

DOSSIER PROFESSIONNEL^(DP)

<i>Nom de naissance</i>	▶ Cabassot
<i>Nom d'usage</i>	▶ Cabassot
<i>Prénom</i>	▶ Emmanuel
<i>Adresse</i>	▶ 22 rue Poucel, 13004 Marseille

Titre professionnel visé

Développeur web et web mobile

MODALITE D'ACCES :

- ☐ Parcours de formation
- ☐ Validation des Acquis de l'Expérience (VAE)

DOSSIER PROFESSIONNEL ^(DP)

Présentation du dossier

Le dossier professionnel (DP) constitue un élément du système de validation du titre professionnel.
Ce titre est délivré par le Ministère chargé de l'emploi.

Le DP appartient au candidat. Il le conserve, l'actualise durant son parcours et le présente **obligatoirement à chaque session d'examen.**

Pour rédiger le DP, le candidat peut être aidé par un formateur ou par un accompagnateur VAE.

Il est consulté par le jury au moment de la session d'examen.

Pour prendre sa décision, le jury dispose :

1. des résultats de la mise en situation professionnelle complétés, éventuellement, du questionnaire professionnel ou de l'entretien professionnel ou de l'entretien technique ou du questionnement à partir de productions.
2. du **Dossier Professionnel (DP)** dans lequel le candidat a consigné les preuves de sa pratique professionnelle.
3. des résultats des évaluations passées en cours de formation lorsque le candidat évalué est issu d'un parcours de formation
4. de l'entretien final (dans le cadre de la session titre).

[Arrêté du 22 décembre 2015, relatif aux conditions de délivrance des titres professionnels du ministère chargé de l'Emploi]

Ce dossier comporte :

- ▶ pour chaque activité-type du titre visé, un à trois exemples de pratique professionnelle ;
- ▶ un tableau à renseigner si le candidat souhaite porter à la connaissance du jury la détention d'un titre, d'un diplôme, d'un certificat de qualification professionnelle (CQP) ou des attestations de formation ;
- ▶ une déclaration sur l'honneur à compléter et à signer ;
- ▶ des documents illustrant la pratique professionnelle du candidat (facultatif)
- ▶ des annexes, si nécessaire.

Pour compléter ce dossier, le candidat dispose d'un site web en accès libre sur le site.

DOSSIER PROFESSIONNEL ^(DP)



<http://travail-emploi.gouv.fr/titres-professionnels>

Sommaire

Exemples de pratique professionnelle

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.	5
▶ Portfolio (maquetter une application, réaliser une interface utilisateur web statique et adaptable)	p.	6
▶ ToDoList (Développer une interface utilisateur web dynamique)	p.	10
Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité	p.	15
▶ Boutique de boutiques (créer une base de données, développer les composants d'accès aux données, développer la partie back-end d'une application web ou web mobile)	p.	15
<hr/>		
Titres, diplômes, CQP, attestations de formation <i>(facultatif)</i>	p.	27
Déclaration sur l'honneur	p.	28
Documents illustrant la pratique professionnelle <i>(facultatif)</i>	p.	29
Annexes <i>(Si le RC le prévoit)</i>	p.	30

EXEMPLES DE PRATIQUE PROFESSIONNELLE

DOSSIER PROFESSIONNEL ^(DP)

Activité-type 1

Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n°1 ▶ Portfolio

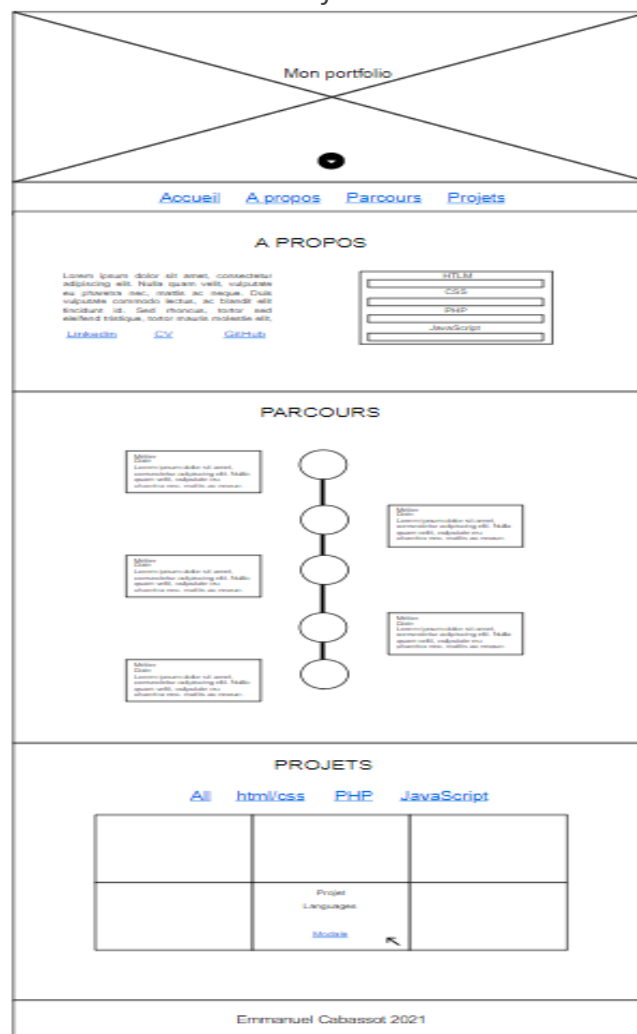
DOSSIER PROFESSIONNEL (DP)

