



KAWA

Desired Flavor



Sommaire

Introduction au projet	3
Résumé du projet	3
Cahier des charges	4
Définition du produit minimum viable	10
Amorçage du projet	11
Création d'une identité graphique	12
Synthétisation du parcours utilisateurs	14
Gestionnaire de Tâche et logiciel Tiers	15
Conception du projet	17
Réalisation du Zoning	17
Réalisation du Wireframe	18
Réalisation du prototype	20
Réalisation du Model Conceptuel de Donnée	21
Réalisation du Model Logique de Donnée	23
Réalisation de notre base de données sur notre SGBD	24
Mise en place de l'outil de versionning	25
Préparation du développement	28
Définition de la stack technique	28
Architecture du projet	30
Le Router	35
Classe & héritage	36
Développement Back-End	38
Composant métier	38
Prévoyance des faille de sécurités	41
Gestion des requête	43
Développement Front-End	44
Gestion des composant	44
SCSS & 7.1	45



Kawa est une application web de e-commerce spécialisé dans la vente de café haut de gamme. Cette application à pour but de mettre en avant des produits à la hauteur des exigences du consommateur et favorisant des méthodes de productions respectueuse de l'environnement et des producteurs locaux. A travers quelques clics, l'utilisateur pourra rechercher un produit de qualité selon les critères qui lui convient. Le producteur, la provenance, les arômes ou la variété du café sont de nombreuses distinctions permettant un choix éclairé. Une description subtile des saveurs et de la méthode de production peut aussi permettre aux plus curieux de se laisser tenté par la nouveauté. Cette application a été pensée pour les amateurs tout comme pour les professionnels du domaine. Ils pourront choisir le café selon leurs conditionnements, en dosette, ceux qui se lèvent tôt le matin, moulu en sachet pour une utilisation quotidienne ou encore en grain pour les baristes.

Le projet de cette application a été réalisé lors de ma formation à l'école La Plateforme en partenariat avec deux autres étudiants, Thomas Doan, Boris Tikhomiroff. Cet exercice fut destiné à rassembler les connaissances acquises jusqu'à lors et à intégrer des notions d'organisation à travers un projet d'envergure plus conséquent. J'ai aujourd'hui décidé de sélectionner ce projet pour ma présentation au titre car, au-delà du plaisir que j'y ai mis, il aborde à travers son développement les compétences visées pour prétendre au titre professionnel développeur web. Ayant été réalisé au milieu de l'année, la reprise de ce projet et pour nous l'occasion d'implémenter de nouvelles connaissances et de refactoriser le code. Cette dernière étape s'effectue au moment même de la rédaction de ce mémoire. En occurrence il est fort probable que certaines captures d'écrans présentées par la suite ne reflètent plus d'actualité.

Cahier des charge

Au lancement du projet nous avons reçu un cahier des charge destiné à nous indiquer les fonctionnalité nécessaires à la validation de l'exercice ainsi que le temps imparti à la réalisation de ce dernier. Bien que le cahier des charges ne définissait pas précisément à quel domaine la boutique devait appartenir. Nous avions les informations nécessaire pour mener à bien ce projet.

Le cahier des charge liste donc les éléments suivant :



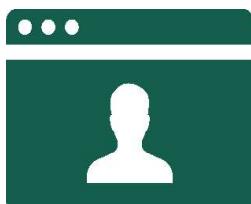
La page d'accueil doit être attractive



Mise en avant des produits phares / derniers produits mis en ligne



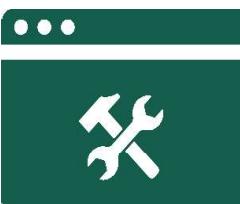
Chaque produit doit avoir une page complète générée dynamiquement



Création de comptes d'utilisateurs



Gestion du profil utilisateur
(informations, historique d'achat, consultation du panier...)



Gestion des produits à l'aide de back office pour les admins



Gestion des catégories et des sous catégories de produits



Barre de recherche de produits

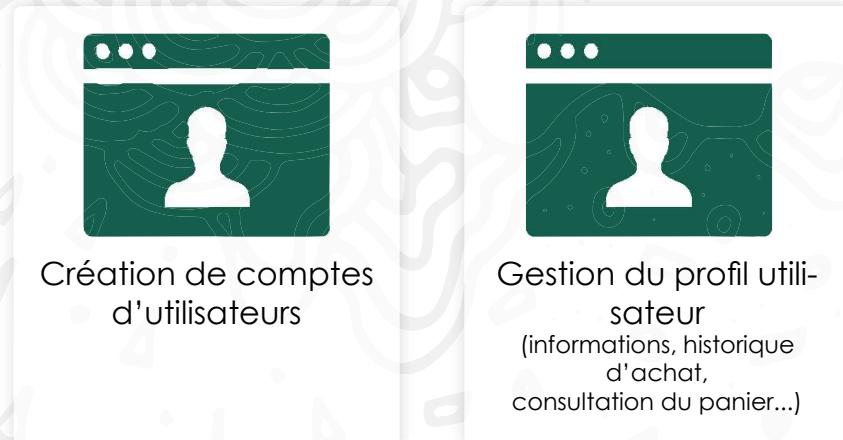


Validation du panier (simulation du paiement)

Que peut on déduire du cahier des charges ?

Nous pouvons constater que le cahier des charges fournis des directives limités. Cependant, si nous ajoutons ses informations aux bonnes pratiques liées à la conception d'un tel projet, nous pouvons dès lors et déjà commencer à synthétiser les composants nécessaires à l'élaboration d'une feuille de route permettant la réalisation du site e-commerce en question.

Concernant l'utilisateur



Dans un premier temps nous pouvons constater la présence d'un compte utilisateur, ainsi que la gestion de son profil. Cela nous permet de définir une première entité nécessaire à la navigation.

En outre la présence d'un compte utilisateurs dans un tel domaine, nous emmène à la conservation de données personnels encadré par les réglementations RGPD. De ce fait nous devons penser à l'accès et la sécurité des données utilisateur ainsi qu'au droit de rectification de ses données ou, dans le cas où l'utilisateur n'aurait plus accès à son compte, la possibilité que l'administrateur puisse lui retourner.

Nous avons là quelques informations supplémentaires qui se rajoute à notre cahier des charges. En soi, pour répondre à ces dernières remarques, le site doit avoir :

- **Un module d'inscription** permettant une navigation fluide et permettre un jeu de données adaptés aux fonctionnalités métiers de l'application
- **Un module de connexion** afin d'authentifier l'utilisateur en question.
- **Un espace Account ou profil** afin que l'utilisation puisse avoir accès à ces informations personnels et/ou liés à ces mêmes informations.
- **Account**, Un espace permettant à l'utilisateur de pouvoir modifier à tout moment
- **CGU**, Une page destinée à la rédaction des termes légal des conditions d'utilisation et de l'utilisation des données recueillies..

Concernant la navigation



Gestion des produits à l'aide de back office pour les admins



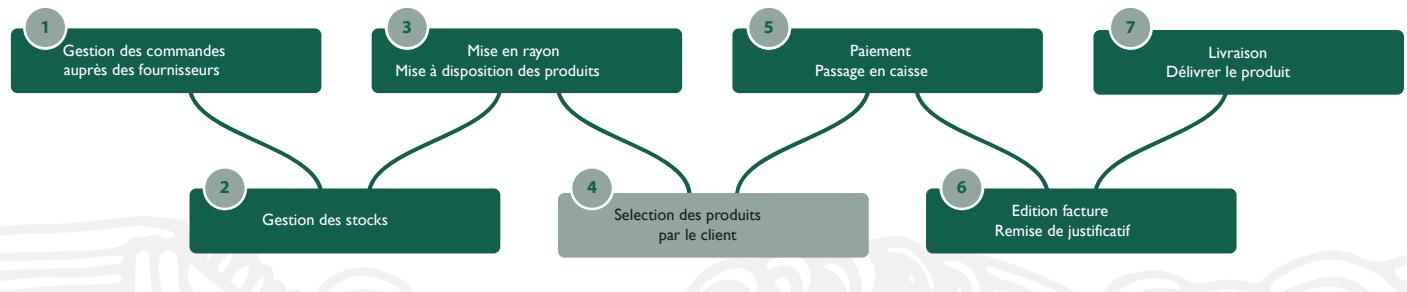
Gestion des catégories et des sous catégories de produits

Nous pouvons notamment constater que le projet comprendra l'élaboration d'un back office nécessaire à la Gestion des produits et Gestion des catégories et des sous catégories de produits part les administrateurs. Par conséquent nous pouvons définir des rôles qui seront intégrés à notre entité ci-dessus : exemple : rôle : admin ou Publics. C'est rôle pourrons définir les droits d'entrée à certaine routes, ou la possibilité d'utiliser certaines fonctionnalités. L'administrateur devra notamment avoir accès aux informations de l'utilisateur dans le cas où celui-ci le demanderait.

En vue de la croissance de l'entreprise nous définissons notamment les fonctionnalités propre au domaine d'activité en permettant à l'administrateur la possibilité d'automatisé la transmission de certaines informations. Partant ainsi du principe qu'au début du projet les rôles sont divisé en deux parties distinctes, le client rôle «publique ou utilisateur» et le commerçant «l'administrateur».

Pour définir les fonctionnalité propre accessible par l'administrateur nous avons synthétisé le domaine d'activité d'un commerçant schématisé sur le diagramme ci-dessous :

Synthèse de l'activité de commerce



■ activité lié au commerçant

■ activité lié au client

Nous pouvons considérer ainsi chaque activités liées au commerçant comme un fonctionnalité potentiels à implémenté. Cependant, il reste à déterminé les fonctionnalités réalisable selon le temps imparti.

A partir des remarques défini précédemment nous pouvons donc en déduire que notre projet devra comprendre :

- **Back Office**, un espace dédié à la navigation de l'administrateur
- **Un module création d'articles** afin de retranscrire les informations concernant les Produits commandé auprès des fournisseurs.
- **Un module de modifications d'article** afin de permettre à l'administrateur de modifier/corriger les informations ou de supprimé un article.
- **Un module de gestion des utilisateur**, afin de permettre à l'administrateur d'accéder facilement aux informations de l'utilisateur ainsi que de changer son rôle.
- **Un module de gestion de commande**, afin de permettre à l'administrateur de prendre connaissance des commandes passer ou de faire les comptes.
- **Un module de gestion des stock**, afin de permettre à l'administrateur de restocker rapidement le nombre d'unité disponible.
- **Un module de gestion de livraison**, afin d'informer l'utilisateur de l'envoie de son colis

Concernant la navigation



La page d'accueil doit être attractive



Barre de recherche de produits



Mise en avant des produits phares / derniers produits mis en ligne



Chaque produit doit avoir une page complète générée dynamiquement

La navigation devra comprendre des notion d'UX afin de permettre une recherche rapide et fluide des éléments par l'utilisateur. En se points certaine notion sont fondamental pour permettre une navigation agréable.

En vue des directive lister ci-dessus, nous pouvons envisagé que le site internet sera composé d'élément suivant :

- **Page d'accueil**, sera composé d'éléments visuels en cohérence avec le branding de la marque.
- **Une Barre de Recherche** disponible sur toute les pages afin de faciliter la navigation de l'utilisateur
- **Une page «Catalogue»** dédié à recenser les articles correspondant au résultat de recherche
- **Recherche avancé**, un système de recherche par catégorie et sous-catégorie accessible et lisible.
- **Une page produit**, permettant à l'utilisateur d'avoir aux informations sur le produits.
- **Un témoin de confiance**, un système de note ou de commentaires rassurant l'utilisateur sur ces choix .
- **Une navigation attractive & cohérente**, destiné à rassuré et accrocher les premier utilisateurs.

