brand | La marque du produit | INT |  |
| model | Le modèle du produit | INT |  |
| size | La taille du produit | INT |  |
| type | Le type du produit | INT |  |
| state | L'état de l'annonce | INT |  |
| nickname | Le pseudo du détenteur de l'annonce | STRING |  |
|  |  | Ad[] | Toutes les annonces existantes |

### 4.3.2 BrandManager

Cette classe contient les fonctions récupérant les données concernant une marque

#### 4.3.2.1 getBrandsName

Fonction récupérant le nom de la marque

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé de la marque |

#### 4.3.2.2 getAllBrands

Fonction récupérant toutes les informations concernant les marques

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de Brand | les informations concernant les marques |

### 4.3.3 CountryManager

Cette classe contient les fonctions récupérant les données concernant les pays

#### 4.3.3.1 getCountrysName

Fonction récupérant le label du pays en fonction de son Code ISO

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| isocode | STRING |  |
|  | STRING | Le libellé du pays |

#### 4.3.3.2 getAllCountries

Fonction récupérant toutes les informations concernant les pays

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de Country | les informations concernant les pays |

### 4.3.4 GenderManager

Cette classe contient les fonctions récupérant les données concernant le type d'un habit

#### 4.3.4.1 getGendersName

Fonction récupérant le label du type en fonction de son ID

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé du type |

#### 4.3.4.2 getAllGenders

Fonction récupérant toutes les informations concernant les types

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de Gender | les informations concernant les types |

### 4.3.5 ModelManager

Cette classe contient les fonctions récupérant les données concernant les différents modèles d'habits

#### 4.3.5.1 getModelsName

Fonction récupérant le label du modèle en fonction de son ID

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé du modèle |

#### 4.3.5.2 getAllModels

Fonction récupérant toutes les informations concernant les modèles

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de Model | les informations concernant les différents modèles |

### 4.3.6 PictureManager

Cette classe contient les fonctions récupérant / Ajoutant les données concernant les images d'une annonce

#### 4.3.6.1 insertPicturesForAd

Fonction insérant l(es) image(s) associé(es)

|  |  |  |  |
| --- | --- | --- | --- |
| paramètre(s) | Description | TYPE | valeur retournée |
| id | L'ID de l'annonce | INT |  |
| Pictures | Tableau d’images encodées en base 64 | STRING[] |  |
|  |  | Boolean | True si OK, False si problème d'insertion |

#### 4.3.6.2 getPicturesForAnAd

Fonction récupérant les images affiliées à une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | L'ID de l'annonce | INT |
|  | Tableau de Picture | L(es) image(s) de l'annonce |

### 4.3.7 SizeManager

Cette classe contient les fonctions récupérant concernant les tailles d'habits

#### 4.3.7.1 getSizesName

Fonction récupérant le label de la taille en fonction de son ID

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé de la taille |

#### 4.3.7.2 getAllSizes

Fonction récupérant toutes les informations concernant les tailles d’habits

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de Size | les informations concernant les différents modèles |

### 4.3.8 StateManager

Cette classe contient les fonctions récupérant concernant les différents états possibles pour une annonce

#### 4.3.8.1 getStatesName

Fonction récupérant le label de l'état en fonction de son ID

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé de l'état |

#### 4.3.8.2 getAllStates

Fonction récupérant toutes les informations concernant les tailles d’habits

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de State | les informations concernant les différents états |

### 4.3.9 StatusManager

Cette classe contient les fonctions récupérant concernant les différents statuts possibles pour un utilisateur

#### 4.3.9.1 getStatusName

Fonction récupérant le label du statut en fonction de son ID

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| id | INT |  |
|  | STRING | Le libellé de l'état |

#### 4.3.9.2 getAllStatus

Fonction récupérant toutes les informations concernant les statuts

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| aucun |  |  |
|  | Tableau de State | les informations concernant les différents états |

### 4.3.10 UserManager

Cette classe contient les fonctions traitant ou récupérant des données concernant un utilisateur