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

PORTFOLIO

Réalisation de mon portfolio. Il a été fait sans template et sans Bootstrap. Il est responsive dans l'optique qu'il puisse être consulté quel que soit le support. (Desktop, Tablette, portable).

Wireframe

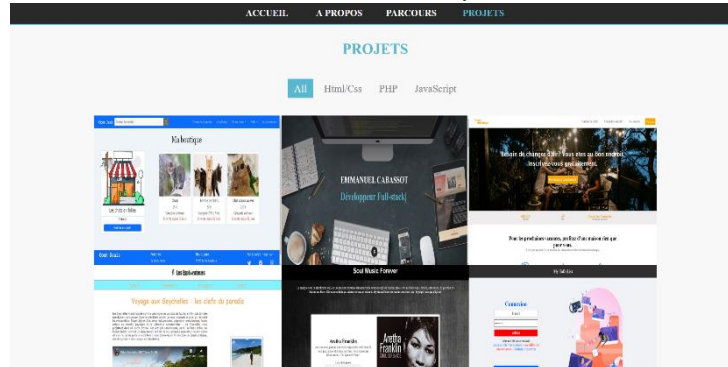


Il comporte les informations suivantes :

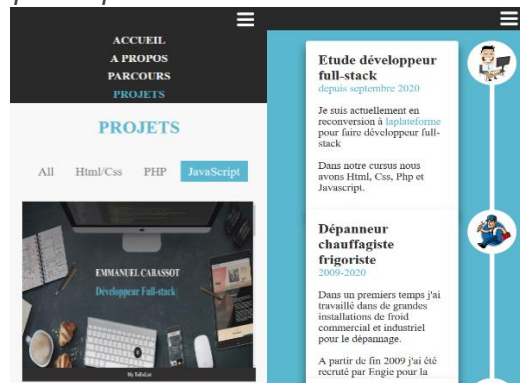
- ✓ La page présente mon profil avec mon cv téléchargeable et mon lien GitHub.
- ✓ Mes compétences techniques avec une animation des barres de compétences.
- ✓ Mon parcours avec une animation descendante des étiquettes.
- ✓ Les photos de mes projets font apparaitre au passage de la souris un lien vers 'learn more'. Ce lien cliquable fait apparaitre une modale avec la photo, la description et deux liens qui mènent vers le site web et lien GitHub du projet.

DOSSIER PROFESSIONNEL (DP)

Version Desktop



Version téléphone portable avec ou sans le menu burger ouvert



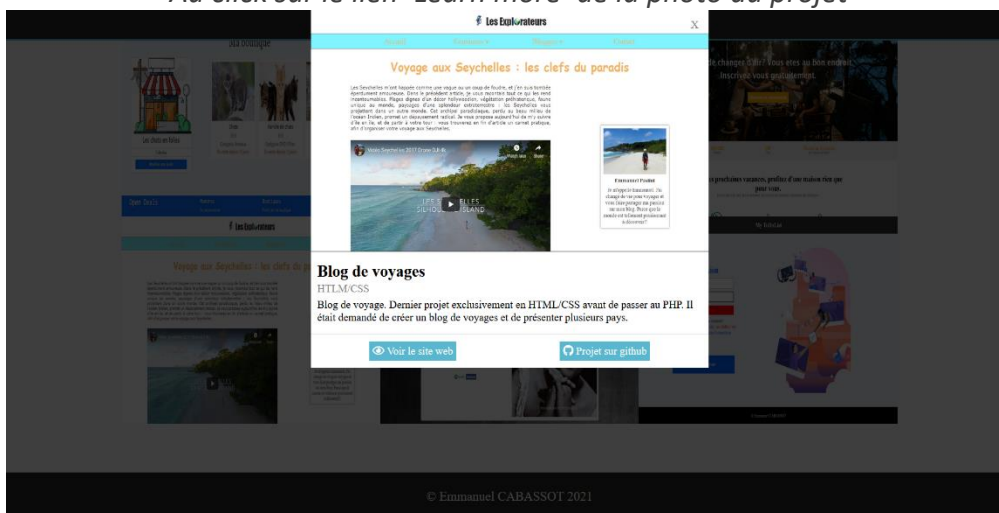
Exemple de Media queries (navbar)

```
/* MEDIA QUERIES */
@media screen and (min-width: 768px){
  .nav-button{
    display: none;
  }
  .navbar{
    flex-direction: row;
    justify-content: center;
    border-bottom: 3px solid var(--main-color);
  }
  .nav-menu{
    display: flex;
    justify-content: center;
    position: static;
    background-color: unset;
  }
  .nav-item{
    display: flex;
  }
}
```

