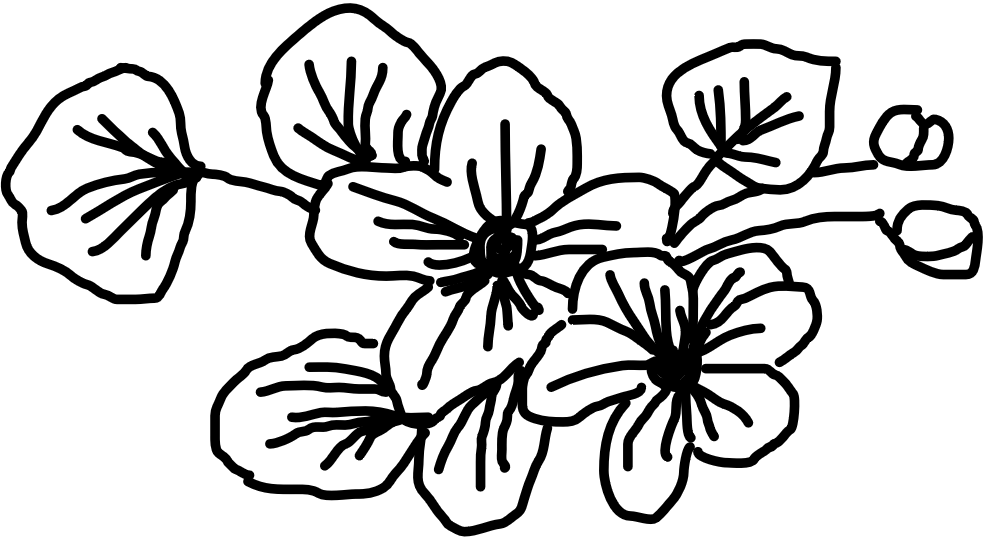
DOSSIER DE PROJET



LAFLEUR

DÉVELOPPEUR WEB - WEB MOBILE

2022 - 2023

ELSA THIÉVET

TABLE DES MATIERES

[1. Les compétences couvertes par le projet 3](#_Toc134266036)

[1.1 Les compétences 3](#_Toc134266037)

[1.2 Résumé du projet 4](#_Toc134266038)

[2. Cahier des charges du projet 4](#_Toc134266039)

[2.1 Les missions et les spécifications fonctionnelles 4](#_Toc134266040)

[2.2 Spécifications techniques 5](#_Toc134266041)

[2.3 Contraintes imposées par le projet 6](#_Toc134266042)

[3. Concevoir une application esthétique et fonctionnelle 7](#_Toc134266043)

[3.1 Créer la structure du site 7](#_Toc134266044)

[3.2 L’identité graphique du site 8](#_Toc134266045)

[3.3 Concevoir la maquette du site 9](#_Toc134266046)

[3.4 Accès au compte et au panier 10](#_Toc134266047)

[3.5 Version mobile 11](#_Toc134266048)

[4. Concevoir la base de données 11](#_Toc134266049)

[4.1 Les entités et les relations 11](#_Toc134266050)

[4.2 Typer les données 12](#_Toc134266051)

[5. Développer la partie frontend de la boutique en ligne 13](#_Toc134266052)

[5.1 Structuration et mise en page d’une page 13](#_Toc134266053)

[5.2 Contenu dynamique 15](#_Toc134266054)

[6. Développer la partie backend de l’application 18](#_Toc134266055)

[6.1 L’architecture MVC 18](#_Toc134266056)

[6.2 La classe PDO 19](#_Toc134266057)

[6.3 Cas pratique : afficher telles infos, extraits de code modèle contrôleur et vue 20](#_Toc134266058)

[7 Gestion de stock 20](#_Toc134266059)

[7.2 Installer Symfony 20](#_Toc134266060)

[7.3 La sécurité 20](#_Toc134266061)

[8 Blog 20](#_Toc134266062)

[8.2 Créer un blog WordPress 20](#_Toc134266063)

[8.3 Coder un plugin 20](#_Toc134266064)

[9 Présentation du jeu d’essai de la fonctionnalité la + représentative (données en entrée, données attendues et données obtenues) 20](#_Toc134266065)

[10 Veille sur les vulnérabilités de sécurité 20](#_Toc134266066)

[11 Description de la situation ayant nécessité une recherche 21](#_Toc134266067)

[12 Resource trouvée et sa traduction (750 signes environ) 21](#_Toc134266068)

[Glossaire 22](#_Toc134266069)

7 veille sur les vulnérabilités de sécurité

Description des risques de sécurité

Htaccess

Validation en php

Nom des fichiers enregistrés

8 description d’une situation de travail ayant nécessité une recherche à partir de site anglophone

9 extrait du site anglophone

Extrait du site

Traduction de l’extrait en français

Conclusion

# Les compétences couvertes par le projet

## 1.1 Les compétences

Le projet Lafleur que je vais présenter couvre l’intégralité des compétences du référentiel. Ces compétences sont :

Activité type 1 : Développer la partie frontend d’une application web ou web mobile en intégrant les recommandations de sécurité

* Maquetter une application
* Réaliser une interface utilisateur web statique et adaptable
* Développer une interface utilisateur web dynamique
* Réaliser une interface utilisateur avec une solution de gestion de contenu ou e-commerce

Activité type 2 : Développer la partie backend d’une application web ou web mobile en intégrant les recommandations de sécurité

* Créer une base de données
* Développer les composants d’accès aux données
* Développer la partie backend d’une application web ou mobile
* Elaborer et mettre en œuvre des composants dans une application de gestion de contenu ou e-commerce

## 1.2 Résumé du projet

J’ai été contactée par un couple de fleuristes indépendants – Sabine et Guillaume Chollet - qui ont actuellement une boutique à Lourmarin dans la région du Luberon. Ils souhaitent étendre leur commerce à la vente en ligne dans le but de se moderniser et développer leur présence sur la toile. Sur leur futur site, ils veulent vendre leurs créations ainsi que des fleurs à l’unité (c’est la particularité de leur commerce).

Cela impliquerait également la possibilité de gérer leur stock en ligne pour plus de praticité. En effet, cela leur permettra de voir les articles bientôt épuisés, pour pouvoir s’occuper du réassort, ainsi que les commandes à honorer. Concernant ces dernières, ils précisent qu’ils ne peuvent livrer que dans un certain périmètre autour de leur boutique à Lourmarin.

Le couple a deux demandes supplémentaires :

Premièrement, M. Chollet est photographe de flore et de nature et souhaite publier régulièrement ses œuvres dans des articles sur un blog. En outre, ce blog contiendra des références à la boutique en ligne, et la boutique en ligne renverra également vers le blog.

Deuxièmement, le couple souhaite pour l’occasion de la fête des mères organiser un jeu en ligne. L’évènement sera une loterie et durera deux semaines. Chaque client qui effectuera un achat en ligne sur leur site aura la chance de pouvoir participer à cette loterie.