Concernant la navigation



Afin de faciliter la validation du tunnel de vente, il est important que l'utilisateur puisse consulté où il en est sûr ces achat à tout moment. Pour cela, il est primordiale que sont panier soit accessible en quelques clic. De la même manière, lors du paiement il faut limité au maximum les points de frictions qui seraient susceptible de découragé l'utilisateur à poursuivre.

Sur ce point nous devons prendre en compte les éléments suivant :

- **Un composant panier** accessible depuis toute les pages.
- **Une étape de validation d'achat** fluide, éclairée et rapide.

A la lecture du cahier des charge nous pouvons en déduire un ensemble d'éléments qui constitueront notre produit final. A partir de ces éléments nous allons pouvoir définir notre produit minimum viable qui sera notre objectif à tenir dans le temps imparti. Ce produit minimum viable permettra une utilisation compète du site tout en écartant les parties complexes qui nécessiterais plus de temps au développement.

Produit Minimum Viable

Le MVP(produit minimum viable) est fondamental dans la manière d'aborder la mise en production d'un projet. Elle permet de définir, en accord avec le client, les besoins liés au métier tout en écartant les fonctionnalités annexes. Le MVP répond seulement au besoins principales de l'application dans son ensemble et permet de définir précisément les tâches qui incombent au développeur.

En ce qui concerne cette application, nous pouvons dès et déjà définir les points sur lesquels nous nous étendrons pas :

La gestion des commandes auprès des fournisseurs :

En vue de la particularité liés aux produits et aux différents manières de se réapprovisionné nous avons décidé de ne pas prendre compte de ce point. Nous estimons le commerçant pourra toujours gérer ces commandes à travers les méthodes de communications habituelles;

L'automatisation de la livraison :

De la même manière, nous avons décidé de développer la partie livraison dans une second temps. Il reviendra donc au commerçant de choisir lui-même la manière de livrer les produits à ces clients à travers des partenaires tiers.

Cependant, nous avons choisi de mettre à disposition du commerçant la possibilité de valider une commande envoyée afin que le client soit informé. Ainsi, le client peut être assuré que l'envoi a bien été effectué et peut être attentif à la réception. Il est cependant probable que ce dernier soit automatisé dans une prochaine version de la boutique.

Système de paiement intégré:

Le cahier des charges stipule que le système de paiement peut être stimulé. Mais, il est considéré comme impératif à intégrer en vue d'un déploiement. Ce sera donc l'une des premières implémentations à faire après la réalisation du MVP.

- > **Un module d'inscription** permettant une navigation fluide et permettre un jeu de donnée adaptés fonctionnalités métiers de l'application
- > **Un module de connexion** afin d'authentifier l'utilisateur en question.
- > **Un espace Acount ou profil** afin que l'utilisation puisse avoir accès à ces informations personnels et/ou liés à ces même informations.
- > **Acount**, Un espace permettant à se dernier de pouvoir les modifier à tout moment
- > **CGU**, Une page destinée à la rédaction des termes légal des conditions d'utilisation et de l'utilisation des données récoltés..
- > **Back Office**, un espace dédié à la navigation de l'administrateur
- > **Un module création d'articles** afin de retranscrire les informations concernant les Produits commandé auprès des fournisseurs.
- > **Un module de modifications d'article** afin de permettre à l'administrateur de modifier/corriger les informations ou de supprimé un article.
- > **Un module de gestion des utilisateur**, afin de permettre à l'administrateur d'accéder facilement aux informations de l'utilisateur ainsi que de changer son rôle.
- > **Un module de gestion de commande**, afin de permettre à l'administrateur de prendre connaissance des commandes passer ou de faire les comptes.
- > **Un module de gestion des stock**, afin de permettre à l'administrateur de restocker rapidement le nombre d'unité disponible.
- > **Un module de gestion de livraison**, afin d'informer l'utilisateur de l'envoie de son colis
- > **Un composant panier** accessible depuis toute les pages.
- > **Une étape de validation d'achat** fluide, éclairée et rapide.

Création d'une identité graphique

L'identité graphique d'une entreprise ou d'un site web est un élément fondamental dans la cohérence et l'appui de la crédibilité auprès des utilisateurs. Dans le cadre de cette exercice aucun élément graphique nous avait été donné. Nous avons donc défini rapidement quelques éléments centraux nous permettant de définir ce qui nous servira de base pour permettre de créer une maquette.



Capture d'écran de l'étape de recherche sur les élément graphique

Pour l'identité Visuel nous voulions abordé un branding en relation avec les enjeux actuels. Nous voulions faire transparître un marque responsable et respectueux des produits. Notre design devait donc refleté quelques chose de naturel et organique en respect avec la nature. Mais avec des finition minimaliste et aéré de manirère à alléger la navigation.

Nous nous sommes naturellement tourné vers un vert forêt comme Primary Color, celui-ci accompagné d'un marron en accord avec la couleur des graines de café torréfié. Les images devaient être aussi soignées de manière à accentuer un sentiment d'apaisement auprès de l'utilisateur.

Le logo devait être lisible et reconnaissable et les polices d'écritures devaient témoigner de la rencontre entre simplicité, minimalité et naturel et authenticité.

Branding associé à l'image de la marque



Logo avec élément typographique

Couleur utilisée



Police de caractère

DM Serif Display Regular
Défini pour l'acroche

Century Gothic

Police simple et élégante
Adapté au Print comme à l'écran
elle permet un lecture fluide et aérée



utilisation en icon



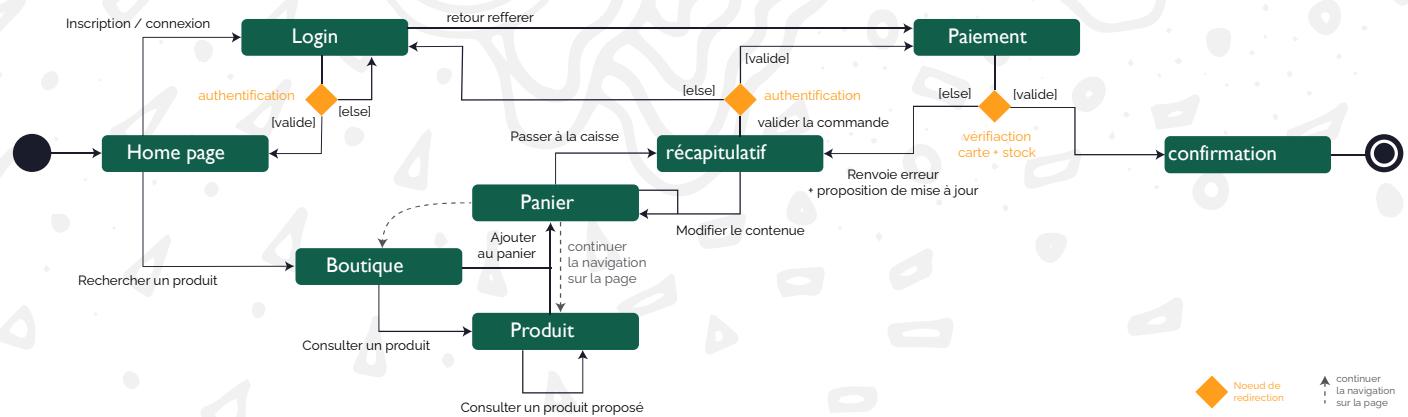
Document définissant l'identité graphique du site Kawa

Nous n'avons pas précisément défini de charte graphique, ni de design système. En vue du temps imparti il fallait se limiter à une documentation comprise par toute l'équipe sans perdre de temps sur des réglementations annexes qui se définiraient pas la suite. C'est différent choix, ce construirait au fur et à mesure de l'avancer du projet. Le Maquettage allait être un élément central pour l'harmonisation de cette identité.

Synthétisation du parcours utilisateurs

Une fois l'identité visuelle et le cahier des charge établi nous avons défini le parcour type de l'utilisateur. Cela nous permis d'évaluer les différent points de frictions potentiels ainsi que de déterminé facilement le tunnel de vente.

Diagramme de navigation - Tunel de vente



Nous pouvons constaté dans le diagramme ci-joint que nous avons pensé la navigation de manière à ce que l'utilisateur puisse librement bouclé entre le mise en panier d'un produit, la consultation de la page produit avec d'autre et la page «boutique» définit plus haut comme étant le catalogue, soit le résultat de recheche de l'utilisateur. Pour ce faire nous avons penserons a ce que les composant utilisé sur ces trois pages puisse être facile d'utilisations.

Nous pouvons notamment constater que nous essayons d'atténuer les point de frictions liée à l'inscription de l'utilisateur. Ainsi l'utilisateur n'a pas besoin d'eêtre inscrit pour naviguer sur la page? Il peut donc librement mettre des élément dans son panier. C'est seulement le moment où il souhaite passer à la caisse que nous le redirigeons vers l'innscription.

De la même manière, ce choix, nous indique que l'inscription devra ce limité au minimum des information nécessaire pour le log. Une fois inscrit l'utilisateur ce voit directement rediriger vers l'étape de paiement. Ainsi nous assurons des reduire le nombre d'abadon potentiel à ce moment là.

Gestionnaire de Tâche et logiciel Tiers

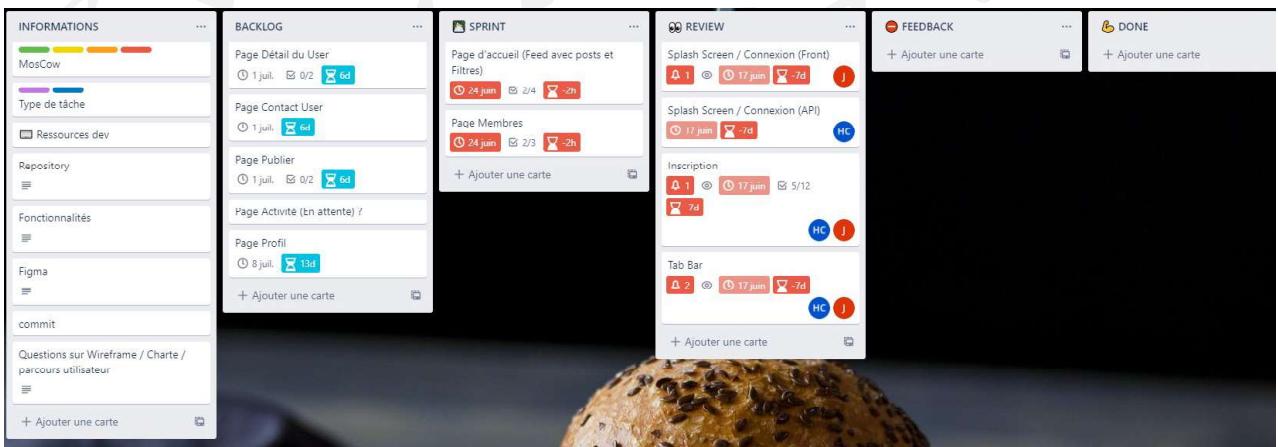
Dès l'élaboration de la conception nous avons décidé de travailler en méthode Agile. Afin de s'assurer une bonne cohésion d'équipe nous avons dû définir les différents canaux d'achèvement à notre disposition ainsi qu'un gestionnaire de tâche adapté à notre méthode.

Pour cela nous avons utilisé les logiciels suivants :



Trello

Très répandue dans le milieu du développement web, Trello est un bon moyen de s'organiser en équipe. Dès le début du projet nous nous sommes munis d'un tableau destiné à ce projet.