DOSSIER PROFESSIONNEL (DP)

```
width: 150px;
flex-direction: row;
line-height: 42.2px;
height: 100%;
padding: 0 10px;
}
}
```

Au click sur le lien 'Learn more' de la photo du projet



- ✓ J'ai réalisé ce portfolio grâce à l'**HTML, CSS, JavaScript** et **PHP** pour la structure du site ainsi que le design.
- ✓ le menu burger est créé avec de l'**HTML, CSS, JavaScript**. Il disparaît en dessous de 768px grâce aux Médiastyles. La liste déroulante apparaît grâce à du JavaScript (toggle).
- ✓ Il y a beaucoup d'animations déclenchées au scroll grâce à l'objet JavaScript `IntersectionObserver()`.
- ✓ La partie SEO a été faite. Taper Emmanuel Cabassot pour trouver le portfolio.

Ce projet m'a permis d'acquérir les compétences du référentiel :

- ✓ **Maquetter une application**
- ✓ **Réaliser une interface utilisateur web statique et adaptable**

DOSSIER PROFESSIONNEL ^(DP)

2. Précisez les moyens utilisés :

- ✓ **Visual Studio Code** pour la rédaction et l'édition de code
- ✓ **Mockup** pour les wireframes
- ✓ **Tortoise Git et Github** pour le versionning
- ✓ **HTML / CSS/ Javascript** pour l'intégration
- ✓ **PHP** pour récupérer les images et les informations des projets
- ✓ **Médias Queries/ Flexbox /Javascript** pour le responsive design

3. Avec qui avez-vous travaillé ?

J'ai réalisé le projet seul.

4. Contexte

Nom de l'entreprise, organisme ou association ► *Emmanuel Cabassot*

Chantier, atelier, service ► Portfolio

Période d'exercice ► Du : *5 Avril* au : *12 Avril*

5. Informations complémentaires (facultatif)

Activité-type 1

Développer la partie front-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n° 2 ► ToDoList


1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Réalisation d'une ToDoList.

Le point important est que le site est créé de façon à être dynamique grâce à l'utilisation de JavaScript et des requêtes AJAX.

- ✓ Le formulaire de connexion demande un mot de passe avec plusieurs types de caractères.
- ✓ Grâce au JavaScript et à `addEventListener` ('input') on écoute le champ pour savoir si il y a bien une majuscule, une minuscule..... Il y a aussi un `addEventListener` qui écoute le champs e-mail.
- ✓ Quand les champs sont au bon format le bouton 'valider' devient bleu et une requête ajax (fetch) est envoyée vers un traitement en **PHP** qui interroge la BDD.
- ✓ Le clique sur le bouton inscription ouvre une modale avec le formulaire d'inscription. Le fonctionnement est le même que pour la connexion.

Ecran de connexion



On écoute l'input du mot de passe

```
// On écoute l'input du mot de passe
password.addEventListener("input", checkPass)
```

DOSSIER PROFESSIONNEL (DP)

Fonction checkPass() qui vérifie les caractères dans l'input password grâce aux regex

```
function checkPass() {
  // On initialise un score
  let score = '';

  // On récupère ce qui a été saisi
  let mdp = password.value;

  // On va chercher les éléments dont on a besoin
  let minuscule = document.querySelector('.connexion #minuscule');
  let majuscule = document.querySelector('.connexion #majuscule');
  let chiffre = document.querySelector('.connexion #chiffre');
  let special = document.querySelector('.connexion #special');
  let longueur = document.querySelector('.connexion #longueur');

  // On vérifie qu'on a une minuscule
  if (/[a-z]/.test(mdp)) {
    // On passe en bleu minuscule (valid)
    minuscule.classList.replace("invalid", "valid");
    score += 'a';
  } else {
    // On passe en rouge minuscule (invalid)
    minuscule.classList.replace('valid', 'invalid');
  }

  // On vérifie qu'on a une majuscule
  if (/[A-Z]/.test(mdp)) {
    // On passe en bleu (valid)
    majuscule.classList.replace('invalid', 'valid');
    score += 'b';
  } else {
    // On passe en rouge (invalid)
    majuscule.classList.replace('valid', 'invalid');
  }

  // On vérifie qu'on a un chiffre
  if (/[0-9]/.test(mdp)) {
    // On passe en bleu (valid)
    chiffre.classList.replace('invalid', 'valid');
    score += 'c';
  } else {
    // On passe en rouge (invalid)
    chiffre.classList.replace('valid', 'invalid');
  }
}
```

```
// On vérifie qu'on a un caractère spécial
if (/[!@!/\%*#&]/.test(mdp)) {
    // On passe en bleu (valid)
    special.classList.replace('invalid', 'valid');
    score += 'd';
} else {
    // On passe en rouge (invalid)
    special.classList.replace('valid', 'invalid');
}

// On vérifie la longueur du mdp
if (mdp.length >= 6) {
    // On passe en bleu
    longueur.classList.replace('invalid', 'valid');
    score += 'e';
} else {
    verif = score.includes("e")

    // On passe en rouge
    longueur.classList.replace('valid', 'invalid')
}
```