# Cahier des charges du projet

## 2.1 Les missions et les spécifications fonctionnelles

1ère mission : développer une boutique en ligne

Elle devra permettre au couple de vendre ses bouquets et fleurs à l’unité comme sur un **site de vente classique**.

Les clients pourront rechercher les produits **par couleur et par catégories**. Ces catégories sont : remerciements, naissance, mariage, amour et sentiments, anniversaire.

Lorsqu’un produit sera en **rupture de stock**, le produit restera affiché mais ne sera pas commandable.

Concernant les **frais de port**, ils seront de 2,99€ pour les commandes de moins de 50€, gratuits au-delà. Le délai de livraison est d’un jour, sauf si le client souhaite une date ultérieure, qu’il peut préciser, c’est une **livraison programmée**. La boutique ne livre que dans un **certain périmètre**, et si un client n’habite pas assez près il sera prévenu de l’impossibilité de commander.

2ème mission : gestion de stock

M. et Mme Chollet devront pouvoir accéder de **façon sécurisée** à une application de gestion de stock où ils pourront saisir les quantités des **articles rentrés en magasin** et recevoir des **alertes** lorsque ceux-ci approchent de la rupture de stock.

Ils pourront également ajouter de nouvelles **références d’articles** et voir les **commandes en cours** et passées pour pouvoir gérer leslivraisons.

3ème mission : blog

M. Chollet souhaite publier des articles sur un **blog**. Ces articles parleront de sa région, le Luberon, et il y publiera par ailleurs de temps en temps des articles qui contiendront des **liens vers son magasin en ligne**. Réciproquement, le magasin en ligne contiendra des références et **liens vers le blog** de M. Chollet. En outre, on implémentera un plugin pour donner un peu de relief aux mots qu’il voudra mettre en avant. Cela améliorera le positionnement du blog sur les résultats des moteurs de recherche.

4ème mission : loterie

Pour l’occasion de la fête des mères, Mme Chollet veut créer un jeu en ligne. Pendant deux semaines, chaque commande sera l’occasion de participer à la **loterie** et de tenter de remporter un lot. Le jeu ressemble à une **machine à sous** et la chance de gagner est **proportionnelle** à la quantité de chaque lot.

## 2.2 Spécifications techniques

L’arborescence de l’application a été pensée sur l’outil de diagramme en ligne **draw.io**.

La maquette et la charte graphique ont été conçues sur le programme de prototypage **Figma**.

Le développement statique et *responsive* du projet s’est effectué sur l’IDE **VS Code** avec les langages **HTML** et **CSS**.

Le langage **JavaScript (ES6)** a également été utilisé pour une interface dynamique et des composants en langage **PHP** pour communiquer avec la base de données, le tout avec un *design pattern* **MVC** (modèle-vue-contrôleur).

Le Modèle Logique de Données (MLD) a été conçu sur le logiciel **Workbench** qui m’a permis de générer un script **SQL** que j’ai exploité conjointement sur Workbench et le système de gestion de base de données (SGBDR) en ligne **phpMyAdmin**.

La partie backend de l’application a été développée sur le *framework* PHP **Symfony** qui a nécessité l’installation de **Composer** (gestionnaire de dépendances), **Yarn** et contient le moteur de templates **Twig.**

Le blog a été conçu sur le CMS (système de gestion de contenu) **WordPress**. On lui a ajouté un *plugin* pour améliorer son référencement (son positionnement et sa visibilité de sites dans les résultats de moteurs de recherche)

Enfin, on peut mentionner **Git** et **GitHub** pour le versionnage du projet et Trello pour l’organisation et le suivi des tâches à effectuer pour avancer.

## 2.3 Contraintes imposées par le projet

En ce qui concerne la boutique en ligne :

* La maquette devra être validée par les clients.
* L’application devra respecter le design pattern MVC.
* Développement en PHP pur sans framework.
* Le site devra passer les validations W3C.
* Le site devra être adaptable à tous les formats d’écran.

Pour la gestion de stock :

* Utilisation d’un framework PHP
* La base de données du gestionnaire de stock doit être la même que la base de données nécessaire à la boutique en ligne

Blog Wordpress :

* Utilisation de WordPress
* Modification d’un thème existant pour correspondre aux désirs des clients  
  Intégration d’un plugin
* Créer un persona pour “Guillaume”, auteur des articles, et fan de nature et  
  de photographie.

Loterie :

* Le jeu doit être réalisé en HTML, CSS et animé en JavaScript.
* La chance de gagner un lot doit être proportionnelle à la quantité de lots à  
  gagner. Il doit être plus facile de gagner un porte-clés qu’un bouquet de roses.

**Elaborées par le candidat y compris pr la securité et le web mobile (cf PDF lafleur)**

**Sécurité : regex, confirm mdp, csrf, requetes preparees pour éviter quoi ?**

**Cahier des charges = l’arborescence du site, accueil : devra comporter…**

# Concevoir une application esthétique et fonctionnelle

## 3.1 Créer la structure du site

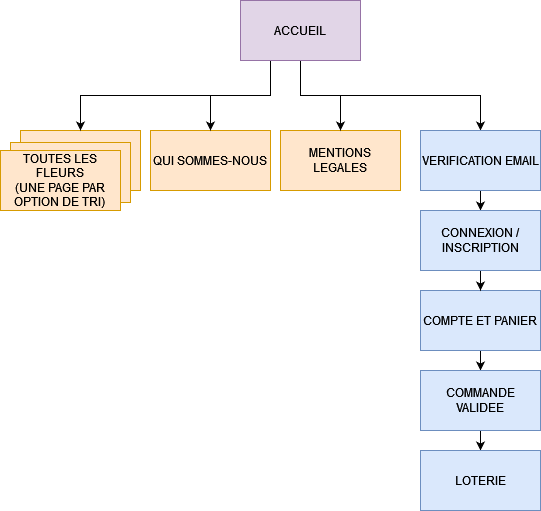
Dans un premier temps et avant toute chose, il faut penser le site en termes de rendu, pour s’assurer que les clients ont été compris dans leur demande.   
Pour cela on va commencer par penser la structure du site et construire une arborescence cohérente avec les désirs des clients et avec les pratiques courantes en termes de site de e-commerce. En effet, les utilisateurs qui viendront voir le site de Lafleur auront surement des habitudes de navigation et d’utilisation de ce genre de site. On choisit donc de respecter les bonnes pratiques en ayant une page d’accueil, puis une page par thème essentiel. Ici il s’agira évidemment des articles que propose la boutique, puis des articles qui sont des roses, puis des fleurs à l’unité. Également en thème important, la page « qui sommes-nous » qui donne un aperçu de l’histoire et les coordonnées du magasin de fleurs.

Inévitablement, on va retrouver une page « compte » qui regroupera les informations de l’utilisateur à la condition qu’il soit connecté (via le formulaire préalable), comprenant aussi le contenu de son panier et ses commandes passées.

Si c’est la période de l’évènement fête des mères, l’utilisateur y aura accès après la validation d’une commande.

On prendra soin d’ajouter des informations annexes mais incontournables : les mentions légales.