ScreenShot du Trello

Utilisation des listes

Nous avons segmenté les listes en 6 parties :

1. La première nous sert à récupérer les informations communes au projet. La légende des tickets comme l'importance de la tâche ou la nature de celle-ci. Les références utilisées, les liens vers le repository partagé, les liens vers la maquette, etc..
2. Le Backlog référence l'intégralité des tâches à faire.
3. Le sprint définit les tâches en cours et le temps imparti pour les terminer.
4. La liste Review et les tâches exécutées par le développeur qui doit être revue par l'ensemble des autres utilisateurs avant implémentation sur la branche commune.
5. Le Feedback est une liste destinée à récupérer des tâches en instance dans le cas où un développeur serait face à une problématique donnée, que cela vienne d'un bug à fixer ou le besoin d'être éclairé sur une partie de code développée par un confrère.
6. Done comprend les tickets finis et implémentés.

Utilisation des Tags

MosCow
Dans la liste [INFORMATIONS](#)

Étiquettes

Doit être fait **Devra être fait** **Pourra être fait**

Non fait pour le moment +

Nous avons utilisé certaines couleurs de tags pour définir l'état de nécessité des tiquets en quatre état :

1. Doit être fait
2. Devra être fait
3. Pourra être fait
4. Non fait pour le moment

Description

De cette manière nous savons quel tiquet est à privilégié. Au fur et à mesure des étapes nous déterminons les tâches à faire dans le backlog. Ainsi nous pouvons établir les tâches à l'avance lors d'un daily et définir ensemble une route à suivre.

Type de tâche

Dans la liste [INFORMATIONS](#)

Étiquettes

Conception **Code** +

D'autres couleurs ont été utilisées afin de définir des étapes clef comme indiqué présent avec ces deux tags de conception ou de développement.

Dans la capture d'écran suivante nous pouvons voir comment nous nous sommes organisés pour au sein d'un même ticket. Segmenter les tâches en différents points permet de s'y prendre pas à pas. Et permettre à chaque checkbox validé de commettre la branche sur laquelle ont travaillé.

FULL FRONT
Dans la liste [Terminé](#)

Membres

J +

Ajouter à la carte

Membres

Étiquettes

Checklist

Dates

Pièce jointe

Image de couverture

Champs personnels

Ajoutez des menus déroulants, des champs de texte, et plus encore à vos cartes

Description [Modifier](#)

Passage au Front - Organisation pour le premier Week-End d'Avril

Front Admin [Masquer les tâches cochées](#) [Supprimer](#)

100%

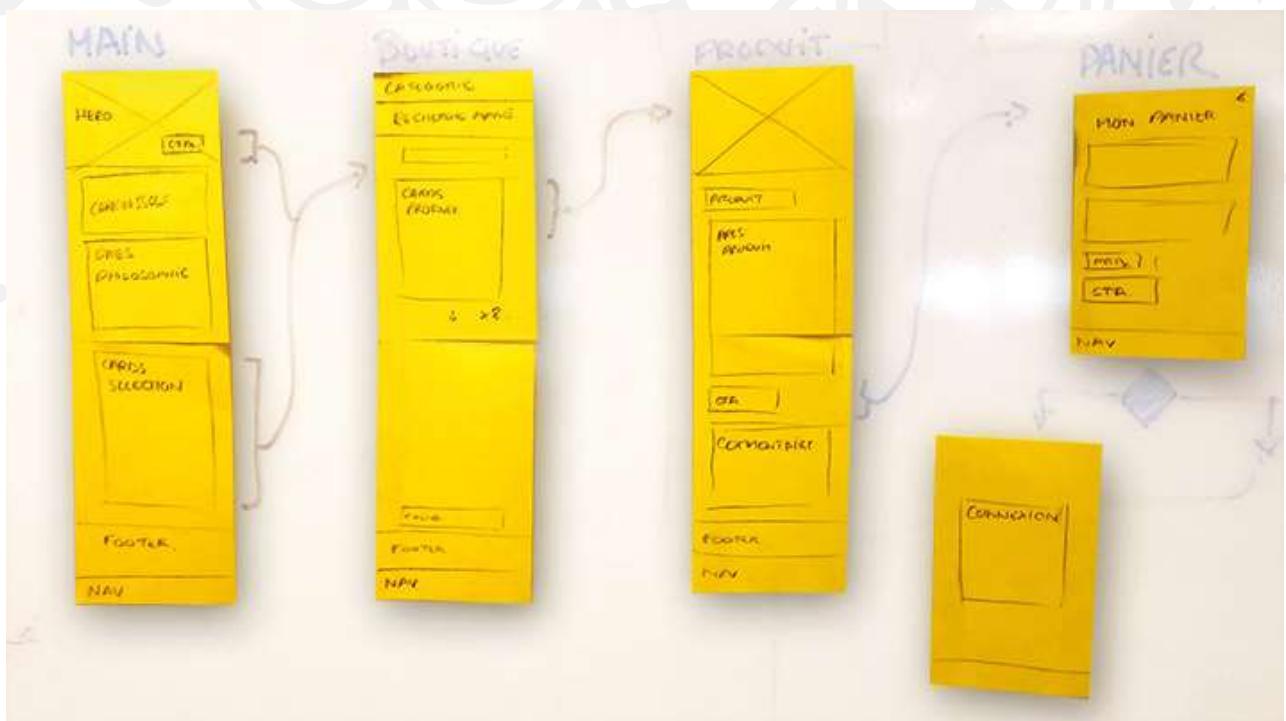
Intégrer la sidebar à jour avec tous les chemins de page

Front Admin - gestion des stocks

Ajouter une image sur l'index

Réalisation du Zoning

Nous passons à présent à l'étape de conception. En nous aidant du diagramme de navigation et du cahier des charge définit plus haut. Et en prenant compte des bonne pratiques. Nous pouvons commancé à construire des pages avec des larges zones dans représentant nos différent composant.



Le zoning est une étape décisive pour définir avec plus de compréhension le parcours utilisateur et ainsi estimer le nombre de pages nécessaires. Cette étape couplée au Modèle Conceptuel de Donnée nous permet de mieux synthétiser les attentes.

Pour ce faire nous avons choisi un approche simple mais efficace. A l'aide d'un tableau et de post-it, nous nous sommes posé à trois autour d'une table et nous avons chercher ensemble meilleur manière d'appréhender la manière d'agencer les différentes fonctionnalités tout en cherchant à définir l'enchainement des screen.



Nous avons notamment décidé de construire et développer l'application en mobile first afin d'optimiser la minimalité de notre application.

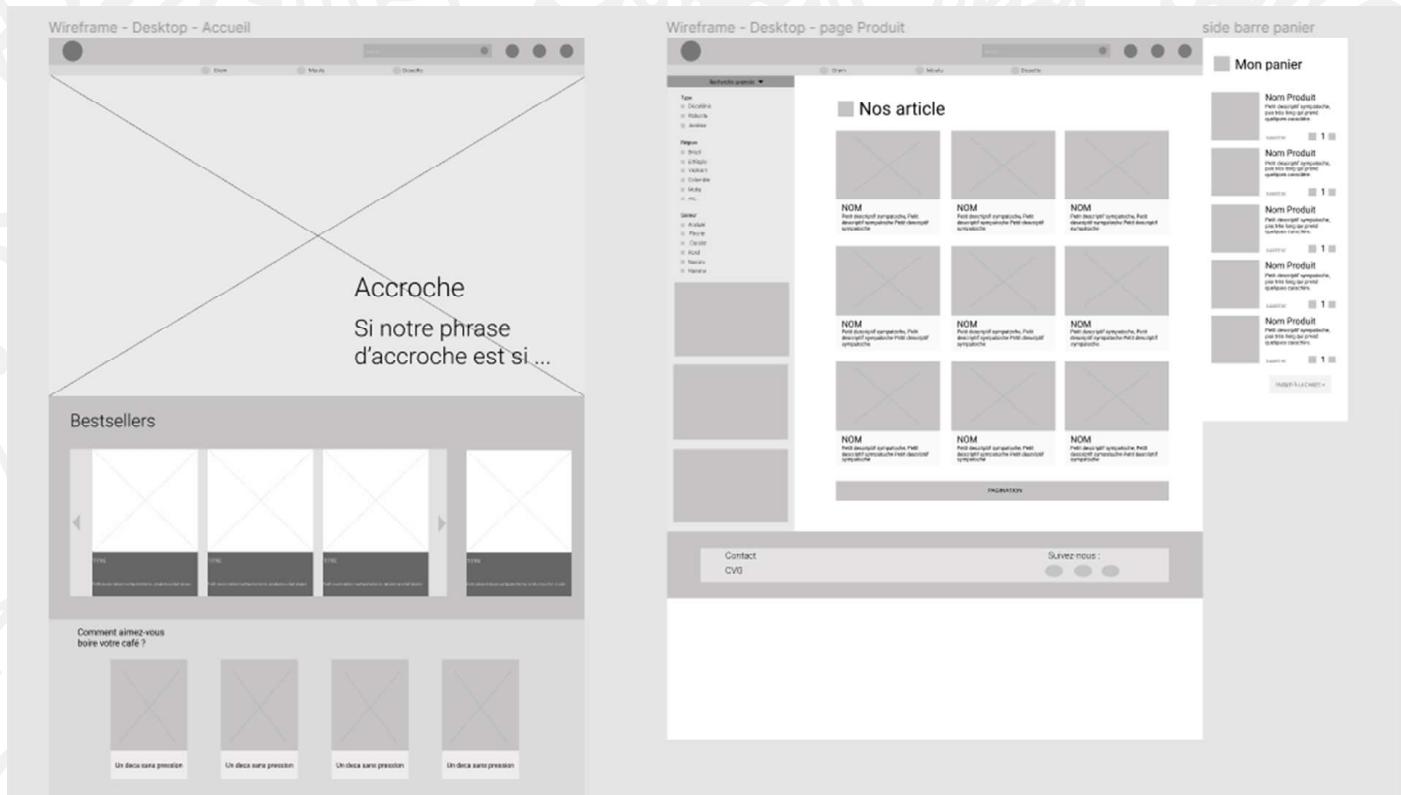
Réalisation du Wireframe

Bien que certain designer reste sur papier pour faire le wireframe, nous avons choisis d'utilisé figma pour réaliser cette étape. Cela nous a permis une meilleur prise en main du prototype par la suite et à la fois de mieux nous organiser sur la structure des composant et de la tailles des éléments.



Parties de réalisation du wireframe format mobile

Nous avons choisi de concevoir l'interface avec une approche mobile first selon le RWD(responsive web design). Afin d'accroître la fluidité de l'utilisateur j'ai choisi d'adapter le placement des éléments selon leurs utilisations. Ainsi les éléments les plus utilisés se trouvent dans la zone de confort en mobile et se replacent selon la mise en page communément admise lors de l'utilisation sur Desktop.



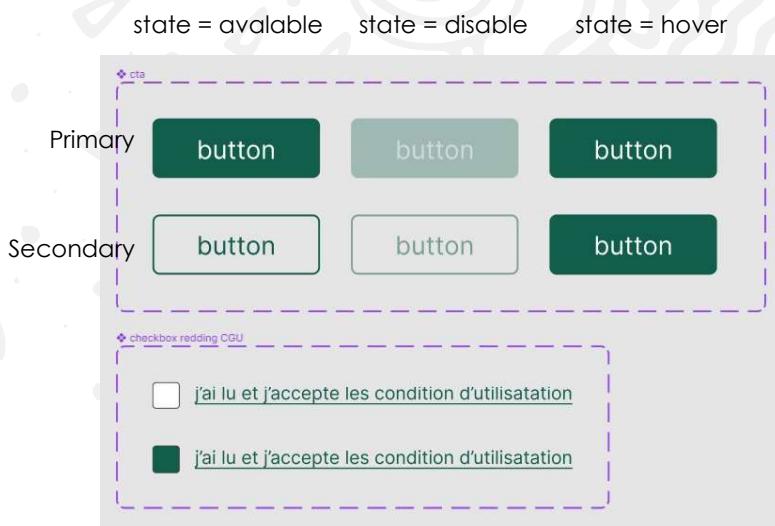
Parties de réalisation du wireframe format Desktop

Dès le Wireframe nous avons décidé de travailler avec des composants dans lesquels nous avons défini des variants ainsi que leurs différents états. Bien que cette observation puisse paraître anecdotique, elle reflète en son sens une notion centrale dans la pratique du développeur, à savoir élaborer et agir en pensant aux conséquences que cela aurait en aval du processus.