Si les conditions du password sont ok on vérifie que les conditions de l'email soient ok pour passer le bouton valider en bleu et enlever le disabled.

```
// Si les conditions du password sont ok on vérifie que les conditions de l'email
soient ok pour passer le bouton valider en bleu et enlever le disabled.
if (score == 'abcde') {
    passwordCok = true;
    if (emailCok == true) {
        button.disabled = false
    }
} else {
    passwordCok = false
    button.disabled = true
}
}
```

A la suite des vérifications du password et de l'email (checkMail) on envoie une requête fetch() à la page de traitement. La page de traitement revérifie les conditions du password et mail et envoie la requête en BDD. Si il y a des erreurs un message est renvoyé. Si tout est ok on bascule vers sa ToDoList. Tout cela de façon dynamique.

DOSSIER PROFESSIONNEL (DP)

La page des listes et tâches est totalement dynamique sans rechargement.

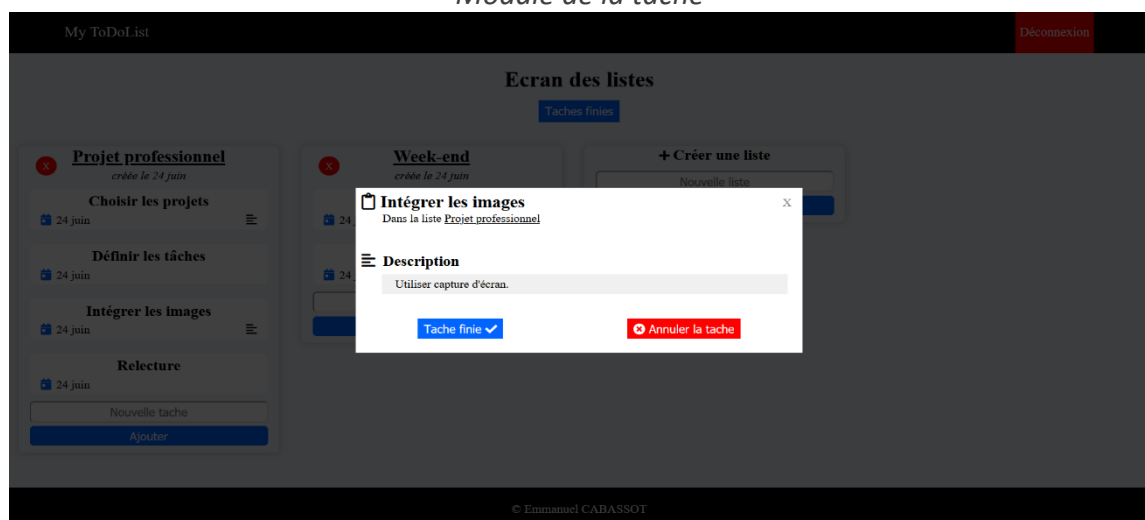
- ✓ On ajoute à la volée des listes, des tâches.
- ✓ Au clic sur Tâches finies l'on accède aux taches finies englobées dans leur liste d'origine.

Page des listes et des tâches en cours



- ✓ Au clique sur une tâche une modale s'ouvre avec la possibilité de marquer une description, d'annuler la tâche ou de la marquer comme finie

Modale de la tâche



DOSSIER PROFESSIONNEL ^(DP)

Ce projet m'a permis d'acquérir la compétence du référentiel :

- ✓ **Développer une interface utilisateur web dynamique**

2. Précisez les moyens utilisés :

- ✓ Requêtes **AJAX** grâce à l'objet **fetch()**.
- ✓ **PHP** pour le traitement et l'enregistrement en BDD.
- ✓ Utilisation des objets : **addEventListener()**, **Date()** ...
- ✓ **Visual Studio Code**.
- ✓ **Visual Studio Code** pour la rédaction et l'édition de code.
- ✓ **Tortoise Git et Github** pour le versionning.
- ✓ **HTML / CSS/ Javascript** pour l'intégration.