Ce qu’on ne retrouvera pas à proprement parler dans l’arborescence mais qui sera quand même mentionné sur l’application : un lien vers le blog WordPress de Guillaume Chollet en pied-de-page, ainsi que des liens vers les réseaux sociaux de la boutique.



L’arborescence de la boutique en ligne

## 3.2 L’identité graphique du site

Premièrement, il va falloir trouver une identité graphique qui correspond à notre couple de fleuristes, ainsi qu’à une boutique de fleurs en général. Pour la palette, on sélectionne des rouges qui rappellent le velours, les roses rouges et l’amour. Le rouge sera un peu plus vif pour le logo pour le faire ressortir, et pour le reste du site (essentiellement la bannière au-dessus de l’en-tête et le pied-de-page), on choisira un rouge un peu plus éteint et moins tonique. En couleurs neutres, pour certains fonds, on choisira du blanc et du crème, ainsi que du noir pour les textes et les boutons d’actions.

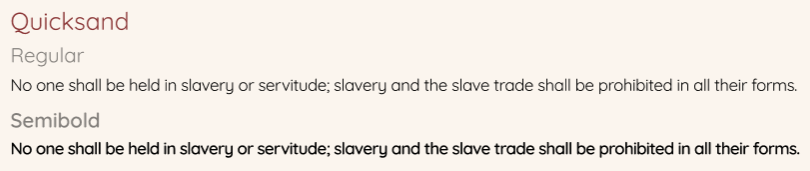
Les couleurs de la palette graphique

Le logo se veut chic et intemporel. On lui choisit une police à empâtements et à l’aspect Art Nouveau pour un côté floral, comme l’était ce mouvement architectural et des arts décoratifs qui a puisé son inspiration dans la nature et les formes organiques.



La police Lancelot, utilisée pour le logo et les titres du site

Pour les textes, on utilisera Quicksand, une police simple et lisible dans un style actuel. Elle se fait discrète tout en étant facile à lire.



La police choisie pour la plupart des textes du site

## 3.3 Concevoir la maquette du site

On retrouve d’abord sur tout le site un en-tête et un pied-de-page qui résument le site.  
L’en-tête indique à l’utilisateur qu’il est bien arrivé chez Lafleur, boutique à Lourmarin. On remarque un bandeau tout en haut qui nous rappelle que les commandes supérieures à 50€ sont livrées gratuitement.

Sur ce site, il sait ce qu’il va pouvoir consulter via la barre de navigation : l’accueil, une page avec tous les articles, une page avec les produits phares que sont les roses et une page où il verra les fleurs à l’unité. Il constatera également une page « qui sommes-nous » qui lui présentera le couple de fleuristes et indiquera leurs coordonnées.

Au bout de la barre de navigation, il verra deux pictogrammes qui lui indiquent qu’il est possible d’avoir un compte utilisateur et un panier.

Au niveau du pied-de-page, on aura tout d’abord un rappel du logo de Lafleur, avec les coordonnées du magasin. Ensuite, on montrera une structure du site un peu différentes : elle reprendra entièrement les liens de la barre de navigation mais on ajoutera la liste des évènements : naissance, mariage, remerciements, amour & sentiments, anniversaire. On n’oubliera pas les mentions légales ainsi qu’un lien vers les informations de livraison.

En ce qui concerne les autres plateformes utilisées par nos clients fleuristes, on mentionnera les réseaux sociaux et le blog de M. Chollet, toujours dans le pied-de-page.

L’en-tête et le pied-de-page ont bien sûr vocation à rester communs à toutes les pages du site.

Pour ce qui est des pages, on retrouve d’abord l’accueil. Il présentera un carrousel pour mettre en avant l’actualité de la boutique. Il crée de l’animation et du dynamisme sur le site.

Juste en dessous, il y aura des *cards* pour chaque évènement où on offre des fleurs.

Ensuite, il y aura dans l’ordre : une sélection roses, fleurs à l’unité et fête des mères de quatre produits chacune. Chaque *card* produit contiendra des informations essentielles : l’image du produit, son nom avec la mention « fleur française », son prix unitaire et un bouton « ajouter au panier ». La *card* sera cliquable et le lien dirigera vers la page du produit, qui le présentera avec les mêmes informations.

Ensuite on aura plusieurs pages qui n’en sont qu’une en réalité : l’index des articles. Il s’agit des pages « toutes nos fleurs », « roses », « fleurs à l’unité », ainsi que chaque page évènement (liens du pied-de-page). Elles afficheront toutes sous l’en-tête un bandeau proposant deux critères de filtres : couleurs et catégories. Les catégories seront les catégories décidées par les clients. A l’heure actuelle les catégories sont celles des évènements, mais aussi « fleurs à l’unité », « tulipes » ou « deuil ».

La page suivante, « qui sommes-nous », présentera d’abord, en deux paragraphes agrémentés deux photos, les propriétaires du magasin et la genèse de leur projet. Dessous, l’utilisateur aura accès aux coordonnées et aux horaires du magasin physique ainsi qu’à une carte GoogleMaps de son emplacement.

La page « mentions légales » sera une page tout à fait classique de mentions légales organisées en paragraphes. Elle contiendra une ancre pour le paragraphe « livraison », ancre qui sera référencée dans le pied-de-page.

## 3.4 Accès au compte et au panier

Les icônes « compte » et « panier » seront une seule et même page regroupant les informations de l’utilisateur, le contenu de son panier avec la possibilité de commander si son adresse est livrable, ou non le cas échéant, et de ses commandes passées et à venir. On choisit d’informer l’utilisateur qu’il est connecté ou qu’il y a des articles dans son panier avec des informations visuelles accolées aux icônes de la barre de navigation, dans l’en-tête :



On voit que l’utilisateur est connecté à son compte

et qu’il a deux articles dans son panier

Si l’utilisateur n’est pas connecté, il verra tout de même le nombre d’articles de son panier, seulement en voulant s’y rendre, il devra d’abord s’identifier ou créer un compte. Pour ces pages de connexion, il s’agira de formulaires sobres sur fond blanc.

## 3.5 Version mobile

On veillera à faire en sorte que le site est fonctionnel et esthétique sur mobile. L’aspect du site en général, décrit précédemment ne changera pas. On ajustera en particulier les nombres de cards affichées sur une ligne et on mettra un menu burger à la place d’une barre de navigation, pour faciliter l’utilisation sur petit écran.