#### 4.3.10.1 getUsers

Fonction récupérant les informations de tous les utilisateurs

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| AUCUN |  |  |
|  | Tableau d’User | Les informations des utilisateurs |

#### 4.3.10.2 getUserByNickname

Fonction récupérant les informations d'un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| nickname | STRING |  |
|  | Objet User | Les informations de l’utilisateur |

#### 4.3.10.2 getUserByEmail

Fonction récupérant les informations d'un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| email | STRING |  |
|  | Objet User | Les informations de l’utilisateur |

#### 4.3.10.3 createUser

Fonction créant un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| User | Objet User |  |
|  | Boolean | True si OK, False si problème d'insertion |

#### 4.3.10.4 UserExist

Fonction vérifiant l'existence d'un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| email | STRING |  |
|  | Boolean | True si OK, False si inexistant |

#### 4.3.10.5 Connection

Fonction connectant un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| email | STRING |  |
| password | STRING |  |
|  | Boolean | True si OK, False si un champ est incorrect |

#### 4.3.10.6 modifyUsersStatus

Fonction modifiant le statut d'un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| nickname | STRING |  |
| idStatus | INT |  |
|  | Boolean | True si OK, False si problème de modification |

#### 4.3.10.7 createToken

Fonction créant un token de validation en comptant le nombre de lettre du pseudonyme de l'utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| nickname | STRING |  |
|  | Boolean | True si OK, False si problème de modification |

### 4.3.11 SessionManager

Classe récupérant et assignant à la variable $\_SESSION les différentes informations à remplir.

#### 4.3.11.1 SetUser

Fonction assignant les informations de l’utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| user | Objet User |  |
|  |  | AUCUNE |

#### 4.3.11.2 GetUser

Fonction récupérant les informations de l’utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| AUCUN | Objet User |  |
|  | Objet User | Les informations de l’utilisateur |

#### 4.3.11.3 setNickname

Fonction assignant le pseudonyme de l’utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| nickname | STRING |  |
|  |  | AUCUNE |

#### 4.3.11.3 GetNickname

Fonction récupérant le pseudonyme de l’utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| AUCUN |  |  |
|  | STRING | Le pseudonyme de l’utilisateur |

#### 4.3.11.3 GetRole

Fonction récupérant le rôle de l’utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| AUCUN |  |  |
|  | STRING | Le rôle de l’utilisateur |

### 4.3.12 MailManager

Classe envoyant les emails

#### 4.3.12.1 initMailer

Fonction créant une instance SMTP

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| AUCUN |  |  |
|  | Instance Swift\_SmtpTransport | Le SMTP |

#### 4.3.12.2 sendMailForTokenValidation

Fonction envoyant un mail lors de la création de compte

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| token | STRING |  |
| emailUser | STRING |  |
|  | boolean | True si ok, False si problème d'envois |

#### 4.3.12.3 sendMailForAPurchase

Fonction envoyant un mail lors d'une demande d'achat

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| userWhoPurchase | STRING |  |
| userPhone | STRING |  |
| userWhoSell | STRING |  |
| adTitle | STRING |  |
|  | boolean | True si ok, False si problème d'envois |

#### 4.3.12.4 sendMailWhenUsersStatusChange

Fonction envoyant un mail lors du changement de statut d'un utilisateur

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| emailUsersStatusModified | STRING |  |
| state | INT |  |
| adTitle | STRING |  |
|  | boolean | True si ok, False si problème d'envois |

#### 4.3.12.4 sendMailWhenUsersAdsStateChange

Fonction envoyant un mail lors du changement de l'état d'une annonce

|  |  |  |
| --- | --- | --- |
| paramètre(s) | TYPE | valeur retournée |
| emailOwnersAd | STRING |  |
| status | INT |  |
|  | boolean | True si ok, False si problème d'envois |

## 4.4 Class Database (Connexion à la base de données)

Pour le TPI, j’ai utilisé la classe rédigée par M. Aigroz lors de ma 3ème année (En école entreprise). Elle utilise le principe du Singleton.