3. Avec qui avez-vous travaillé ?

J'ai travaillé seul sur ce projet

4. Contexte

Nom de l'entreprise, organisme ou association ► *LaPlateforme_*

Chantier, atelier, service ►

Période d'exercice ► Du :*15 avril* au :*28 avril*

5. Informations complémentaires (facultatif)

Activité-type 3 Développer la partie back-end d'une application web ou web mobile en intégrant les recommandations de sécurité

Exemple n° 1 ► Boutique de boutiques

1. Décrivez les tâches ou opérations que vous avez effectuées, et dans quelles conditions :

Réalisation d'une boutique de boutiques qui permet à un utilisateur d'acheter des articles.

L'utilisateur peut créer sa boutique pour y vendre des articles juste en confirmant son adresse et son nom.

Un professionnel crée sa boutique grâce aux informations de sa société et son numéro SIRET.

Une fois la boutique de pro ou de particulier créée, le membre se retrouve avec **une liste déroulante 'boutique'** depuis laquelle il peut :

- ajouter des articles
- modifier ou supprimer ses articles
- modifier les informations de sa boutique
- modifier la photo de présentation de sa boutique
- un page de présentation de la boutique est visitable par tout utilisateur ce qui lui permet d'avoir des informations sur les articles vendus par la boutique depuis quand a été créée la boutique, boutique pro ou particulier...

List déroulante user ('profil') depuis laquelle un user qu'il ait une boutique ou non:

- voir son profil
- modifier sa photo, son adresse
- voir la liste de ses commandes

Une page d'accueil avec une barre de recherche et les derniers articles mis en ligne.

L'idée était de faire un site avec une inscription facile qui permet de passer d'un statut d'utilisateur sans boutique et donc acheteur à utilisateur ayant une boutique très facilement.

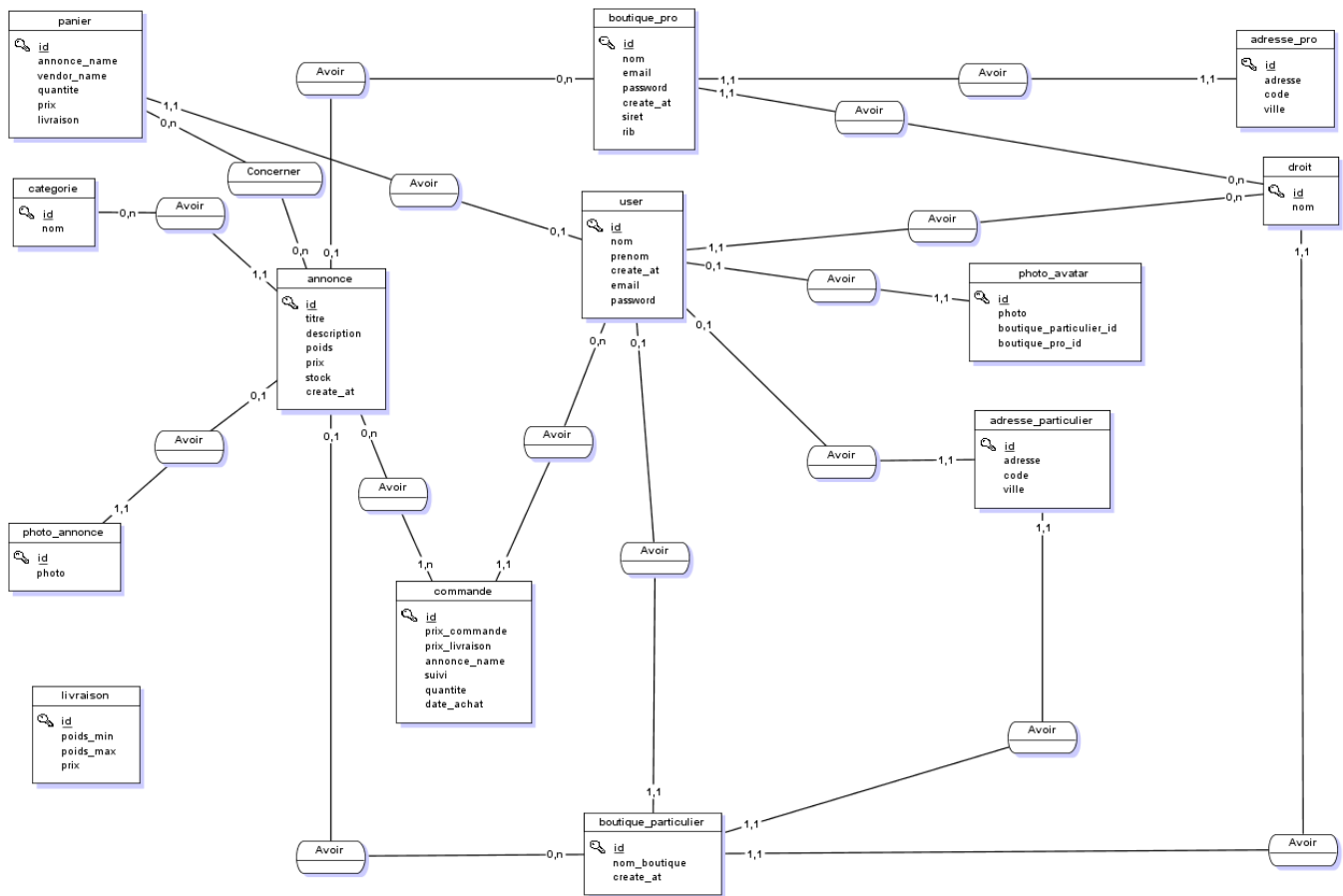
Il n'y a pas d'échange d'argent entre le vendeur et l'acheteur puisque l'acheteur nous paie et une fois la livraison effectuée c'est le site qui paie le vendeur.