Ainsi élaborer des composants dès le wireframe, nous a permis de faciliter l'étape du prototypage.

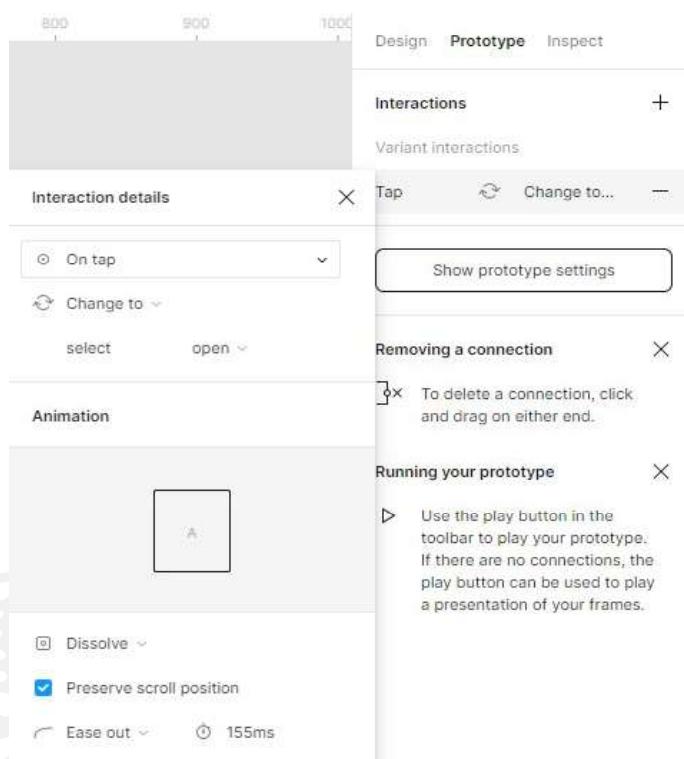
Réalisation du Prototype

Comme indiqué dans le chapitre précédent, l'utilisation des composants et de leurs variants nous a permise de mettre en forme rapidement le prototype à partir du wireframe. La réalisation du prototype consistait donc à designer et à programmer les différents évènements et la navigation selon les action données.



Cette capture d'écran montre l'utilisation des composant proposer par figma. Ces élément ont la particularité d'être lié à un élément parent où tout changement de visuel de son état va entraîne obligatoirement un changement chez les clones associés.

Nous pouvons notamment constater que ces composants se trouvent dans plusieurs versions. Figma appelle cela des opération boolléennes. Elle permet au designer de définir différents attributs qui fera varié le design selon leurs état.

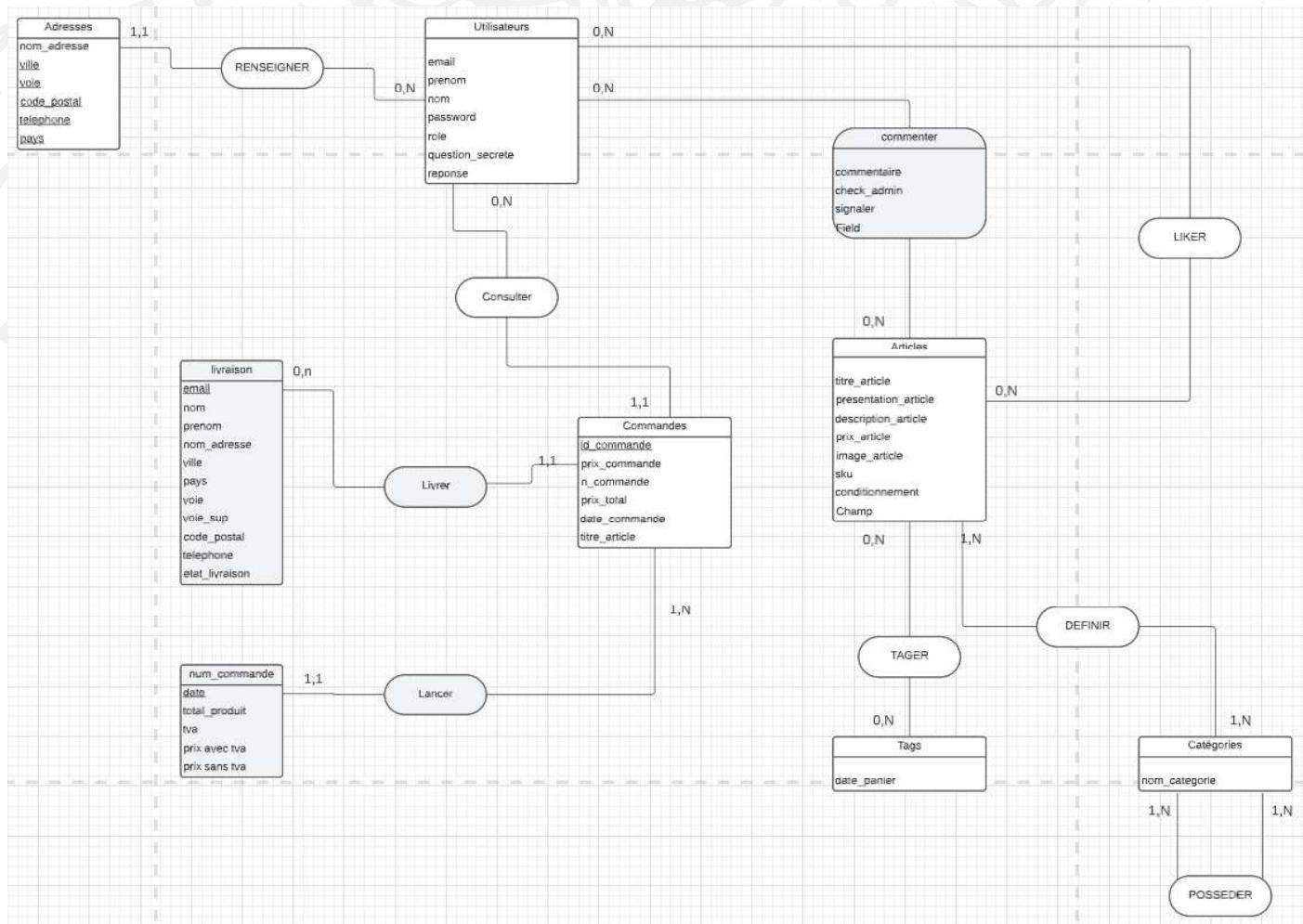


Une bonne partie du prototypage consisté à organiser les états du prototype à travers les événements. Elle permit ainsi à tous les développeurs de comprendre l'enchaînement lié au différents composants.

Model Conceptuel de Donnée

Réaliser le Model Conceptuel de donnée nous a permis d'identifier les principales entités à représenter, leurs relations et leurs attributs, et d'analyser la structure conceptuelle du système d'information.

Model Conceptuel de Donnée de Kawa



Les parties pris

Les Adresses :

Commençons par constater en haut à droite du diagramme que nous avons choisi de détacher l'adresse qui pourrait être conçue comme un attribut propre à l'utilisateur en une entité distincte. L'adresse doit être renseignée par un et un seul utilisateur mais un utilisateur peut ne pas renseigner d'adresse ou plusieurs adresses. Ce choix s'explique par le point de friction lors qui peut s'établir lors de l'inscription. Nous avons préféré retarder la saisie d'une telle information lors de l'acte d'achat. De cette manière l'utilisateur n'est pas contraint d'utiliser l'adresse renseignée et peut ce permettre de passer des commandes ponctuelles.

Nous pouvons notamment en déduire que l'utilisateur à l'option d'enregistrer plu-

sieurs adresses, ce choix se comprends dans le cadre où l'application est destiné à une clientèle professionnel qui peut se retrouver à alimenter plusieurs magasins.

Les catégories

Nous avons décidé de gérer nos catégories sous la forme d'une liaison réflexive. Le choix de cette approche structurelle est dû à la prédominance des sous catégories liés au produits. En effet, nous voulions que nos articles puissent être facilement répertoriés selon les diverses besoins du consommateur. Ainsi cette approche nous permettait de lier chaque catégorie à une sous catégorie qui elle-même pouvait avoir des liaisons plus en profondeur. Ainsi un café Moka pouvait être associé à plusieurs sous-catégories de provenance à savoir : Afrique -> Ethiopie -> Moka

Cette approche nous a permis aussi d'ouvrir le champ des possibilités sur une potentiel deuxième version du site notamment à travers catégories principales des articles. Pour l'instant les catégories ont été réfléchies de la manière suivante :

Un article a une seule catégorie associée soit Dosette, Moulu ou Grain. Si un fournisseur permettait la possibilité à son produit d'être conditionné sous ces trois formes alors l'administrateur doit créer trois articles différents. Avec une liaison réflexive, il serait possible de les lier sur un même article.

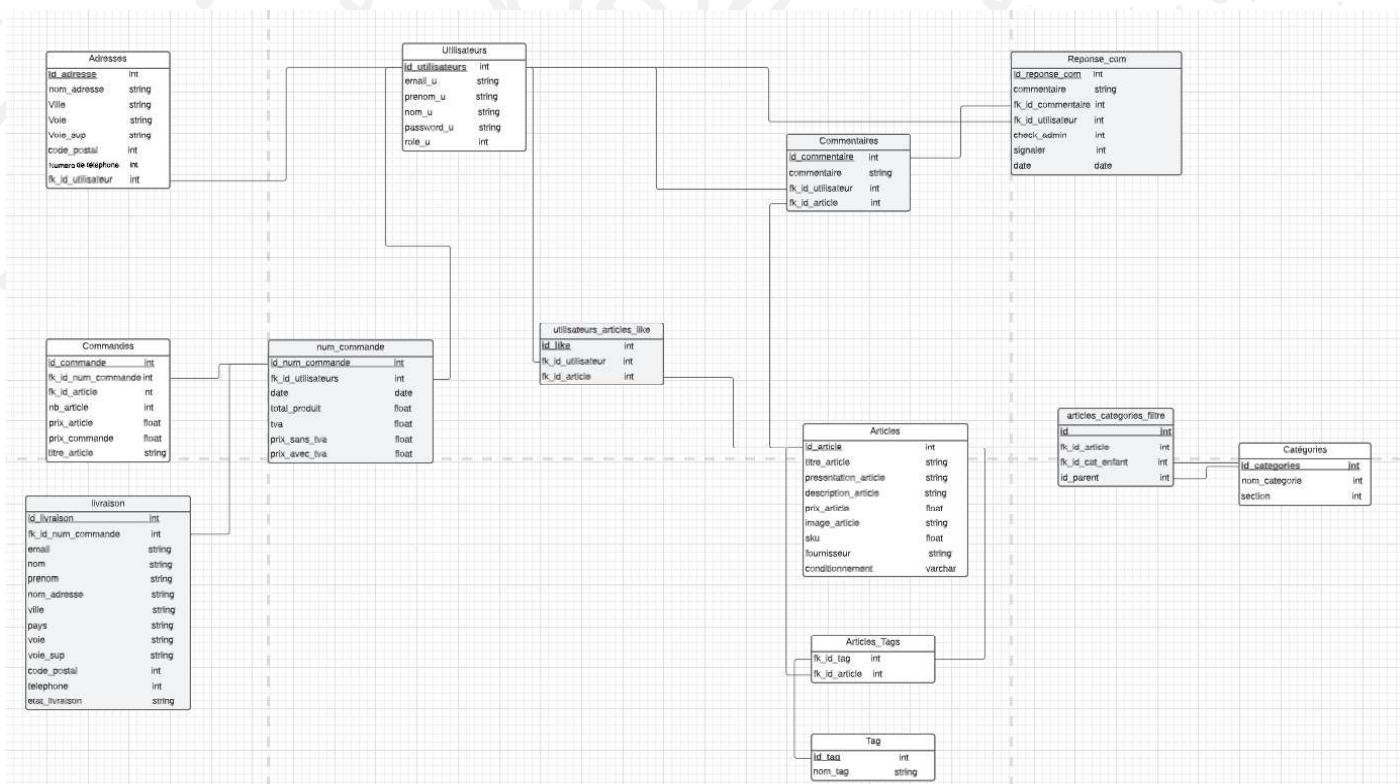
Les entités grise

Les entités grises sont des entités sur lesquelles nous avons eu une réflexion mais ne faisant pas parti intégrante de MVP. Nous verrons par la suite les modifications apportées sur ces tables.