### 4.4.1 Variable de classe

|  |  |  |
| --- | --- | --- |
| VARIABLE | DESCRIPTION | TYPE |
| pdoInstance | L’instance PDO permettant l’accès à la base de données | PDO |

### 4.4.2 Fonctions de la classe

|  |  |  |
| --- | --- | --- |
| NOM DE LA FONCTION | DESCRIPTION | PARAMÈTRE(S) |
| *private* \_\_construct() | Constructeur privé pour que personnes ne puisse créer de nouvelle instance via  ' = new Database();' | **AUCUN** |
| *private* \_\_clone() | Clone privé pour que personne ne puisse cloner l'instance | **AUCUN** |
| *public static* getInstance() | Retourne l'instance PDO ou en créer une si l’instance est inexistante | **AUCUN** |
| *final public static* \_\_callStatic ($, $arrArguments) | Est lancée lorsque l'on appelle des méthodes inaccessibles dans un contexte statique | **chrMethod** : Le nom de la fonction appelée  **arrArguments** : Le tableau d’arguments à passer à chrMethod |

## 4.5 Arborescence de mon projet

**/www**  
Répertorie les différents fichiers et les différents sous-dossier de l'application web

**/www/server**  
Répertorie les différents sous-dossier des scripts stockés sur le serveur

**/www**/**server/ajax**  
Répertorie les différents script d'appels AJAX

**/www/server/constants**  
Répertorie les différents fichiers de constantes

**/www/server/config**Répertorie les différents fichiers de config

**/www/server/css**  
Répertorie les différents fichiers de style

**/www/server/database**  
Répertorie le script de connexion à la base de données

**/www/server/dhtmlxSuite**Répertorie les différentes scripts de la librairie dhtmlxSuite

**/www/server/fonts**Répertorie les différentes police d'écritures

**/www/server/imgProduct**  
Répertorie les différentes images de produit à insérer pour les tests

**/www/server/inc**Répertorie les différents scripts devant être inclus dans quasi chaque pages

**/www/server/js**Répertorie les différentes script javascript de bootstrap

**/www/server/manager**Répertorie les différents Manager

**/www/server/model**Répertorie les différentes classes conteneurs utilisées

**/www/server/screenshot**Répertorie les différentes capture d'écran de l'application pour la partie d'aide

**/www/server/swiftmailer5**Répertorie les différents scripts de la librairie dhtmlxSuite

**/www/server/templates**Répertorie les différents template graphique de l'application

**/www/server/tests**Répertorie les différents tests unitaires

**/www/server/userManual**  
Répertorie les différentes pages pour le manuel utilisateur

**/docs**Répertorie les différents documents d'analyse, de documentations technique ainsi que les différentes scripts de création de base de données

**/docs/Analysis**Répertorie les différents documents d'analyse

**/docs/Analysis/Maquettage**Répertorie les différentes images de maquettage de l'application

**/docs/Analysis/Database**Répertorie les différents sous-dossier pour l'importation du schéma de base de données ainsi que les scripts pour remplir les tables

## 4.6 Mise en place de l'application

Scripts a utilisé dans l'ordre ci-dessous

1. createUserDB.sql
2. secondHand-Structure.sql
3. secondHand-Datas.sql

# 5. Tests

## 5.1 Scénarios de test

Les tests unitaires des fonctions manipulant la base de données ont été réalisés en PHP et sont classés dans le sous-répertoire /tests. Pour le reste, ce sont des fonctions globales de l’application.