Ce projet a été fait avec le design pattern MVC ce qui a entraîné l'utilisation d'un router, de la POO avec les namespaces et un autoloader 'maison'.

Créer une base de données

Pour notre projet je me suis occupé de la création du MCD.

J'ai utilisé Jmerise pour créer le MCD.



Architecture du projet

Dans le but d'une maintenabilité du site et d'un code propre, nous avons donc opté pour le design pattern MVC.

A la racine du projet nous avons donc :

- dossier Controller : les controllers.
- dossier Core :

- dossier functions :
- dossier Models
- dossier Public
- dossier Views
- fichier .htaccess
- fichier Autoloader.php
- fichier index.php
- Sont venus s'ajouter d'autres éléments pour l'API Stripe.

Développer les composants d'accès aux données

Pour faire des requêtes à la BDD on passe par 3 niveaux.

-La **connexion à la BDD** grâce à la **classe Db** qui hérite de PDO.

-La **classe Model** qui **hérite** de la **classe Db**. Elle contient des **méthodes** telles que **prepare, hydrate...**

-Les classes étant dans le dossier Models qui sont **associées aux tables de la BDD**. Elles contiennent les attributs de leur table associée, des méthodes servants à faire des requêtes, des getters et setters pour les attributs.

Ces classes héritent de la classe Model.

Création de la classe Db qui extends de PDO et va faire la connexion à la BDD.

Boutique/Core/Db.php

```
namespace App\Core;

// On "importe" PDO
use PDO;
use PDOException;

class Db extends PDO
{
    // Instance unique de la classe
    private static $instance;

    // Informations de connexion
    private const DBHOST = 'localhost:3306';
    private const DBUSER = 'emmanuel';
    private const DBPASS = '*****';
    private const DBNAME = 'emmanuel-cabassot_boutique';
```

Constructeur de la classe Db qui utilise les constantes et fait appel au constructeur de la classe PDO.

```
private function __construct()
{
    // DSN de connexion
    $_dsn = 'mysql:dbname='. self::DBNAME . ';charset=utf8; host=' . self::DBHOST;

    // On appelle le constructeur de la classe PDO
    try{
        parent::__construct($_dsn, self::DBUSER, self::DBPASS);

        $this->setAttribute(PDO::MYSQL_ATTR_INIT_COMMAND, 'SET NAMES utf8');
        $this->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_OBJ);
        $this->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }catch(PDOException $e){
        die($e->getMessage());
    }
}
```

Permet de ne pas recréer l'instance si celle-ci existe déjà.

```
public static function getInstance():self
{
    if(self::$instance === null){
        self::$instance = new self();
    }
    return self::$instance;
}
```

La classe Model qui extends de Db

```
namespace App\Models;

use App\Core\Db;

class Model extends Db
{
    // Table de la base de données
    protected $table;

    // Instance de Db
    private $db;
```

Quelques méthodes de la classe Model :

- requete(),
- create(),
- update(),
- delete()
- hydrate().
- findBy()
- find()

Méthode requete

```
public function requete(string $sql, array $attributs = null)
{
    // On récupère l'instance de Db
    $this->db = Db::getInstance();

    // On vérifie si on a des attributs
    if ($attributs !== null)
    {
        // Requête préparée
        $query = $this->db->prepare($sql);
        $query->execute($attributs);
        return $query;
    }
    else
    {
        // Requête simple
        return $this->db->query($sql);
    }
}
```