Model Logique de Donnée

Le modèle logique des données consiste à décrire la structure de données utilisée sans faire référence à un langage de programmation. Dans le cadre de kawa cela nous a permis de définir plus précisément les tables qui seront appelées lors des requêtes et quel type de données devait être envoyé.

Model Logique de Donnée de Kawa



Réalisation de notre base de données sur notre SGBD

Une fois la synthèse de notre base de donnée établi nous sommes passé à son installation en utilisant MySQL comme système de gestion de bases de données relationnelles et l'interface phpMyAdmin permettant à travers son back office une bonne prise en main de cette réalisation.



Les contraintes de clefs étrangères

Nous avons commencé par définir nos contraintes de clé étrangère en leur donnant des règles sur leurs manières d'interagir entre elles. De cette manière nous installons déjà un schéma propice au bon fonctionnement et contrôle de nos données.

A screenshot of the phpMyAdmin interface showing the 'Contraintes de clé étrangère' (Foreign Key Constraints) page. It displays two entries under the 'Actions' tab. The first entry is for 'commentaires_ibfk_1' with 'ON DELETE CASCADE' and 'ON UPDATE RESTRICT'. The second entry is for 'fk_id_article_delete' with 'ON DELETE CASCADE' and 'ON UPDATE RESTRICT'. Both entries point to the 'utilisateurs' table with 'id_utilisateur' as the column. There are also fields for 'Base de données' (kawajyo), 'Table' (utilisateurs), and 'Colonne' (id_utilisateur). Buttons at the bottom include 'Aperçu SQL' and 'Enregistrer'.

Capture d'écran de l'interface de phpMyAdmin liés aux contraintes de clé étrangères

L'indexation

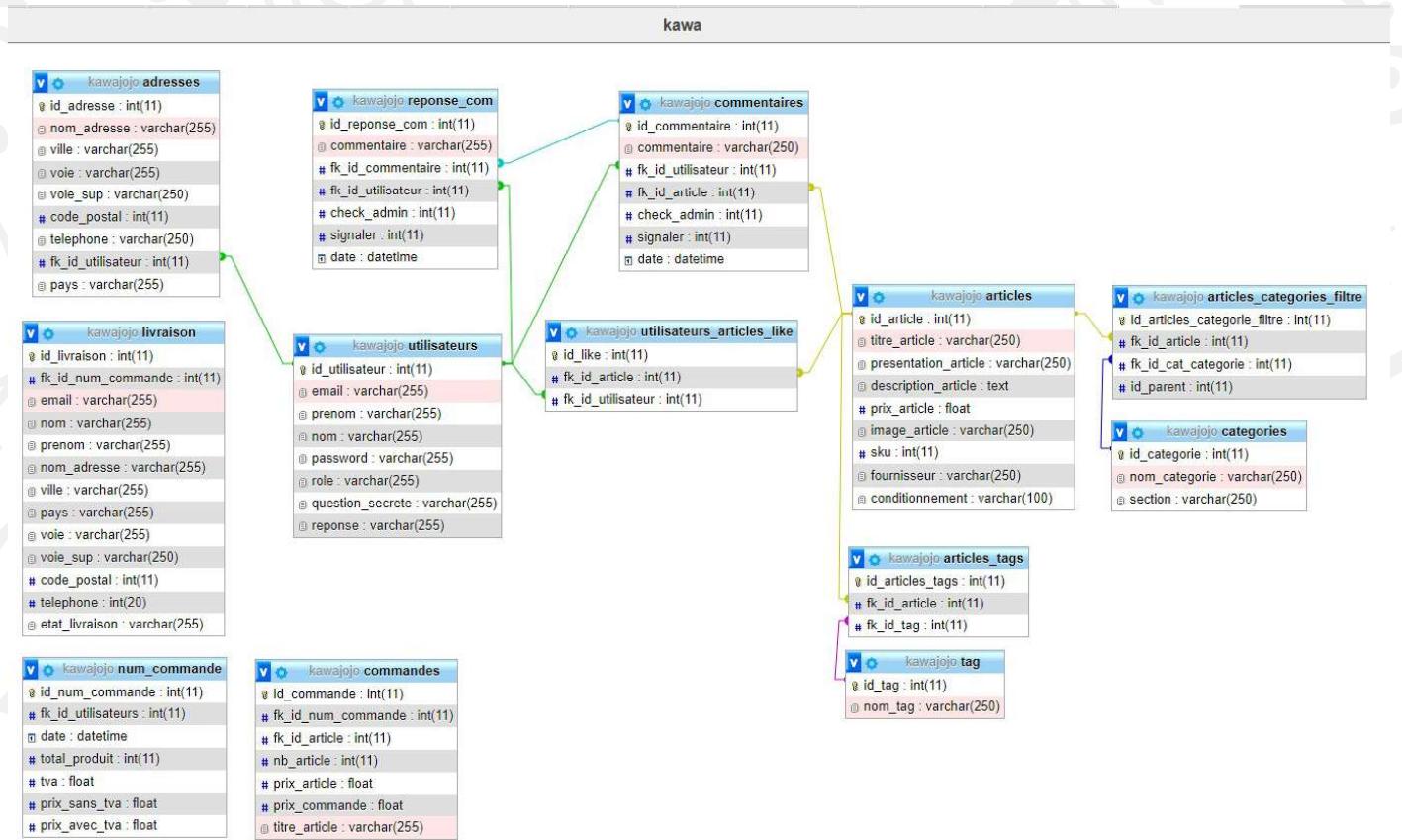
Nous avons aussi profité de cette étape pour organiser les différents éléments susceptibles d'être pertinent à la recherche. Ainsi nous avons défini certaines colonnes dont le contenu nous apparaît être un bon élément de recherche.

A screenshot of the phpMyAdmin interface showing the 'Index' (Indexes) page for the 'articles' table. It lists four indexes: 'PRIMARY' (BTREE, Unique Yes, Compressé Non, Colonne id_article, Cardinalité 23, Interclassement A, Null Non); 'titre' (FULLTEXT, Unique Non, Compressé Non, Colonne titre_article, Cardinalité 23, Interclassement Non); 'presentation' (FULLTEXT, Unique Non, Compressé Non, Colonne presentation_article, Cardinalité 23, Interclassement Non); and 'description' (FULLTEXT, Unique Non, Compressé Non, Colonne description_article, Cardinalité 23, Interclassement Non). Buttons at the bottom include 'Créer un index sur' and 'Exécuter'.

Capture d'écran de l'interface de phpMyAdmin liés aux indexs

Le concepteur

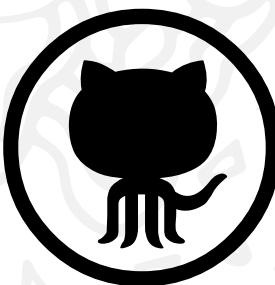
PhpMyAdmin nous aussi permis de vérifier l'état de nos liaisons et de l'architecture de nos données à travers son concepteur. Cela nous a permis de nous assuré d'être en concomitance avec notre conceptualisation.



Capture d'écran de l'interface du concepteur de phpMyAdmin

Mise en place de l'outil de versioning

Afin d'être plus efficace dans notre production nous avons décidé d'organiser notre manière de travailler. Cela passe notamment sur l'organisation d'un outils de versioning ou nous avons dupliqué une branche commune qui rassemblé les éléments communs.



Préparer en amont tout les assets du dossier

Dans un premier temps nous avons travailler ensemble sur prods de manière à pouvoir avoir notre disposition tout les éléments qui constituaient notre dossier. Ainsi, tout les dossier internet était créer ainsi que la mise en place de notre routeur et des namespace associé.

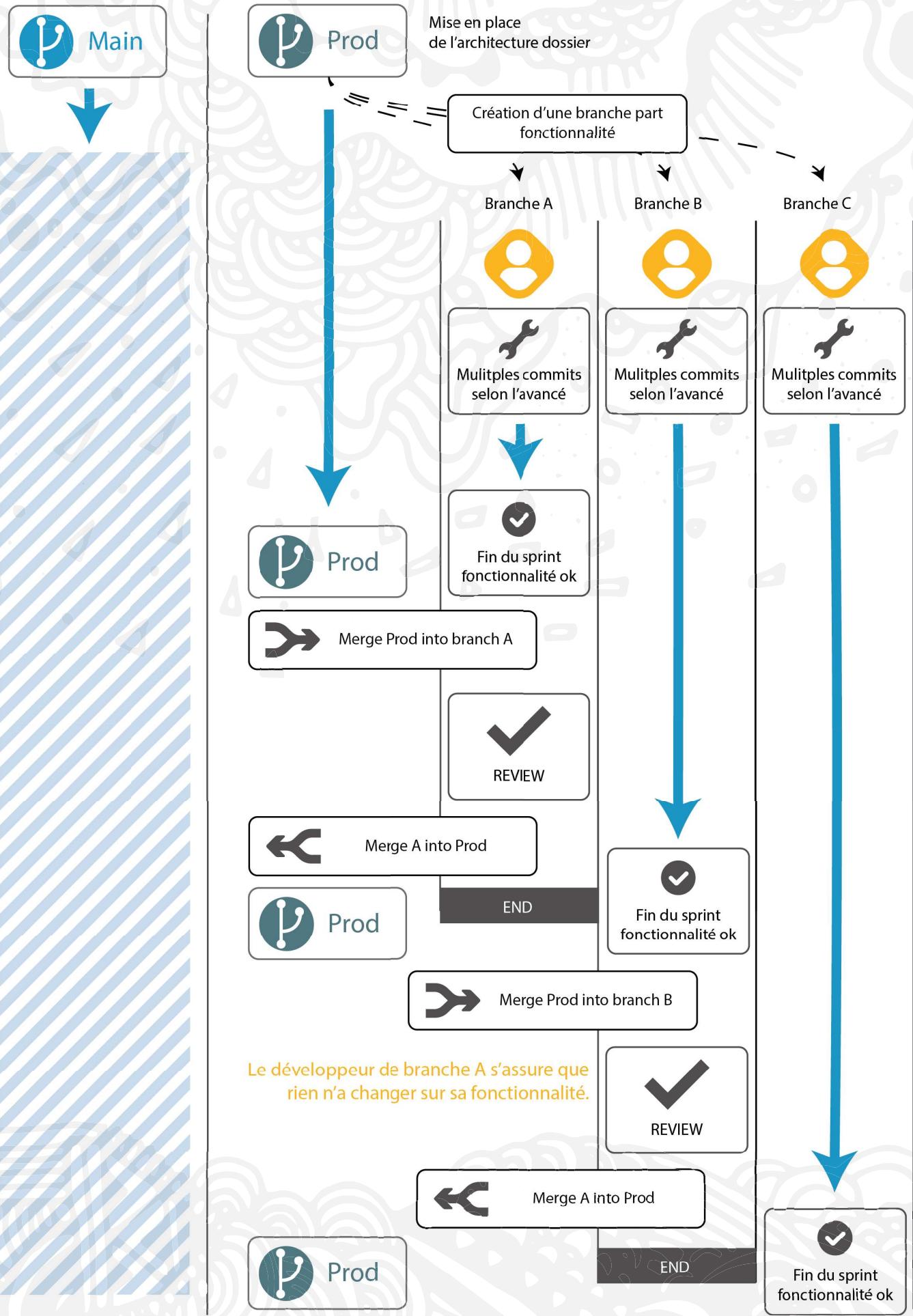
Organiser ces branches par fonctionnalité

Nous avons choisi d'organiser nos branche par fonctionnalités. De cette manière chaque développeur était libre d'ajouté tout les fichier liés à son travail.