# Concevoir la base de données

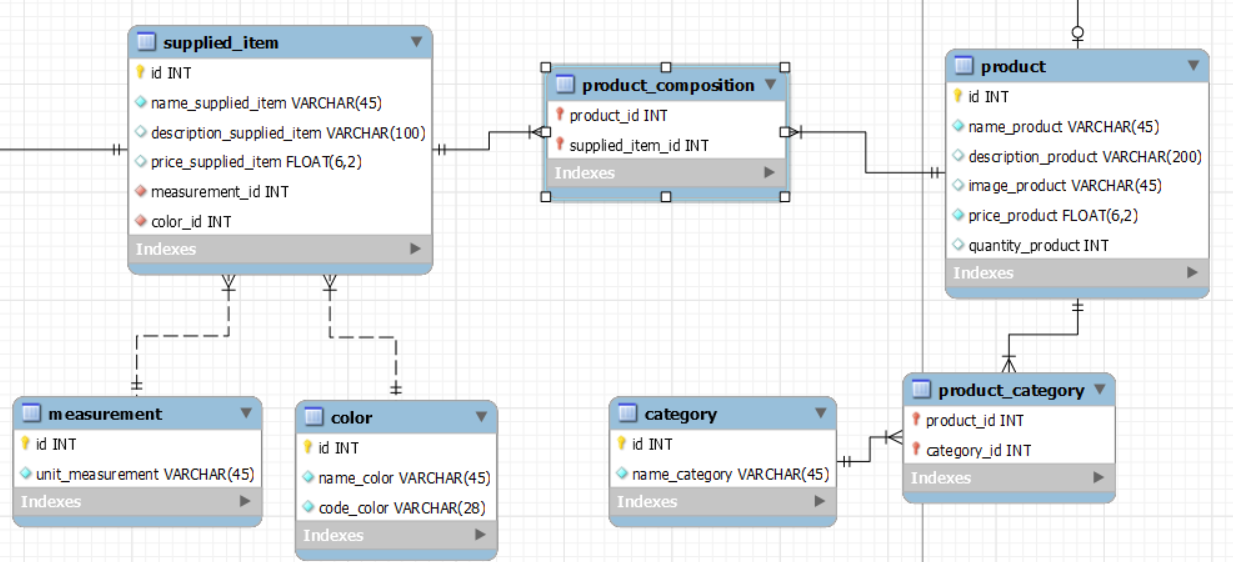
## 4.1 Les entités et les relations

Pour rendre possible la vente en ligne des articles vendus par Guillaume et Sabine Chollet, il a fallu en amont concevoir la base de données. J’ai utilisé l’outil de gestion et d’administration de bases de données Workbench pour créer un modèle logique de données (MLD). En pensant au projet, je me suis dit que les 3 entités les plus essentielles du projet étaient les clients, les produits et les fournisseurs avec les produits qu’ils livrent.

* On inclut dans les entités *customer, product, supplier* et *supplier\_item* les informations qu’on attendrait normalement dans de telles entités.

Il a été décidé qu’il serait utile d’avoir une entité spécifique pour les produits des fournisseurs.

En effet, je suis partie du principe que les fleuristes reçoivent des fleurs en gros pour ensuite soit les vendre à l’unité, soit faire des créations originales. Avec cela en tête j’ai créé une relation many-to-many et obtenu une table de jointure que j’ai appelée *product\_composition*. Elle contient les clés étrangèresdes produits réceptionnés et ceux vendus. Cela peut faire penser à une ligne de commande car pour chaque produit vendu, on retrouve un tuple par produit fournisseur qui le compose dans la table de jointure. Grâce à cela, on pourra obtenir par la suite les couleurs qui composent un produit vendu.



Extrait du MLD. Les relations qu’on ne voit pas dans la capture sont les lignes de commandes avec les fournisseurs (à gauche), et les lignes de commande des clients (en haut à droite). Voir le MLD entier en annexe.

On remarque également des entités telles que *color*, qui intègre *supplied\_item* car un produit fourni ne peut avoir qu’une couleur (sa couleur dominante s’il en a plusieurs), et une couleur peut concerner plusieurs produits fournis.

Autre exemple, l’entité *category* reliée à *product* via une table de jointure car un produit peut avoir plusieurs catégories et une catégorie peut concerner plusieurs produits.

## 4.2 Typer les données

Pour chaque attribut, on choisit un type de données adéquat. Pour les prix un nombre décimal (*float*) avec deux chiffres après la virgule et quatre avant maximum, pour les clés primaires des entiers auto-incrémentés (*int*), pour les données textuelles des chaînes de caractères (*varchar*).

On fait également attention à optimiser la longueur maximale de chaque donnée : l’attribut *code\_color* de l’entité des couleurs sera forcément beaucoup plus court que la description d’un produit (ici 28 contre 200).

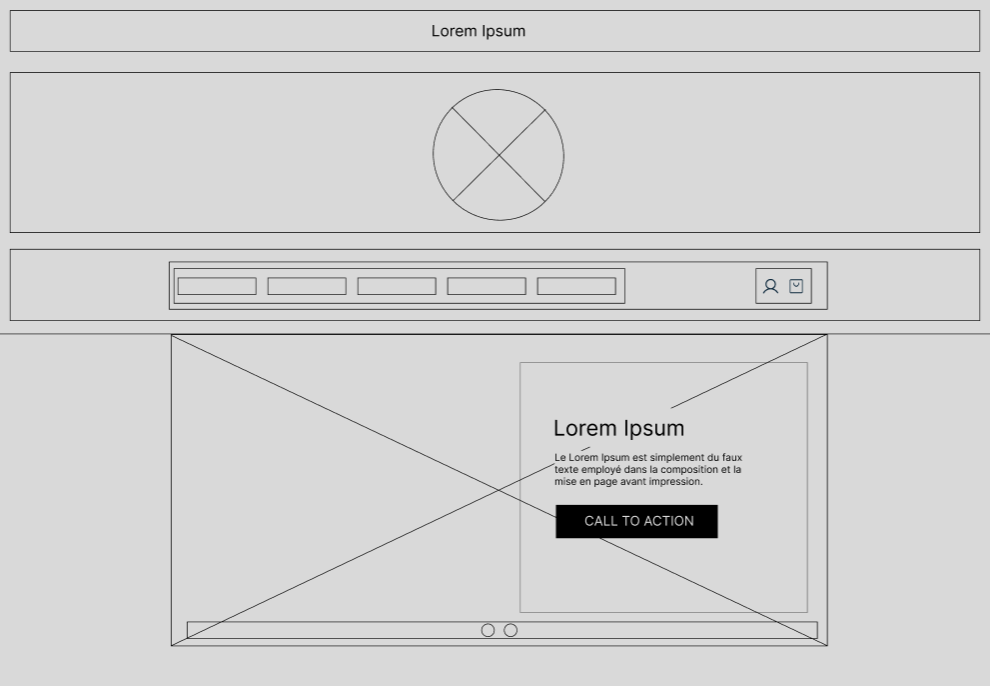
Dernier type un peu particulier, le booléen (*tinyint*), qui concrètement sera un 1 ou un zéro. On le retrouve dans l’entité *city* pour l’attribut *deliverable* et il indique si la ville est dans le périmètre livrable par les fleuristes ou pas.

En termes d’options, on a les clés primaires qui sont auto-incrémentées et certains attributs qui sont nullables ou non-nullables suivant leur caractère essentiel dans l’entité où ils se situent.