Méthode create

```
public function create()
{
    $champs = [];
    $inter = [];
    $valeurs = [];

    // On boucle pour éclater le tableau
    foreach ($this as $champ => $valeur) {
        // INSERT INTO annonces (titre, description, actif) VALUES (?, ?, ?)
    }
}
```

```
        if ($valeur !== null && $champ != 'db' && $champ != 'table') {
            $champs[] = $champ;
            $inter[] = "?";
            $valeurs[] = $valeur;
        }
    }

    // On transforme le tableau "champs" en une chaîne de caractères
    $liste_champs = implode(',', $champs);
    $liste_inter = implode(',', $inter);

    // On exécute la requête
    return $this->requete('INSERT INTO ' . $this->table . ' (' . $liste_champs . ')VALUES(' . $liste_inter . ')', $valeurs);
}
```

Méthode update

```
public function update()
{
    $champs = [];
    $valeurs = [];

    // On boucle pour éclater le tableau
    foreach ($this as $champ => $valeur) {
        // UPDATE annonces SET titre = ?, description = ?, actif = ? WHERE id= ?
        if ($valeur !== null && $champ != 'db' && $champ != 'table') {
            $champs[] = "$champ = ?";
            $valeurs[] = $valeur;
        }
    }
    $valeurs[] = $this->id;

    // On transforme le tableau "champs" en une chaîne de caractères
    $liste_champs = implode(',', $champs);

    // On exécute la requête
    return $this->requete('UPDATE ' . $this->table . ' SET ' . $liste_champs . ' WHERE id = ?', $valeurs);
}
```

Méthode hydrate

```
public function hydrate($donnees)
{
    foreach ($donnees as $key => $value) {
        // On récupère le nom du setter correspondant à la clé (key)
        // titre -> setTitre
        $setter = 'set' . ucfirst($key);

        // On vérifie si le setter existe
        if (method_exists($this, $setter)) {
            // On appelle le setter
            $this->$setter($value);
        }
    }
    return $this;
}
```

Des classes sont associées aux tables, elles héritent de la classe Model.php.

Classe AdresseParticulierModel.php

```
namespace App\Models;

class AdresseParticulierModel extends Model
{
    protected $id;
    protected $user_id;
    protected $adresse;
    protected $code;
    protected $ville;
```

Constructeur pour intégrer le nom de la table en BDD.

```
public function __construct()
{
    $this->table = 'adresse_particulier';
}
```

Les setters et getters

```
/**
 * Get the value of id
 */
public function getId()
{
    return $this->id;
}

/**
 * Set the value of id
 *
 * @return self
 */
public function setId($id)
{
    $this->id = $id;

    return $this;
}
```

Exemple de méthode dans la classe

```
public function findAdresse($user_id)
{
    return $this->requete("SELECT * FROM $this->table WHERE user_id = $user_id")->fetch();
}
```

Exemple d'utilisation des Models dans les Controllers

```
$boutique = new BoutiqueProModel;
$boutique = $boutique->find($boutique_pro_id);

$photo = new PhotoAvatarModel;
$photo = $photo->findPhotoBoutique($boutique->id);

$adresse = new AdresseProModel;
$adresse = $adresse->findAdresse($boutique->id);

$annonce = new AnnonceModel;
$annonce = $annonce->findAnnoneProLimit($boutique->id);
```

Publier une annonce

Développer la partie back-end d'une application web ou web mobile

Etant dans le design pattern MVC c'est grâce aux méthodes des Controllers que nous traitons les données.

Traitement des informations remplies par l'utilisateur

Fonction ajouterPar

```
public function ajouterPar()  
{
```

Méthode statique valide que le formulaire est complet

```
// On vérifie que le formulaire est complet  
if (Form::validate($_POST, ['titre', 'description', 'prix', 'stock'])) {
```

Sécurise les données récupérées grâce à strip_tags() et le typage des variables

```
// Le formulaire est complet  
// On se protège des failles xss  
// On type les variables  
$titre = strip_tags($_POST['titre']);  
$description = strip_tags($_POST['description']);  
$prix = (int) $_POST['prix'];  
$poids = (int) $_POST['poids'];  
$stock = (int) $_POST['stock'];
```

On instancie les classes nécessaires et on hydrate l'objet pour créer l'objet l'annonce en BDD

```
// On instancie notre modele et la classe boutique_par  
$annonce = new AnnonceModel;  
$boutique = new BoutiqueParticulierModel;  
$boutique = $boutique->findBoutiqueByUser($_SESSION['user']['id']);  
  
// On hydrate l'objet  
$annonce->setTitre($titre)  
->setCategorie_id($_POST['categorie'])  
->setDescription($description)
```

DOSSIER PROFESSIONNEL (DP)

```
->setPrix($prix)
->setPoids($poids)
->setStock($stock)
->setBoutique_particulier_id($boutique->id);

// On crée l'annonce en BDD
$annonce->create();

// On retour chercher l'annonce créée pour obtenir son id
$annonce = $annonce->findAnnonceByPar($boutique->id);
```