Organiser des codes reviews à chaque fin de sprint

Pour s'assurer une cohérence dans le code, nous nous organisons de manière a ce que les merge puisse être suivit par l'ensemble des développeurs. Pour nous en assurer le développeur qui était chargé de la réalisation d'une fonctionnalité commencé a merge la branche la plus à jours, à savoir prod vers sa branche spécifique. Si il y avait des conflit alors il opérait de manière méticuleuse en prenant soin de demandé au développeur ayant travaillé sur le fichier à l'origine du conflit ce sur quoi il avait travaillé et ce qui pouvait être changé.

Ces reviews sont des étapes fondamentales, par exemple si un développeur c'est chargé de réaliser le CRUD catégorie, l'ensemble des développeur connaisse les méthode associé à l'objet et peuvent se permettre de le réutiliser correctement. A se titre nous comprenons l'importance de bien commenter son code afin de s'assurer d'une bonne lisibilité.



Définition de la stack technique

A présent que nous avons défini toute les étapes de conceptions, nous avons envisagé les manières le plus adapté à la réalisation de se projet et les différent langages nécessaire à la mise en œuvre de sa réalisation.

• **Php prédominent**

En vue du contexte dans lequel a été réaliser se projet, php se trouve prédominent. En effet, comme indiqué précédemment, ce projet est une réhabilitation d'un exercice ayant était fait en cours d'année. Lors de sa réalisation le programme était destiné à la compréhension de se langage de programmation.

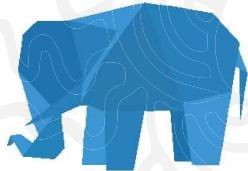
Utilisation de Composer

Nous avons décidé d'utiliser composer comme outil de gestion des dépendance afin de télécharger facilement les librairie nécessaire à notre projet. Notamment pour l'intégration du routeur et de l'autoloader.

Définir sa stack celons la manière de travailler

Nous avons défini la stacks en fonctions des besoin que nous avions sur le projet ainsi que la manière dont nous nous étions organiser et celons le temps qui nous était impartis. Ils nous paraissait utiles d'avoir un router pour avoir plus facilement l'oeil sur nos routes. Nous avons notamment décider de travailler selon le paradigme orienté objet afin que chaque développeur puisse jouer avec les données de manière fluide et en évitant les répétitions de code inutile.

De la même manières, nous avons décidé de structurer notre architecture de style selon la méthode 7.1 afin de segmenté les fichier en StyleSheet en plusieurs parties évitant ainsi les conflits lors des merges et les répétitions de code inutile pour même composants



PHP

Nous avons choisi d'utiliser principalement php pour l'ensemble du projet. Que ce soit au niveau de la génération de nos vues, du développement de nos controllers ou que ce soit nos requêtes en base de données avec nos models.



HTML5

Nos vue seront notamment coder en html et respecterons les normes définit par la W3C.



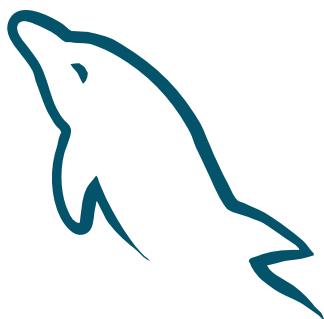
JavaScript

Nous utiliserons Javascript pour développer des composant nécessitant un algorithme devant s'exécuter sans rechargement de page tel que le panier ou l'auto-complétion



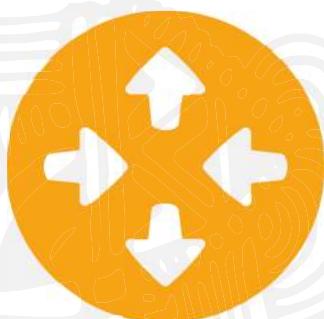
SCSS

Couplet à la méthode 7.1, Nous avons choisi d'utiliser SCSS comme méthode de script en StyleSheet. Cela nous a permis de travailler à plusieurs sur le style en évitant les conflits lors des merges.



MySQL

Nous avons décidé de gérer notre liaison en base de donnée avec MySql



AltoRouter

Nous avons fait le choix d'importer altorouter dans notre projet pour nous assurer router fluide et performant offrant une flexibilité par ses multiples options

Architecture du projet

Une fois la stack technique défini nous commençons à organiser notre architecture de dossier en vue de tout installer sur la branche Prod qui sera par la suite cloner sur toutes les branches du répertoire.

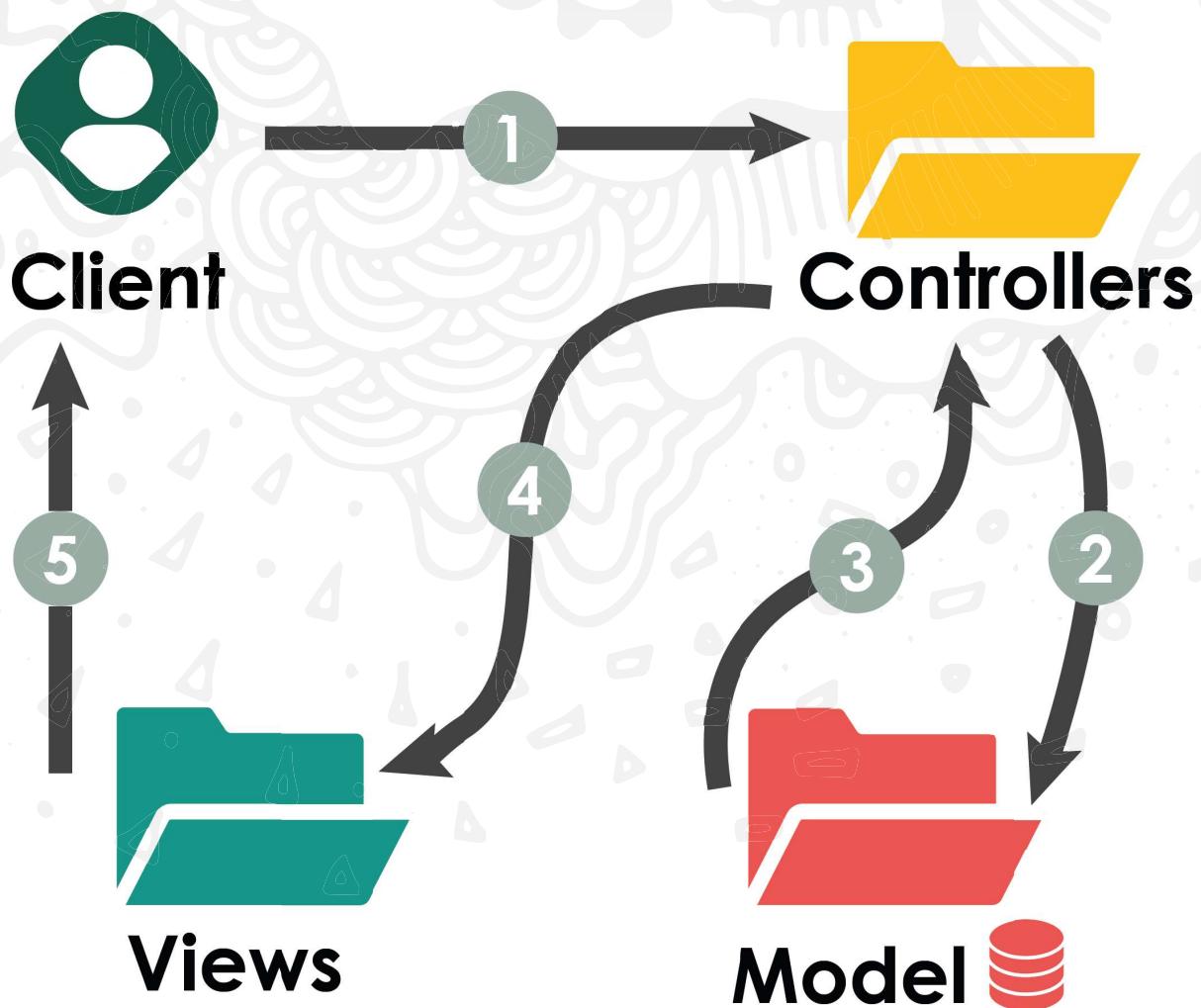
Modèle, Vue, Contrôleur

Afin de structurer au mieux notre projet nous avons décidé d'architecturer celui-ci selon le modèle MVC. Cette architecture découpe le code en trois parties distinct :

- Une partie modèle destinée à répertorier toute les requête en base de donnée
- Une partie Contrôleur qui contrôle et qui traite les informations en entrée et en sortie du modèle afin de s'assurer de répondre au mieux l'exécutif des méthodes métiers.
- Une partie Vue destiné a restitué les éléments nécessaire a l'utilisateur pour continuer sa navigation.

Schéma disponible à la page suivante

Schematisation de la concordances des élément selon l'architecture MVC



- 1 A la suite d'une interaction avec le client, la requête est envoyée côté serveur
- 2 Le contrôleur traite l'information et celons les conditions envoie les données traitées au Modèle
- 3 Le modèle exécute la requête préalablement programmé à la base de donnée. Et les renvoie au contrôleur
- 4 Une fois que le contrôleur à réceptionner et traité les données reçus par le modèle. Il renvoie les donnée traité à la vue
- 5 La vue ce charge de rendre ces donnée visible et compréhensible par le clients celons le retour du contrôleur

Programmation Orienté Objet

Pour que la méthode de structuration MVC soit la plus efficace, il quasiment obligatoire d'adopter une programmation orienté objets. Nous verrons donc par la suite quelques schéma montrant la manière que nous avons eu de liés les composants entre eux notamment dans le chapitre classe et héritage.



Controllers

Gestion des méthodes métier



Les components

Classes conçues pour être exploitées dans d'autres controllers.



Les classes de table

Classes de table directement liées à des tables en base de donnée.



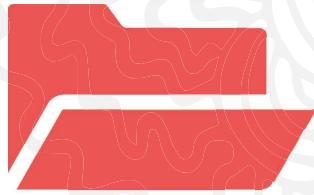
Les classes liées aux pages

Classes liées directement à la fonctionnalité d'une pages.

Les différents types de composants se trouvant sous le namespace Controller

Nous retrouvons trois type de composant majeurs appartenant au namespace Controllers :

- Nous retrouvons en premier les classes liées aux tables de données se trouvant en base de données. À travers ce composant nous allons retrouver certains traitements génériques pouvant être utiles dans de nombreux cas. La particularité de ces méthodes c'est qu'elles peuvent être directement liées à une page ou être appelées dans une autre méthode avec une structure plus définie afin de répondre au mieux au métier quelle desserte
- En deuxième nous retrouvons les controllers liées aux pages, ces composants ont une classe parente liée directement au router permettant de générer le rendu de la vue. Elles sont bien souvent composées de composants liés aux tables ou aux «components».
- En troisième nous retrouvons les components, ce sont des controllers très spécifiques liés à un certain rendu. Contrairement au composant de pages ceux-ci représentent une «molécule» au sens figuré. Ils forment en eux-mêmes un composant qui peut être appellé plusieurs fois : exemple = une carte de produits



Model

Gestion des requêtes envoyées



Model Mère

Classes comprenant des requête génériques.