### 5.1.1 Plan de test

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| N° test | But | Fonction | Fichier | OK |
| 1 | Si le mot de passe saisie par l’utilisateur concorde avec celui en base, retourne true sinon false | UserManager::Connection($email, $pwd) | testConnection | **X** |
| 2 | Si l’utilisateur a bien été inséré en base, retourne true sinon problème d’insertion | UserManager::createUser($user) | testCreateAd | **X** |
| 3 | Si l’annonce a bien été insérée en base, retourne true sinon false donc problème d’insertion | AdManager::createAd($ad, $pictures) | testCreateUser | **X** |
| 4 | En passant en paramètre l’ID de l’annonce, retourne le pseudonyme de l’utilisateur sinon false donc l’annonce est inexistante | AdManager::getAdsUsersNickname($id) | testGetAdsUsersNickname | **X** |
| 5 | Lors de l’application du filtre, retourne que les annonces filtrées si aucunes annonces n’est présente retourne false | AdManager::getAdsWithFilter(15, null, 1, null, null, 1) | testGetAdsWithParameters | **X** |
| 6 | Retourne toutes les annonces existantes | AdManager::getAds() | testGetAllAds | **X** |
| 7 | Retourne toutes les marques existantes | BrandManager::getAllBrands() | testGetAllBrands | **X** |
| 8 | Retourne tous les pays existants | CountryManager::getAllCountries() | testGetAllCountries | **X** |
| 9 | Retourne tous les modèles existants | ModelManager::getAllModels() | testGetAllModels | **X** |
| 10 | Retourne tous les états existants | StateManager::getAllStates() | testGetAllStates | **X** |
| 11 | On passe en paramètre l’id de l’annonce, retourne un tableau d’images encodées en base64 | PictureManager::getPicturesForAnAd($idAd) | testGetAllUsers | **X** |
| 12 | Retourne tous les utilisateurs existants | UserManager::getUsers() | testGetInfosUserByEmail | **X** |
| 13 | Passe en paramètre l’email de l’utilisateur, retourne un User | UserManager::getUserByEmail($email) | testGetInfosUserByNickname | **X** |
| 14 | Passe en paramètre le pseudonyme de l’utilisateur, retourne un User | UserManager::getUserByNickname($nickname) | testGetPicturesForAnAd | **X** |
| 15 | Si l’utilisateur existe, retourne true sinon false si inexistant | UserManager::UserExist($email) | testInsertPicturesForAd | **X** |
| 16 | On passe en paramètre l’Id de l’annonce, retourne un tableau d’image encodée en base 64 | PictureManager::getPicturesForAnAd($idAd) | testMailSample | **X** |
| 17 | La modification d’annonce retourne true si bien inséré sinon false | AdManager::modifyAd($ad) | testModifyAd | **X** |
| 18 | Lors du clique sur le bouton acheter la fenêtre de popup s’ouvre |  | testPopupBuy | **X** |
| 19 | Dans la popup, lors du clique sur acheter l’envois de demande d’achat s’effectue |  | testPopupBuy | **X** |
| 20 | Pour chaque <select> lors du changement d’item sélectionné appliqué le filtre aux annonces affichées |  | accueil, mesAnnonces, gestionAnnonces | **X** |
| 21 | Lors de la connexion, les variables de sessions se remplissent |  | connexion | **X** |
| 22 | Seul le détenteur de l’annonce a accès à la page de modification de l’annonce |  | modification | **X** |
| 23 | Lors du changement de l’état d’une annonce, un mail est envoyé au propriétaire de l’annonce | MailManager::sendMailWhenUsersAdsStateChange($email, $idState, $title) | ajaxModifyAdsState | **X** |
| 24 | La page mes annonces affiche les annonces de l’utilisateur connecté |  | mesAnnonces | **X** |
| 25 | On passe en paramètre le pseudonyme de l’utilisateur, retourne la date d’expiration ainsi que la clé d’activation par référence | UserManager::getTokenAndDateExpiration($nickname, $dateExpiration, $tokenUser) |  | **X** |
| 26 | Si la clé d’activation est valide et qu’elle n’est pas périmée, elle change le statut de l’utilisateur en Validé | UserManager::modifyUsersStatus($nickname, STATUS\_USER\_VALIDATED) |  | **X** |