# Développer la partie frontend de la boutique en ligne

## 5.1 Structuration et mise en page d’une page

Pour montrer comment j’ai construit l’application en HTML, CSS et JavaScript, le plus parlant est de montrer la maquette fil de fer qui représente bien la structure HTML et les choix de mise en page. Pour cela, prenons l’en-tête et le carrousel de la page d’accueil :



Chaque bloc et élément est un jeu de balises HTML. On essaye au maximum d’utiliser des balises HTML spécifiques : header pour l’en-tête, nav pour la barre de navigation, ul et li pour une liste et ses items. On peut citer également section, footer, main, container, img, a, p et les h1, h2, etc. pour les titres. Tout cela aide au référencement de la page d’une part, mais également à expliciter la fonction de chaque élément, à les hiérarchiser, et rend plus efficace le développement.

Également les inputs avec leur label attitré, contenus dans une balise form.

Ce qui est notable aussi c’est les largeurs maximums de la barre de navigation et du carrousel. En effet, il est beaucoup plus confortable pour les yeux d’avoir une zone de lecture limitée en largeur. C’est pourquoi j’ai utilisé les propriétés et valeurs width: 80% et max-width: 1024px. Ainsi, la largeur du bloc sera toujours de 80% de son parent (ici le body), sauf si elle atteint 1024 pixels, auquel cas elle ne dépassera pas cette valeur. J’ai appliqué ces règles sur tout le site.

La propriété flex :

On remarque un grand bloc pour l’en-tête. Il est lui-même composé de trois blocs : le bandeau tout en haut, le grand container avec le logo au milieu et la barre de navigation en bas. Pour chaque élément centré (logo ou texte du bandeau) on a utilisé conjointement les propriétés display: flex, align-items: center (pour centrer verticalement), puis la propriété justify-content avec center en valeur s’il n’y a qu’un élément à centrer horizontalement (le logo par exemple), space-between s’ils sont plusieurs et qu’on veut qu’ils soient de bord à bord (la barre de navigation avec les liens/accès au compte, la liste de liens de la barre de navigation).

On retrouve cette méthode sur toute l’application, en particulier pour les blocs de *cards*.



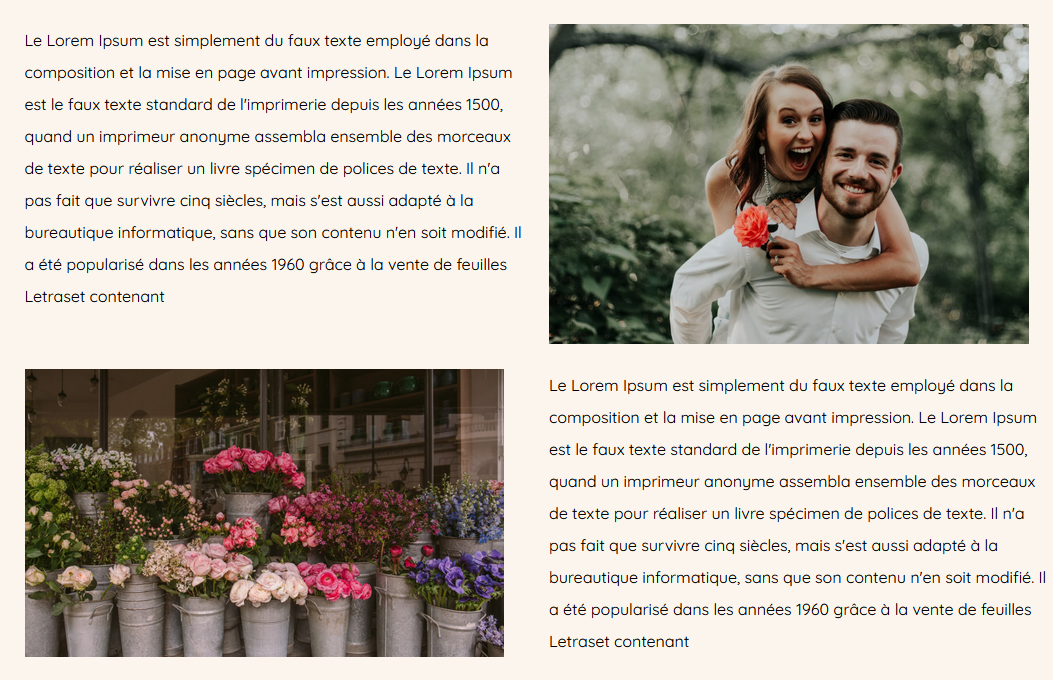
Les cards des évènements sont équidistantes et vont de bord à bord

La propriété grid :

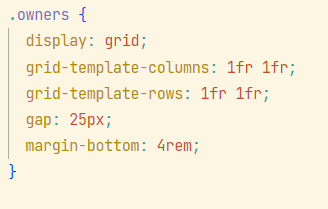
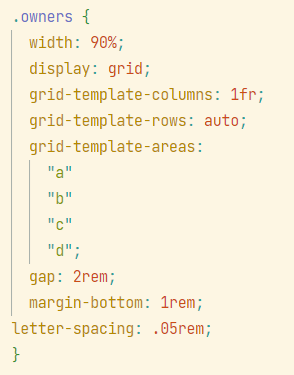
Une autre valeur de la propriété display est grid. Comme son nom l’indique, elle nous permet de traiter notre bloc comme une grille. On lui applique les caractéristiques de notre choix : nombre de colonnes et de ligne, ou zones avec des noms spécifiques.

Pour les *cards* ci-dessus par exemple, la grille a les dimensions 1 x 1 (une fraction par une fraction, appelée fr), et on y place une photo ainsi qu’un texte. Cela nous permet de créer une superposition très simplement. C’est aussi très pratique car on peut appliquer un traitement à l’image (flou, opacité, effet au survol) sans impacter le texte.

On a aussi utilisé grid pour réaliser la page « à propos », mais en lui ajoutant cette fois la propriété grid-template-areas. Cette propriété permet de définir spécifiquement le nom et l’emplacement des zones dans nos blocs. Ci-dessous, voici la première section de la page. Pour la version ordinateur, on a juste eu à définir les propriétés grid-template-columns et grid-template-rows à ‘1fr 1fr’. Changer seulement grid-template-columns à 1fr nous ferait obtenir les deux photos à la suite car c’est leur ordre dans l’HTML. On a donc fait appel à grid-template-areas pour définir les zones et pouvoir les placer dans un ordre différent de la version ordinateur (j’ai choisi de les nommer avec des lettres alphabétiques).

Extrait de la page « à propos », version ordinateur et version mobile.

Le CSS du bloc, version ordinateur et version mobile

On attribut une zone à un élément avec la propriété grid-area ().

## 5.2 Contenu dynamique

L’application contient des contenus dynamiques. On peut mentionner le menu burger de la version mobile ou le carrousel de l’accueil. Nous allons voir regarder de plus près comment fonctionne le jeu de loterie voulu par nos clients à l’occasion de la fête des mères.

Il s’agit d’un jeu type machine à sous, où au clic sur un bouton, trois emplacements nous montrent au hasard une image chacun. Si les trois images sont identiques, on a gagné.

Au niveau du HTML, nous avons trois  qui contiennent chacune une .