Les classes liées aux pages

Classes liées directement à une fonctionnalité de pages

Les différents types de composants se trouvant sous le namespace Model

Nous retrouvons deux type de composant appartenant au namespace Model :

- Le modèle mère dont le constructor instancie la base de donnée en PDO. Celle ci à la particularité d'avoir des requête générique préparée correspondant au CRUD(create,read,update,delete) classique. Ce modèle est majoritairement étendue aux autres modèle.
- Les classe liées au page ou nous allons retrouver les geter et les seter permettant leurs hidratations ainsi que les requête très spécifique.



Views

Gestion de l'affichage de l'information



Layout

Fichier principal englobe les retours dynamiques



Component

Fichier constamment appellé sur toute les pages



Views classique

Fichier appeler selon l'url

Les différents types de composants se trouvant sous le namespace Views

Nous retrouvons trois type de composant majeurs appartenant au namespace Views :

- Le Layout, le layout est la vue parente à toute les vue. Celle-ci se compose principalement de Components qui vont être appelé sur chaque page comme le header, le panier et le footer. Elle comprends en son sain les balise méta nécessaire à la lecture de l'entête par le navigateur.
- Les components comme stipulé plus haut sont des élément que nous allons pouvoir retrouver sur plusieurs pages ou qui seront appellés constamment. Il peut s'agir d'élément de pagination, du panier ou de la barre de recherche, etc...
- Les vues classique sont les pages structurer en html qui vont être hydrater par le controller. Ces fichiers représentent pour la plupart la structure classique que va prendre la page en relation avec les informations envoyer. Elle ne comporte que trois peut d'algorithme

Le Router

Le router nous a permis de nous organiser plus facilement la gestion de nos appelle de fichier tout nous permettant un indexation des routes facilitant la compréhension du mécanisme.

Le router se comporte de manière similaire à la structure mvc synthétisé dans le chapitre précédent. Nous pouvons le comparé à un chef d'orchestre. Le développeur lui donne des instructions sur lesquels le routeur va se basé pour exécuté un certain nombre de commande.

Précisément nous allons définir une route/un chemin celons lequel le routeur s'exécutera en cas de consultation.

Cette route est associé un contrôleur qui sera appeler.

Se contrôleur appellera ainsi un contrôleur parent souvent nommé le render ou le renderer qui va se charger d'envoyé une vue associée à la vue parente.

La route permet aussi de définir le type de donnée que nous attendons en paramètre cela accorde une sécurité supplémentaire à notre application. De plus le contrôleur peut nous permettre de nettoyer l'url de manière a ce que certaine informations ne puisse t'être donnée à un utilisateur potentiellement mal veillant

```
77  /* ----- INTERFACE USER -----  
78  ----- INTERFACE USER ----- */  
79  
80  $router->map(  
81      'GET|POST',  
82      '/inscription',  
83      function () {  
84          $controller = new App\Controllers\InscriptionController();  
85          $controller->index();  
86          $controller->SignUp();  
87      },  
88  );
```

Exemple de route défini

Dans l'exemple ci-dessus nous pouvoir les différent paramètres nous permettant de définir le comportement du routeur. Ainsi nous pouvons voir que le controller ConnexionController est instancier lorsque l'url répond à <https://nomdedomaine/connexion>. Celui-ci s'attendra à pouvoir réceptionner des information en POST ou en GET

Classes & héritages

Comme nous l'avons vu sur les chapitres précédents, c'est sans surprise que je vais détaillé l'organisation des classe et de leurs héritages. Nous avons déjà parler de Model Mère lors de l'explication de l'architecture dossier ainsi que du Controller Render lors de l'explication du routeur

Les attributs et leurs états

La création peut être vue comme un modèle comportant des directive qui vont donné du sens à notre objets. Cette objets se définira donc selon les attributs qui le définisse ainsi que sont comportements. Il se caractérise par trois phénomène propre à la programmation orienté objet :

Le constructeur

Le constructeur est un comportement appelé méthode qui s'exécutera automatiquement à l'instanciation de notre objet. Il a pour rôle de poser les base nécessaire à la création de notre objet. En php le constructeur n'est pas obligatoirement défini, si celui-ci n'est pas défini alors php générera un constructeur par défaut. Dans le cas inverse il faut prendre en compte que le constructeur est obligatoirement une méthode publique, c'est à dire que ce certain paramètre nécessaire à son fonctionnement peuvent être modifiable. Il faut donc faire très attention à bien controller les arguments qu'il pourrait recevoir.

Les attributs

Les attributs, comme leurs noms l'indique sont les élément qui vont définir notre objet en soit. Par exemple pour tout les contrôleurs liés à une table en base de données nous allons retrouver la liste complète de leurs attributs. Ainsi pour un utilisateur nous retrouverons son adresse mail, son nom, son prénom, etc. Ces attributs sont en liens directe avec nos modèle, c'est pourquoi il faut faire très attention au données qu'il peuvent contenir. Pour s'assurer d'une sécurité supplémentaire les attribut son généralement protégé. L'état protected les défini donc comme pouvant être modifier seulement par un méthode appartenant à la classe en elle même soit par une classe donc celle-ci se verrait étendue.

Les méthodes particulières avec lesquels nous transmettons cette informations son nomé des geter (pour ce qui les récupère et qui les lire) et les seter (pour ce qui les modifie). Il se peut notamment qu'un attribut soit privé lorsque celui-ci contient un donnée sensible et que celui-ci est propre au fonctionnement de la classe.

Les Méthodes

Les méthodes sont un corpus de fonction spécifique à la classe. Là aussi nous pouvons les définir sous différents état, il y a les méthode privée qui sont clos à la classe et qui ne peuvent être appellé que par celle-ci. Les méthodes protected qui peuvent t'être transmise à une classe qui en hérite et les méthode publique comme nos geter et nos seter qui ont un scope plus large et qui peuvent nous permettre un interaction extérieur.

Dans notre projet nous avons eu à faire à deux type d'héritage spécifique. Le premier est le plus commun. Nous avons notre contrôler de page qui hérité du Render. Celui-ci se voit instancier un Modèle représentant un table liés directement au modèle mère. Dans cette combinaison classique notre router se charge de tous les appeler et l'instanciation à la base de donnée se fait automatiquement. La deuxième situation concerne les composant, n'étant pas destinée à être appeler sur une page unique il ne bénéficie pas par défaut de l'héritage du Render. Dans ce genre de cas il faut alors manuellement dire à notre classe d'instencier la connexion à la base de donnée en héritant du constructeur parent.

Pour cela il suffit de faire appel à une classe static native dans php : parent::__construct();

```
1 <?php
2
3 namespace App\Models;
4
5 class Utilisateurs extends Model
6 {
7     protected $table = 'utilisateurs';
8     protected $id = "id_utilisateur";
9     protected $id_utilisateur;
10    protected $email;
11    protected $prenom;
12    protected $nom;
13    protected $password;
14    protected $role;
15    protected $question_secrete;
16    protected $reponse;
17
18    /**
19     * Get the value of id_utilisateur
20     */
21    public function getId_utilisateur()
22    {
23        return $this->id_utilisateur;
24    }
25
26    /**
27     * Set the value of id_utilisateur
28     *
29     * @return self
30     */
31    public function setId_utilisateur($id_utilisateur)
32    {
33        $this->id_utilisateur = $id_utilisateur;
34
35        return $this;
36    }
37
```

Dans l'exemple suivant nous pouvons constater en ligne 3 que la classe appartient au namespace App\Models donc que cette classe est destiné à envoyer de requête à notre base de donnée.

Nous pouvons constater en ligne 5 quel hérite de toute les méthode protected ou public du Model Mère nommée Model.

De la ligne 7 à la ligne 16 nous pouvons voir la définition de cette objet à travers les attributs. Tous sont en protected, ce qui veut dire que leurs modifications pourra être fait que par une méthode qui en hérite ou à travers ces propre méthode.

Et nous retrouvons à partir de la ligne 18 les geter qui permette l'hydratation de ces informations.

Composant métier

Les composants métier sont des fonctionnalités qui ont été spécifiquement programmé pour répondre au besoin de l'application. Elle se définissent par leurs aspect «sur mesure» et adapté au condition d'utilisation. Dans le cas de notre boutique en ligne nous avons deux familles principales de composant métier. A savoir, le système de tri et de recherche de l'information par l'utilisateur. Et le système de gestion de l'information par l'administrateur.

Dans les quelques lignes que j'accorde à cette importance, je vais présenter quelques exemple typique des composant métier que nous avons programmé pour l'application Kawa. Afin d'en faciliter la lisibilité je vais m'attarder sur un composant particulier.

Le système de recherche avancé par filtre de catégorie

Comme indiqué lors de la conception, il nous tenait à cœur de permettre à l'utilisateur de pouvoir trouver un produit adapté à sa demande en quelques clics. Pour ce faire nous avons vu plus haut que nous avions choisi de gérer les catégories avec un liaison réflexive.

Sur ce point là, je ne vais pas m'attarder sur la manière dont elle sont créées bien que ceci correspond bien à une fonctionnalité métier, mais plutôt de la manière dont elles peuvent être utilisées par le client.

```
1  <?php
2
3  namespace App\Controllers;
4
5  use App\Models\Search;
6  use Database\DBConnection;
7  use App\Controllers\Components\CardComponent;
8  use App\Controllers\Components\ProductComponent;
9  use App\Controllers\Components\CategoriesComponent;
10
11 class BoutiqueSearchController extends Controller
12 {
13     public $error = array();
14     protected $Product;
15     protected $Categories;
16     protected $search;
17     protected $FooBag;
18
19     public function __construct()
20     {
21         $this->Product = new ProductComponent();
22         $this->Categories = new CategoriesComponent();
23         $this->search = new Search();
24         $this->FooBag = new ProductController();
25
26         $this->error;
27     }
}
```

Capture d'écran de la classe de page appellée par le router.

Nous pouvons voir de la ligne 5 à 9 les composants appeler pour son fonctionnement.

Nous pouvons voir que nous instancions les attributs de cette classe par les composants qui nous serviront.

Nous pouvons constater dans le screen suivant que nous instention un tableau dans lequel nous faisons appel à une méthode d'un controller qui va nous fournir un liste de données répertorieant toutes les sous-catégories créer par l'administrateur, elle-même paramaitré par des section par défaut comme «principale», «variétés», «spécificité», «saveur», etc...

```

91     /**
92      * On récupère toute les catégories pour soumettre le formulaire à l'utilisateur
93     */
94     $result_request = array(
95         'principale' => $this->Categories->chooseCategoriesBySection(['section'], 'PRINCIPALE'),
96         'variете' => $this->Categories->chooseCategoriesBySection(['section'], 'VARIÉTÉ'),
97         'specificite' => $this->Categories->chooseCategoriesBySection(['section'], 'SPÉCIFICITÉ'),
98         'flavor' => $this->Categories->chooseCategoriesBySection(['section'], 'SAVEUR'),
99         'strong' => $this->Categories->chooseCategoriesBySection(['section'], 'FORCE'),
100        'origin' => $this->Categories->chooseCategoriesBySection(['section'], 'PROVENENCE')
101    );

```

Nous pouvons présumé en vue du commentaire laisser en dessus de la fonction que celle-ci servira à alimenter un formulaire mise à disposition de l'utilisateur. Mais nous reviendrons dessus dans la parti dédier au front. Nous avons là un bon exemple de controller qui envoie l'information venue du modèle à la vue afin de l'alimenté en donnée et/ou en contenu.