### 5.1.2 Scénario de test

#### 5.1.2.1 Scénario 1 – Accéder à l’application

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur le page : <http://127.0.0.1> ou <http://localhost> | - | La page d’accueil s’affiche. |  |  |

#### 5.1.2.2 Scénario 2 – Visualisation des annonces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur la page accueil | - | Affiches toutes les annonces à vendre. |  |  |

#### 5.1.2.3 Scénario 3 – Utilisations des filtres

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| L’utilisateur rentre un prix minimum | Prix minimum : 20 | Affiche toutes les annonces coûtant plus de 20CHF. |  |  |
| L’utilisateur rentre un prix maximum | Prix maximum: 120 | Affiche toutes les annonces coûtant moins de 120CHF. |  |  |
| L’utilisateur rentre un prix minimum et maximum | Prix minimum : 20  Prix maximum: 120 | Affiche toutes les annonces ayant pour pris entre 20 et 120 CHF. |  |  |
| L’utilisateur sélectionne une marque | Marque : Nike | Affiches toutes les annonces ayant pour marque « Nike ». |  |  |
| L’utilisateur sélectionne un modèle | Modèle: Sweatshirt | Affiches toutes les annonces ayant pour modèle « Sweatshirt». |  |  |
| L’utilisateur sélectionne une taille | Taille: M | Affiches toutes les annonces ayant pour taille « M ». |  |  |
| L’utilisateur sélectionne un genre | Genre : Unisexe | Affiches toutes les annonces ayant pour genre « Unisexe». |  |  |
| L’utilisateur sélectionne un état | État :Vendue | Affiches toutes les annonces ayant pour état « Vendue». |  |  |
| L’utilisateur rentre un prix minimum, une marque et une taille | Prix minimum : 20  Marque : Nike  Taille: XL | Affiches toutes les annonces ayant pour prix minimum 20CHF, la marque « Nike » ainsi que la taille XL. |  |  |

#### 5.1.2.4 Scénario 4 – Achat d’un article

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| L’utilisateur clique sur l’image de couverture de l’annonce | - | La page de détails de l’annonce s’affiche. |  |  |
| L’utilisateur cliquer sur acheter | - | Une popup s’ouvre en bas à droite. |  |  |
| L’utilisateur rentre son email dans la popup d’achat | Email : [gawen.ackrm@eduge.ch](mailto:gawen.ackrm@eduge.ch) | La demande d’achat est envoyée par mail au détenteur de l’annonce. Celle-ci n’est donc plus achetable. |  |  |

#### 5.1.2.5 Scénario 5 – Création d’un compte

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| L’utilisateur clique sur le lien en bas à droite du formulaire de connexion | - | la page de création de compte s’affiche. |  |  |
| L’utilisateur rentre ses informations pour créer un compte | Pseudo : ACKERMANNGUE Nom de famille : Ackermann  Prénom : Gawen  Pays : Switzerland  N° de téléphone : 0798898694  Email : gawen.ackrm@eduge.ch  Mot de passe : 123 | Il est redirigé sur la page d’accueil. Un mail de confirmation de compte lui a été envoyé. |  |  |
| Depuis la boite mail, l’utilisateur clique sur le lien « clique ici pour valider ton compte » | - | Son compte est maintenant « Validé », il peut donc se connecter. |  |  |

#### 5.1.2.6 Scénario 6 – Connexion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| se rendre sur la page de connexion | - | La page de connexion s’affiche. |  |  |
| L’utilisateur saisie son email et son mot de passe et clique sur « Connexion ! » | Email : [gawen.ackrm@eduge.ch](mailto:gawen.ackrm@eduge.ch)  Mot de passe : 123 | Il est connecté, puis redirigé sur la page d’accueil. |  |  |
| L’utilisateur saisie son email et son mot de passe et clique sur « Connexion ! » | Email : [gawen.ackrm@eduge.ch](mailto:gawen.ackrm@eduge.ch)  Mot de passe : 122 | Une alerte lui survient lui disant qu’il a soit écrit faux son email et/ou son mot de passe ou que son compte ne soit pas encore activé / temporairement bloqué. |  |  |