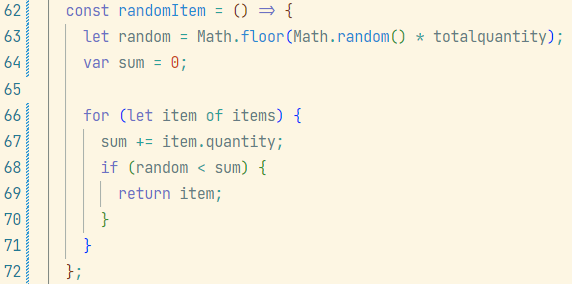
Explications de l’évènement déclenché au clic :

Premièrement, le bouton pour jouer est retiré et remplacé par un lien ‘retour’.  
Ensuite on instancie un tableau items contenant les images, quantités et id de chaque lot.

On récupère la somme des quantités avec la méthode array.educe() :



Grâce à cette variable, on va pouvoir générer un lot (item) au hasard, d’abord en générant un chiffre au hasard inférieur à totalquantity, puis on va récupérer un item au hasard avec une boucle for… of.



La variable random génère un chiffre au hasard inférieur au total de la quantité disponible de lots

Dans la boucle for… of (ligne 66), on va comparer une somme qui commence à zéro et est incrémentée de la quantité de l’item concerné à l’itération i. Si random lui est inférieur, on continue à boucler. Ainsi, plus random est un chiffre élevé, plus on a de chance de tomber sur un lot rare, puisque items commence par les lots qui comptent le plus de quantité.

Dans la fonction suivante (checkWin()), on copie le tableau des trois div.symbol (méthode array.from()) qu’on transforme (array.map()) en un tableau des émojis qu’elles contiennent grâce à textContent. On saura donc de cette façon le résultat obtenu au tirage. Avec une condition, on vérifie l’égalité entre les trois div et on affiche un message gagnant le cas échéant via un switch… case.

Toujours dans cette fonction, on colore le résultat s’il est gagnant en lui ajoutant une classe CSS, on personnalise le lien de retour avec l’id du lot gagné.

Dans le else, on détermine ce qu’il se passe si le tirage au sort est un échec.

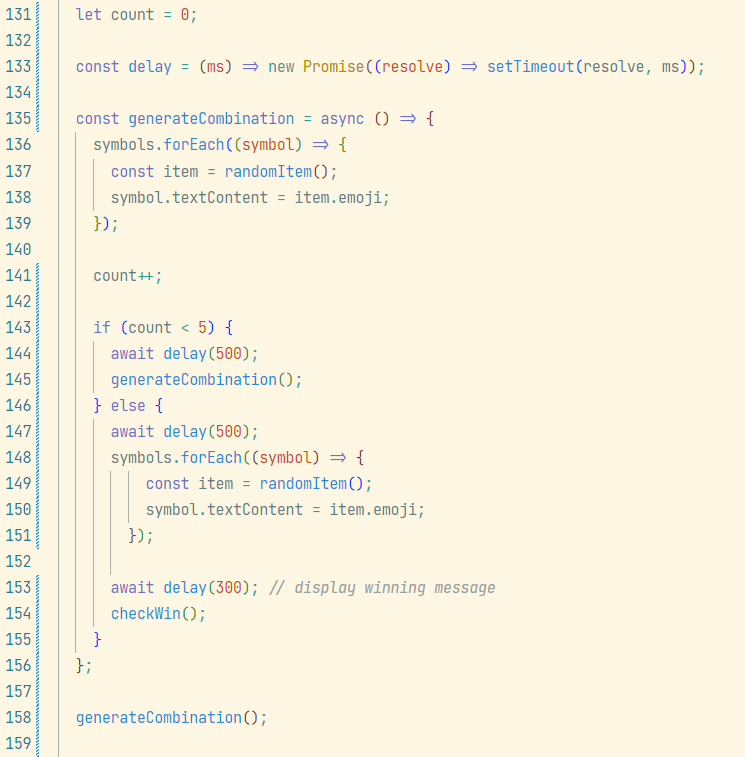


Ci-après, les dernières fonctions qui sont déclarées dans l’évènement :

D’abord une fonction asynchrone pour créer un intervalle de temps. Son paramètre est le temps d’attente que l’on veut attendre. Pour cela, on crée une promesse avec une fonction de rappel en argument et la méthode setTimeOut() avec un chiffre en argument, qui correspond au temps d’attente qu’on veut établir.

Ensuite la fonction generateCombination(). C’est elle qui rassemble l’exécution de toutes les fonctions déclarées jusque-là. Elle génère un émoji aléatoire (randomItem()) dans chaque div.symbol quatre fois avec delay() avant de s’arrêter sur un résultat définitif et d’appeler la fonction checkWin().

Enfin, on appelle la fonction generateCombination() pour l’exécuter.



Les déclarations de delay() et generateCombination() puis l’appel de generateCombination()

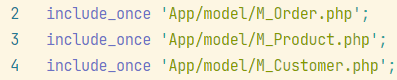
# Développer la partie backend de l’application

## L’architecture MVC

Pour développer l’architecture du site, le modèle MVC semblait le plus approprié. L’architecture Modèle-Vue-Contrôleur propose de séparer les requêtes à la base de données (la partie modèle) et l’affichage de l’application (la partie vue) via un contrôleur qui agît comme une sorte de bras articulé en prenant les résultats des requêtes pour les distribuer dans les vues.

La force de ce modèle est d’organiser et séparer les parties qui le composent. Cela apporte de la clarté et rend plus aisée la maintenance. Le modèle est indépendant des autres parties, et les autres parties peuvent dépendre d’un ou plusieurs modèles.

Par exemple, sur la boutique que nous développons, la vue compte nécessite d’afficher les informations de l’utilisateur connecté, de ses commandes passées et du contenu de son panier. En conséquence, dans le contrôleur du compte, on dépendra des modèles client (informations du client), commandes (historique des commandes) et article (informations des produits dans le panier, contenus dans la variable superglobale $\_SESSION) :

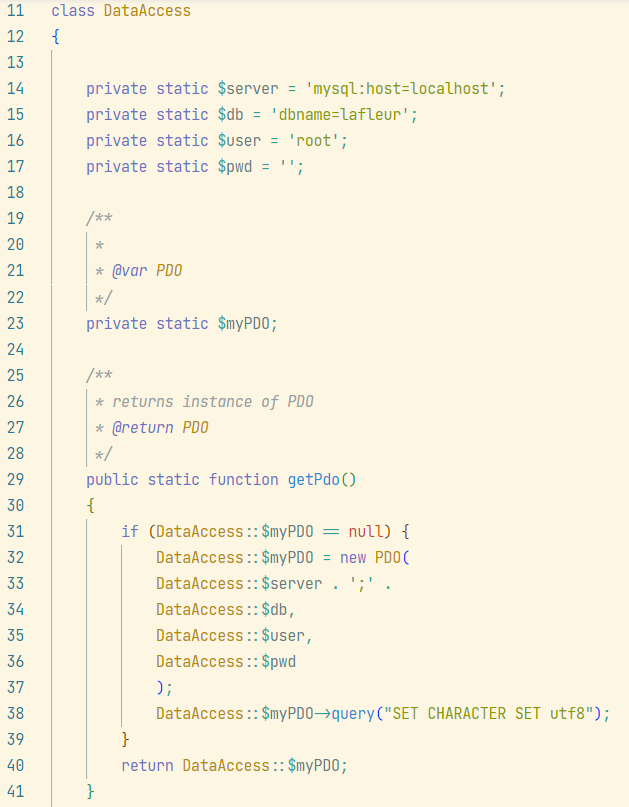


Le contrôleur inclut les modèles commande, article et client

## La classe PDO

Avant d’exécuter toute fonction de requête à la base de données, on doit s’y connecter à l’aide de la classe PDO (PHP Data Object). PDO « représente une connexion entre PHP et un serveur de base de données » (documentation officielle de PHP).