Traitement de la photo de l'annonce

```
// On instancie la classe photo
$photo = new PhotoAnnonceModel;
```

Vérification que la taille ne soit pas > à 2mo et tableau des extensions valides

```
//Taille max de la photo
$tailleMax = 2000000;

// Extensions valides pour la photo
$extensionValides = ['jpg', 'jpeg', 'gif', 'png'];
```

Si ok : \$extensionUpload récupère le nom de la photo, la convertit en minuscule, récupère l'extension après le point.

```
if ($_FILES['photo_principale']['size'] <= $tailleMax) {
    // La taille du fichier est bien inférieure à ce que l'on demande
    // On vérifie l'extension
    $extensionUpload =
    strtolower(substr(strrchr($_FILES['photo_principale']['name'], '.'), 1));
```

Si l'extension est valide l'on enregistre l'image dans le public/img/annonce/id_annonce.extension

```
if (in_array($extensionUpload, $extensionValides)) {
    // La taille et l'extension de la photo sont valides

    // Chemin et nom du fichier que l'on va enregistrer
    $chemin = 'public/img/annonce/' . $annonce->id . '.' . $extensionUpload;
```


DOSSIER PROFESSIONNEL (DP)

```
// On enregistre le fichier grace à move et $resultat = false ou true
$resultat = move_uploaded_file($_FILES['photo_principale']['tmp_name'], $chemin)
```

Si la photo à bien été déplacée l'on enregistre l'image en BDD

```
if ($resultat) {
    // On hydrate l'objet
    $photo->setAnnonce_id($annonce->id)
    ->setPhoto($annonce->id . '.' . $extensionUpload);
    // On crée insert la photo en BDD
    $photo->create();
}
```

Messages d'erreurs s'il y a eu un problème en cours de processus avec redirection

```
} else {
    $_SESSION['erreur'] = "Erreur durant l'importation du fichier";
    header('location: ' . ACCUEIL . 'annonce/ajouterpar');
    exit;
}
} else {
    $_SESSION['erreur'] = "Votre photo doit être au
format jpg, jpeg, gif ou png";
    header('location: ' . ACCUEIL . 'annonce/ajouterpar');
    exit;
}
} else {
    $_SESSION['erreur'] = "Votre photo ne doit pas dépasser 2 mo";
    header('location: ' . ACCUEIL . 'annonce/ajouterpar');
    exit;
}
}
```

Si tout est ok message de succès et redirection vers l'accueil de la boutique de l'utilisateur

```
// On redirige
$_SESSION['success'] = "votre annonce à été enregistrée avec succès";
header('location: ' . ACCUEIL . 'boutiqueAccueil/accueilPar');
}
```

DOSSIER PROFESSIONNEL ^(DP)

Ce projet m'a permis d'acquérir la compétence du référentiel :

- ✓ **Créer une base de données**
- ✓ **Développer les composants d'accès aux données**
- ✓ **Développer la partie back-end d'une application web ou web mobile**

2. Précisez les moyens utilisés :

- ✓ Design pattern **MVC**.
- ✓ Utilisation des **namespaces**, **autoloder** et fichier **htaccess**.
- ✓ **JMerise** pour le MCD.
- ✓ **Visual Studio Code** pour la rédaction et l'édition de code.
- ✓ **Tortoise Git** et **Github** pour le versionning.
- ✓ **HTML / CSS/ Javascript** pour l'intégration.

3. Avec qui avez-vous travaillé ?

Mathias Tavernier-roux

4. Contexte

Nom de l'entreprise, organisme ou association ► *LaPlateforme_*

Chantier, atelier, service ►

Période d'exercice ► Du : *3 février* au : *23 février*

5. Informations complémentaires (facultatif)

DOSSIER PROFESSIONNEL ^(DP)

Titres, diplômes, CQP, attestations de formation

(facultatif)

Intitulé	Autorité ou organisme	Date
Cliquez ici.	Cliquez ici pour taper du texte.	Cliquez ici pour sélectionner une date.

DOSSIER PROFESSIONNEL ^(DP)

Déclaration sur l'honneur

Je soussigné(e) [prénom et nom] **Emmanuel Cabassot** ,
déclare sur l'honneur que les renseignements fournis dans ce dossier sont exacts et que je suis
l'auteur(e) des réalisations jointes.

Fait à **Marseille** le **2 Juillet 2021**

pour faire valoir ce que de droit.

Signature : Emmanuel Cabassot

DOSSIER PROFESSIONNEL ^(DP)

Documents illustrant la pratique professionnelle

(facultatif)

Intitulé
Cliquez ici pour taper du texte.

DOSSIER PROFESSIONNEL ^(DP)

ANNEXES

(Si le RC le prévoit)