#### 5.1.2.7 Scénario 7 – Création d’une annonce

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur la page de création d’annonce | - | La page de création d’annonce s’affiche |  |  |
| L’utilisateur créer une annonce. Il remplit les différents champs. Il ajoute les images du produit 1 par 1 | Titre : T-shirt en bon état  Description : Je mets en vente mon T-Shirt Obey car il appartenait à mon frère mais il ne l’a jamais utilisé.  Marque : Obey  Modèle : T-Shirt  Taille : S  Type : Homme  État : à vendre  Prix : 15  Photo : obey\_Tshirt.jpg | Création de l’annonce. Redirection sur la page d’accueil. |  |  |

#### 5.1.2.8 Scénario 8 – Visualisation et modification de mes annonces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur la page mes annonces | - | La page répertoriant les annonce de l’utilisateur connecté s’affichent |  |  |
| L’utilisateur clique sur le bouton « Modifier » en dessous de l’annonce. | - | La page de modification de l’annonce s’affiche. |  |  |
| L’utilisateur modifie l’annonce souhaitée. Il modifie les différents champs. Il ajoute les images du produit 1 par 1 et / ou en supprime. | Titre : T-shirt bon état !  Description : Je mets en vente mon T-Shirt Obey. Il n’a été porté que très peu.  Marque : Obey  Modèle : T-Shirt  Taille : S  Type : Homme  État : à vendre  Prix : 15  Photo : obey\_Tshirt.jpg | Modification de l’annonce. Redirection sur la page d’accueil. |  |  |

#### 5.1.2.9 Scénario 9 – Gestion des utilisateurs

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur la page de gestion des utilisateurs | - | La page de gestion des utilisateur s’affiche |  |  |
| L’administrateur change le statut de l’utilisateur souhaité | Pseudonyme de l’utilisateur : ACKERMANNGUE  Statut : Bloqué | L’utilisateur **ACKERMANNGUE** est maintenant bloqué. Un mail de changement de statut lui a été transmis. |  |  |

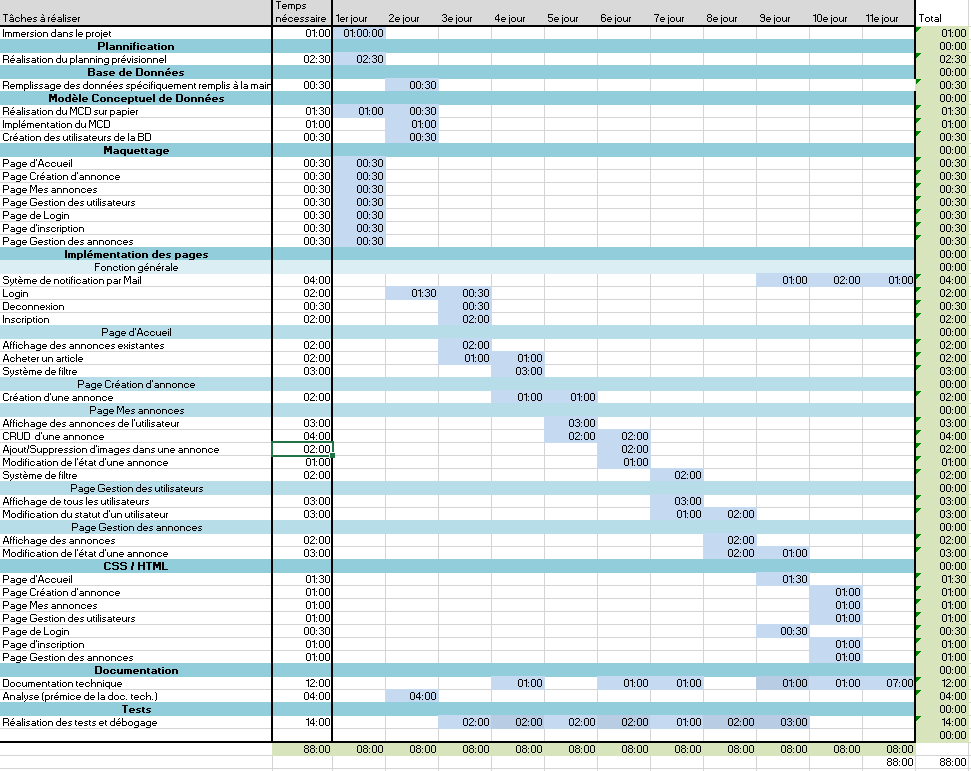
#### 5.1.2.10 Scénario 10 – Gestion des annonces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| Se rendre sur la page de gestion des annonces | - | La page de gestion des annonces s’affiche |  |  |
| L’administrateur change l’état de l’annonce souhaitée. | État : Suspendue | L’annonce a changé d’état et un mail de notification du changement d’état a été envoyé au détenteur de l’annonce. |  |  |