On configure les paramètres d’accès à la base de données (adresse du serveur, nom de la base de données, nom de l’utilisateur et mot de passe) pour pouvoir créer la fonction d’instanciation de PDO :



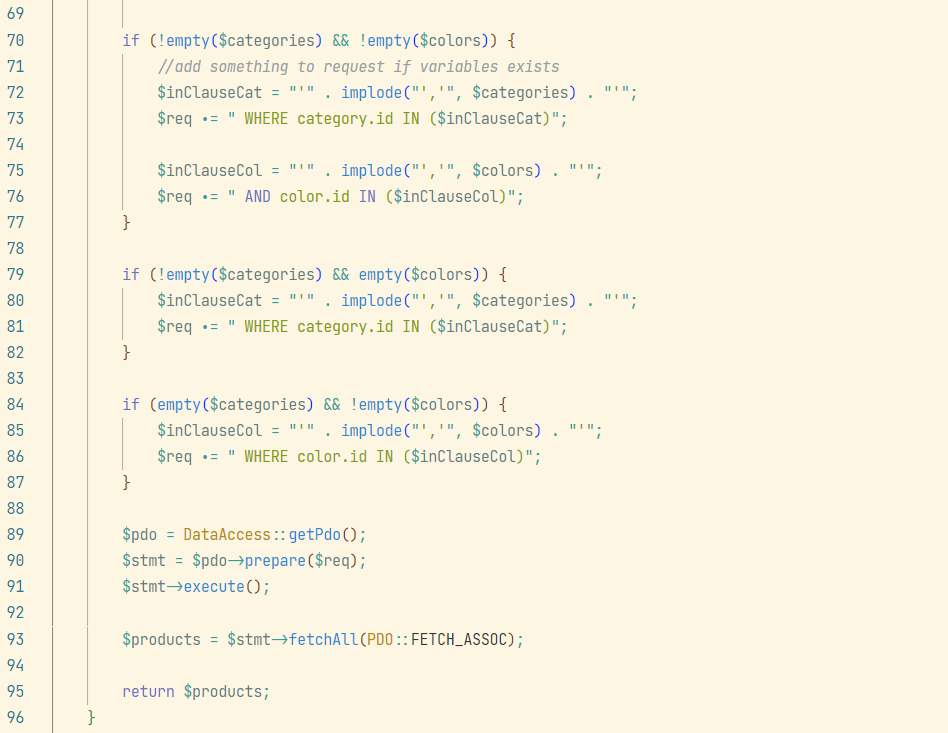
La fonction d’accès à la base de données de Lafleur

## Scénario

Imaginons un scénario où un utilisateur serait arrivé sur le site de Lafleur pour commander un bouquet de roses. Il habite dans une ville livrable par le couple de fleuristes. Utilisateur clique sur le lien vers toutes les fleurs. Au niveau du MVC les choses se passent comme ceci :

* Dans le modèle qu’on appelle M\_Product, il y une fonction indexSelected() qui récupère tous les articles :





On remarque des conditions qui peuvent ajouter des options de tri par couleurs et/ou catégories

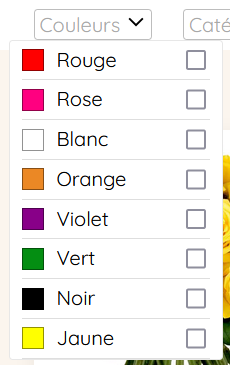
* Dans le contrôleur c\_consultation, en récupérant les informations contenues dans l’URL on sait que l’utilisateur est sur la page qui affiche tous les articles, on fait donc appel à la fonction que l’on vient de voir :



On remarque qu’on inclut le modèle préalablement (ligne 3). Lignes 7 et 8, on commence d’abord par les fonctions findColors() et findCategories() pour pouvoir afficher les formulaires de sélection des couleurs et catégories.

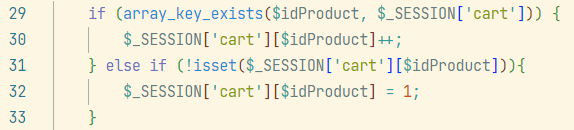
On voit aussi qu’on récupère les options de couleurs et/ou de catégories si l’utilisateur en a sélectionnées ; car valider ces options entraîne une redirection sur la même page.

* Enfin dans la vue v\_all on traite les tableaux de données transmis par le contrôleur.

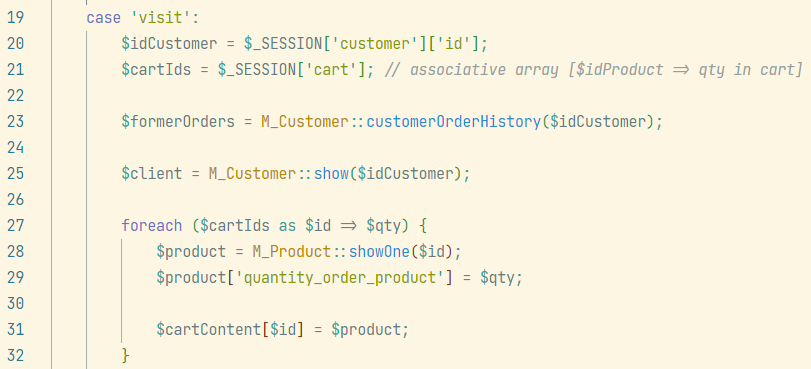
L’affichage du formulaire de couleurs et le rendu sur l’interface

Les produits récupérés par le contrôleur seront exploités de la même façon dans la vue, avec une fonction PHP foreach. Ils auront un bouton d’ajout au panier dont l’URL contiendra une information complémentaire : l’id du produit afin de pouvoir l’ajouter à $\_SESSION[‘cart’] (notre panier) :

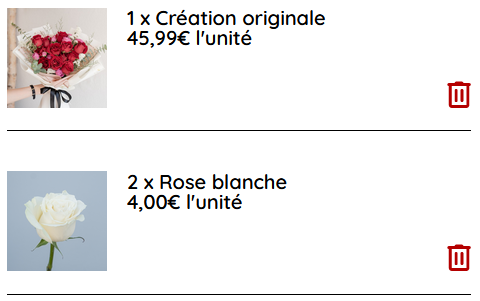


L’extrait de la fonction qui ajoute un produit au panier : elle ajoute le produit au tableau associatif ou incrémente sa quantité, suivant s’il était déjà présent ou non. Cette fonction figure dans un fichier dans un dossier à part car elle ne fait pas appel à la base de données.