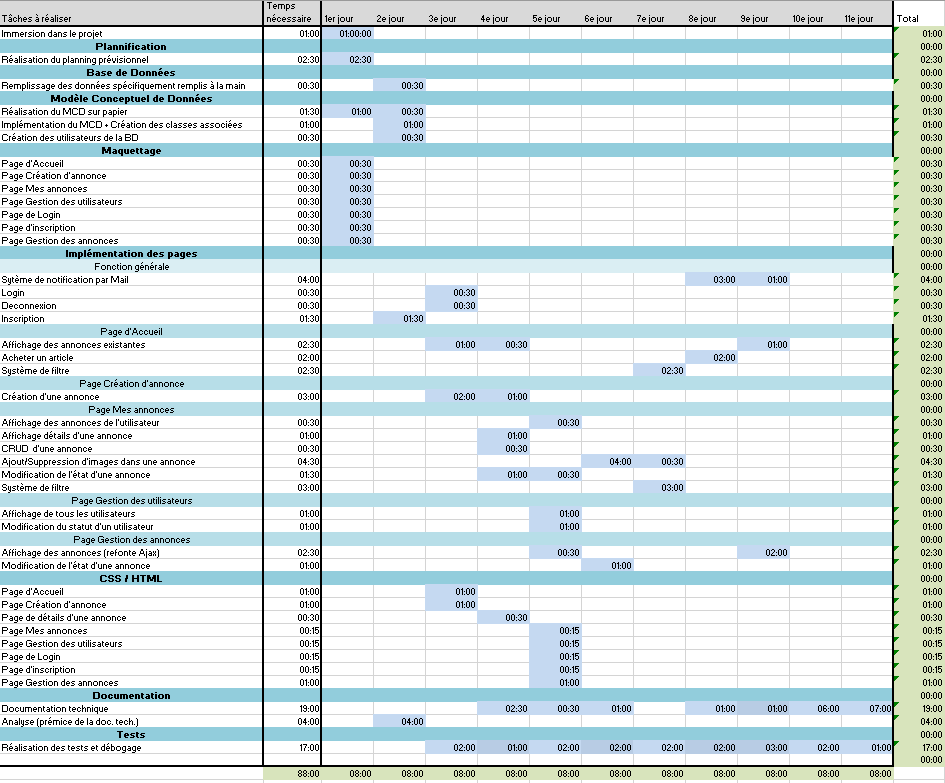
#### 5.1.2.11 Scénario 11 – Déconnexion

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACTION | DATA(S) | RÉSULTAT ATTENDU | FONCTIONNE ? | REMARQUE |
| L’utilisateur se déconnecte | - | Les informations de l’utilisateur sont vidées et il est redirigé sur la page d’accueil |  |  |

# 6. Planning prévisionnel



# 7. Planning effectif



# 8. Conclusion

Après avoir passé 11 jours sur mon projet de TPI, j’ai pu constater plusieurs choses.