L’utilisateur voit donc l’index des articles de la boutique, il choisit d’ajouter un article dans son panier, dont le détail est rendu dans la vue du compte. Comme on n’a aucune information sur les produits du panier, on a une requête qui obtient les obtient :

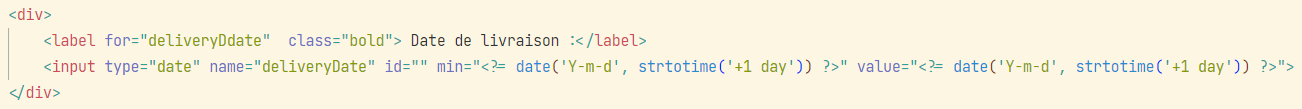


Pour chaque produit du panier, on récupère l’id auquel on applique la méthode showOne() de M\_Product qui nous retourne un tableau associatif des informations auxquelles on ajoute la quantité présente dans le panier. On ajoute tous les $product au tableau de produits $cartContent.



Le tableau associatif $cartContent nous permet d’obtenir ce rendu.

Pour résumer les étapes suivantes, l’utilisateur a la possibilité de choisir la date de livraison qui lui convient grâce à un input de type date avec comme attributs une date minimum (min) et une valeur par défaut (value) égales au lendemain de la date du jour puisque c’est le délai minimum possible :



On utilise PHP pour paramétrer ces attributs.

En cliquant sur « Passer commande », on crée une nouvelle commande dont on récupère l’id (méthode lastInsertId()) puis on ajoute chaque ligne de commande (id de l’article et quantité commandée) dans la table correspondante. Enfin, on veille à décrémenter la quantité en stock de chaque article commandé. Cet aspect stock nous permet de passer à la mission suivante, la gestion de stock.

# Gestion de stock

## Installer Symfony

Présentation de Symfony : framework architecture MVC

Les entités, controllers, repo, installer bootstrap, choisir un design fonctionnel et pratique, afficher telles infos sur l’accueil qu’on appelle tableau de bord

## La sécurité

Créer user et security, token csrf

# Blog

## Créer un blog WordPress

Qu’est-ce que wordpress, avantages, …

Ce que doit faire le blog en détails, installation, explication de la publi d’un article, les plugins qu’on a installés.

Proposer des templates wordpress aux clients, naviguer dans les options proposées par le template

## Coder un plugin

Pour le référencement, pour mettre en valeur des mots

# Présentation du jeu d’essai de la fonctionnalité la + représentative (données en entrée, données attendues et données obtenues)

# Veille sur les vulnérabilités de sécurité

* Regex et fonctions de vérifications
* Csrf token
* Requetes préparées
* Rôles dans symfony
* Privilèges dans phpmyadmin
* La vulnérabilité des sessions php
* Mots de passe hashés



Extrait de la documentation Symfony

# Description de la situation ayant nécessité une recherche

Comme vu précédemment, j’ai décidé d’utiliser Symfony pour développer la plateforme de gestion de stock de mes clients. La raison étant que j’ai eu l’occasion de travailler sur ce framework PHP lors de mon stage. J’avais acquis suffisamment de familiarité avec les contrôleurs, les entités, le moteur de templates Twig, etc.

Seulement je n’avais pas eu l’occasion d’installer un projet Symfony dès sa création. Je me suis donc lancée en allant directement sur la documentation officielle, en anglais.

# Resource trouvée et sa traduction (750 signes environ)

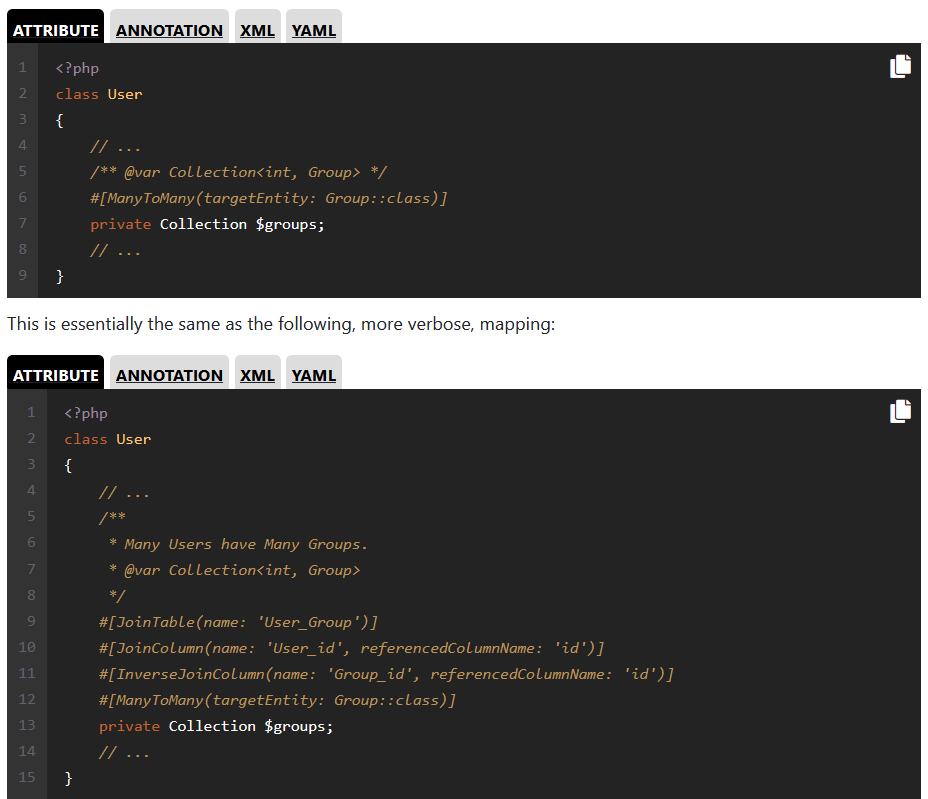
<https://symfony.com/doc/current/setup.html>

5. Réalisations du candidat comportant les extraits de code les plus significatifs et en les argumentant, **y compris pour la sécurité et le web mobile**

Article en anglais : <https://www.doctrine-project.org/projects/doctrine-orm/en/current/reference/association-mapping.html#many-to-many-bidirectional>

Utilisé pour : créer une propriété dans une entité qui revient à l’inverse d’une foreign key, et avec pour particularité d’être une manytomany donc une table qui n’a pas de colonne id. Pour éviter de rendre Doctrine confus, on fait une relation classique manytoone ou l’inverse, mais…

Lignes 10 et 11, on voit que que le ‘name’ de la colonne est « Group\_id » alors que c’est une colonne qui n’existe pas… ce groupe id va faire office de colonne id classique habituel d’une table qui n’est pas une table manytomany.



# Glossaire

Design Pattern

CSS

Framework

HTML

MLD

MVC

PHP

SGBDR

SQL