Premièrement, j’ai vraiment apprécié d’être lancé sur un projet « *from scratch* » car je trouve que devoir faire toute la partie d’analyse est très important pour comprendre au mieux le projet sur lequel on est lancé.

Secondement, durant la partie d’analyse j’ai aussi bien aimé de devoir réaliser des maquettes pour être sûr à 100% que l’on a toutes les informations et fonctions nécessaires. Je suis fière de moi car j’ai réussi à respect mon cahier des charges et créer une application qui peut être réellement utile aux personnes souhaitant vendre leurs habits.

Troisièmement, j’ai apprécié par-dessus tout de coder cette application. J’ai d’abord créer mes classes conteneurs puis les managers. Avant d’implémenter les fonctionnalités, j’ai d’abord créer des tests unitaires pour les plus gros éléments devant être implémentés.

Quatrièmement, j’ai pu apprendre à perfectionner mes recherches lors de problème ou d’incompréhension car lorsque j’avais besoin de créer du code JS en utilisant du JQuery pour effectuer des appels AJAX ou tout simplement interagir avec la page sur laquelle l’utilisateur se trouve je cherchais beaucoup sur internet pour voir en général qu’est-ce qui fait et mieux comprendre la logique derrière. J’ai aussi pu apprendre à utiliser la librairie « dHTMLx », c’est avec cette dernière que j’ai créé la fenêtre de popup lors de l’achat d’un article.

Cinquièmement, je pense que l’aide de M. Aigroz m’a été très gratifiante car en effet les échanges avec M. Aigroz ont tous été très rapide et très favorable au bon déroulement de mon TPI. Lors d’incompréhension sur des points de mon cahier des charges, il s’est à chaque fois efforcé à ce que je comprenne. Pareil lorsque je ne comprenais pas comment utiliser certaines fonctionnalités JQuery que j’avais regardé sur internet mais que je ne comprenais pas.

Pour finir, j’ai trouvé qu’en étant seul, certes la charge de travail est bien plus importante mais le concept de pouvoir gérer ce que l’on veut développer (tout en gardant un ordre logique).

# 9. Glossaire

## 9.1 TPI

L’abréviation TPI signifie Travail Pratique Individuel.

## 9.2 « From Scratch »

L’expression « From Scratch » signifie qu’on part de rien du tout.

Exemple : Il y a tellement d’erreurs dans ce programme que le développeur décida de le réécrire « from scratch ». Dans ce contexte il signifie « à partir de rien. »

## 9.3 Javascript (JS)

C’est un langage de programmation opérant côté client.

## 9.4 JQuery

C’est une librairie Javascript permettant de faciliter certaines fonctionnalités Javascript. L'utilisation de cette librairie permet de gagner du temps de développement lors de l'interaction sur les balises HTML d'une page web, l'AJAX ou la gestion des évènements.

## 9.5 AJAX

AJAX est l'acronyme d'Asynchronous JavaScript And XML, autrement dit JavaScript Et XML Asynchrones.   
AJAX n'est ni une technologie ni un langage de programmation ; AJAX est un concept de programmation Web reposant sur plusieurs technologies comme le JavaScript et le XML. Dans mon cas je retourne des informations en JSON et pas en XML

## 9.6 XML

« eXtensible Markup Language » autrement appelé XML, il s’agit d’un langage de structuration de données, utilisé pour la gestion et l'échange d'informations sur Internet.

## 9.7 JSON

« JavaScript Object Notation » autrement appelé JSON est un format de données textuelles dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l’information structurée comme le permet XML.

## 9.8 Popup

Il s’agit d’une fenêtre surgissant lors d’un clique ou contre son gré ou une fenêtre secondaire qui s'affiche.

## 9.9 Template

Il s’agit d’un terme désignant un exemple sur lequel on se base pour concevoir un logiciel, un design de site ou autre.