Dans l'exemple suivant nous pouvons voir les conditions d'exécution de la recherche de filtre, nous commençons par définir les condition d'exceptions qui pourrait entraîné le bon déroulement de la recherche

```

59     if (isset($_GET['filterDelete'])) {
60         unset($_SESSION['filter']);
61     }
62     $result = $this->mergeCattoProduct(); //On merge les tables produit avec la tables catégorie
63
64     $this->getSelection();
65     if ($param != 'all') //Si une ategorie autre que all a été selectionner on trie le tableau selon la categorie principale
66     {
67         $result = $this->Product->selectArrayByValue($result, 'cat parent', $param);
68     }
69
70     if (!empty($_POST['deleteFilter'])) //Si le Post delet est utiliser on unset la valeur ciblé de notre varaiable de section
71     {
72         $deleteValue = explode('-', $_POST['deleteFilter']);
73         if (count($deleteValue) == 3) {
74             unset($_SESSION['filter'][$deleteValue[0]][$deleteValue[1]]);
75         } else unset($_SESSION['filter'][$deleteValue[0]]);
76     }
77     if (!empty($_POST['filtre'])) //Si un nouveau formulaire de recherche avancé est soumis alors on ecrasé ou on créer une vriable de section
78     {
79         $_SESSION['filter'] = $_POST;
80         unset($_SESSION['filter']['filtre']);
81         $resultFilter = $_SESSION['filter'];
82         $result = $this->gestionfilter($result);
83     } elseif (!empty($_SESSION['filter'])) //Si la vraible de section existe déjà alors on fait une nouvelle recherche selon les changement
84     {
85         $resultFilter = $_SESSION['filter'];
86         $result = $this->gestionfilter($result);
87     } else $resultFilter = null; //Si rien est effectuer on assigne nul à la variable de section pour eviter les erreurs
88
89     $card = new CardCompenent(); //On instancie nos card pour l'envoyer dans le render

```

Nous pouvons constater dans ce passage que dès la soumissions du formulaire par l'utilisateur nous instancions ou modifiions une variable de session afin trier un tableau de résultat. L'utilisation de la variable de session nous permet d'enregistrer les demande fait au préalable afin que l'utilisateur puisse renseigner plus d'information sur son choix ou au contraire permettre d'enlever certaines catégories

```

180  /**
181  * Permet de générer un nous veauable multidimensionsnel de résultat
182  * en le triant par les les variable de session définit
183  * @param array Le tableau de résultat original
184  */
185 public function gestionFilter(array $result)
186 {
187     if (isset($_SESSION['filter']['VARIÉTÉ']) && count($result) >= 1) {
188         $result = $this->Product->selectArrayByValue($result, 'VARIÉTÉ', $_SESSION['filter']['VARIÉTÉ']);
189     }
190     if (isset($_SESSION['filter']['SPÉCIFICITÉ']) && count($result) >= 1) {
191         foreach ($_SESSION['filter']['SPÉCIFICITÉ'] as $value) {
192             if (count($result) < 1) {
193                 return $result;
194             } else {
195                 $return = $this->Product->selectArrayByValue($result, 'SPÉCIFICITÉ', $value);
196                 if (!empty($return)) {
197                     $result = $return;
198                 } else {
199                     $this->error = "Le filtre de la spécificité " . $value . " n'a pas aboutir à un résultat, nous vous proposons cependant les articles suivant :";
200                     return $result;
201                 }
202             }
203         }
204     }
205     if (isset($_SESSION['filter']['SAVEUR']) && count($result) >= 1) {
206         foreach ($_SESSION['filter']['SAVEUR'] as $value) {

```

Dans la méthode présenter dans la page précédente nous avons pu voir que je faisait appel à une autre méthode spécifique à la classe nommé **gestionFilter**. Cette fonction a pour but de filtrer les données apporté par la variable de session des filtres de manière à ne retenir que les objets contenant l'information spécifier. Nous voyons notamment que ce tableau ce transmet de conditions en conditions.

Cela s'explique part la volonté d'intégré un système de poids à la recherche. Ainsi si la l'utilisateur a filtrer sa recherche selon une variété en particulier et un spécificité tel que café Arabica et Bio. Alors j'accorde arbitrairement plus de poids à la variété qu'à la spécificité. A chaque changement je consulte le nombre d'élément constituant mon tableau. Si celui-ci ne comprend pas suffisamment de résultat alors je lui retourne le dernier filtre exécuté et je lui laisse un message afin que l'utilisateur soit informé que sa requête n'a pu totalement aboutir

Prévoyance des failles de sécurité

Tout ces appelles de fonctions peuvent être une menace pour l'intégrité des données personnelles de l'utilisateur et de la sécurité du site web. Pour prévenir une intrusion nous avons choisi de placer un middleware au niveau de la route qui nous assure d'échapper tout les caractères spéciaux qui pourraient être présent dans les entrées.

Faille d'injection SQL ou XSS

```
41 // Sécurité de tout les formulaire Get|POST
42 $securityAll = new Security();
43 if (isset($_GET)) {
44     $securityAll->controlAll($_GET);
45 }
46 if (isset($_POST)) {
47     $securityAll->controlAll($_POST);
48 }
49
```

```
public static function controlAll($compact)
{
    if (is_array($compact)) {
        foreach ($compact as $key => $value) {
            // On regarde si le type de string est un nombre entier (int)
            // If (ctype_digit($value) && is_array($value)==false) {
            //     $value = intval($value);
            // }
            // Pour tous les autres types
            // else {
            if (is_array($value)) {
                foreach($value as $underKey => $underValue) {
                    if (ctype_digit($underValue)) {
                        $underValue = intval($underValue);
                    }
                    else{
                        $underValue = strip_tags($underValue);
                        $underValue = htmlentities($underValue);
                        $underValue = htmlspecialchars($underValue);
                    }
                }
            }
            else {
                if (ctype_digit($value)) {
                    $value = intval($value);
                }
                else{
                    $value = strip_tags($value);
                    $value = htmlentities($value);
                    $value = htmlspecialchars($value);
                }
            }
        }
    }
    // Pour tous les autres types
    else {
        $compact = strip_tags($compact);
        $compact = htmlentities($compact);
        $compact = htmlspecialchars($compact);
    }
}
```

Appel de la méthode de sécurité des entrées dans la page index avant le lancement du routeur

Détail de la fonction appeler

Faillle par l'URL

Nous nous assurons notamment que l'utrl consulter par l'utilisateur répond bien à ces droit. L'exemple suivant comprend un contrôle en variable de session non haché. Cependant ce genre d'information sensible doivent répondre au normes de sécurité.

```
25 if ($pathControl[2] == 'admin' && $_SESSION['user']['role'] != 'Admin') {  
26     if (isset($_SESSION['user'])) {  
27         // echo 'redirection';
```

Faile par l'upload de fichier

Nous devons notamment sécurisé tout intrusions dans le dossier. Cela est notamment possible lors de l'upload d'un fichier. Par exemple une image. Afin de s'assurer d'aucune entrée malveillant nous limitons et vérifions les types de fichiers.

```

public function stock_picture(string $chemin = '/boutique-en-ligne/public/assets/pictures/pictures_product/', ?string $lastName = null)
{
    if ($this->verify_upload('image_article') == true) {
        //Verification de l'extention du fichier reçu
        $explode_file = explode(".", $_FILES['image_article']['name']);
        $extention = ['jpeg', 'jpg', 'JPEG', 'JPG'];
        $approuve = false;
        foreach ($extention as $value) {
            if ($value == $explode_file[1]) {
                $approuve = true;
            }
        }
        if ($approuve == false) {
            $_SESSION['flash']['format'] = "L'image doit être en jpeg";
        }
    }

    if ($approuve == true) {
        //si approuver, traitement du fichier
        if (empty($lastName)) {
            $explode_file[0] = uniqid(); //Renomage du fichier avec une string unique
        } else $explode_file[0] = $lastName; //Sinon on garde le nom choisi en param
        $explode_file[1] = ".{$explode_file[1]}"; //Ajout du dole avant concatenation
        $_FILES['image_article']['name'] = $explode_file[0] . $explode_file[1]; //Concataination du nom et de l'exention
        $miniature = $_FILES['image_article']['name'];

        //Modification de la taille de l'image
        $taille = getimagesize($_FILES['image_article']['tmp_name']); //Traitement de la largeur et de la hauteur d'origine du fichier
        $largeur = $taille[0];
        $hauteur = $taille[1];
        $largeur_minihure = 720; //Definition de la nouvelle largeur
        $hauteur_minihure = $hauteur / $largeur * 720; //Definition de la nouvelle hauteur relative à la largeur
        $im = imagecreatefromjpeg($_FILES['image_article']['tmp_name']); //Creation d'une nouvelle image dans la memoire tampon
        $im_minihure = imagecreatetruecolor($largeur_minihure, $hauteur_minihure); //creation d'un gabarit selon les dimentions defini
        imagecopyresampled($im_minihure, $im, 0, 0, 0, 0, $largeur_minihure, $hauteur_minihure, $largeur, $hauteur); //redimensionnement de l'image à la taille de l'image
        imagejpeg($im_minihure, $chemin . $_FILES['image_article']['name'], 90); //Création de l'image dans le dossier assigner

        $this->image_article = $_FILES['image_article']['name'];

        return $this->image_article;
    } else $this->error['image'] = 'Assurez-vous que l\'image soit bien en jpg,jpeg.';

    return null;
}

```

Nous pouvons voir dans l'exemple ci-joint que nous nous assurons du poids du fichier, du format de celui-ci. Nous renommons le fichier, et nous réons une seconde image dans lequel l'image uploader va être intégrer afin de s'assurer que ce bien une image

Gestion des requêtes

La gestion des requêtes vers notre base de données est un élément central dans la gestion d'une application web. Les requêtes se doivent d'être préparées en avance et sécurisées. Pour s'assurer de cela nous avons décidé d'utiliser PDO. Cela nous permet une sécurité de plus en préparant nos requêtes à l'avance et en les exécutant juste après l'exécution.

Nos requêtes sont préalablement testées sur le back office de phpMyAdmin afin de s'assurer de la validité de celle-ci :

The screenshot shows the phpMyAdmin interface with the following details:

- Header:** "Affichage des lignes 0 - 7 (total de 8, traitement en 0,0021 seconde(s))."
- SQL Query:**

```
SELECT `categories`.`nom_catégorie`, `categories`.`id_catégorie`, `categories`.`section` FROM `articles_categories_filtre` AS `intermediaire` INNER JOIN `categories` ON `intermediaire`.`fk_id_article` = `categories`.`id_catégorie` WHERE `intermediaire`.`fk_id_article` < 28;
```
- Buttons:** Profilage, Éditer en ligne, Éditer, Explicquer SQL, Créez le code source PHP, Actualiser.
- Table Headers:** "Tout afficher", "Nombre de lignes : 25", "Filtrer les lignes", "Chercher dans cette table", "Trier par clé", "Aucun(e)".
- Table Data:** A table with columns "nom_catégorie", "id_catégorie", and "section". The data includes:

nom_catégorie	id_catégorie	section
Décafiné	19	SÉPÉCITÉ
Fleuri	14	SAVEUR
Rond	15	SAVEUR
Équilibré	16	SAVEUR
4	7	FORCE
Réunion	25	PROVENANCE
Arabica	12	VARIÉTÉ
Biologique	18	SÉPÉCITÉ
- Buttons at the bottom:** "Tout afficher", "Nombre de lignes : 25", "Filtrer les lignes", "Chercher dans cette table", "Trier par clé", "Aucun(e)".
- Bottom Navigation:** Opérations sur les résultats de la requête, with links: Imprimer, Copier dans le presse-papiers, Exporter, Afficher le graphique, Créez une vue.

Gestion des composants

Nous pasons maintenant à la gestion des composant. Dans ce chapitre j'aborde les composant liés à un script permettant à un contenue dynamique à remonter dans le DOM sans rechargement de la page.

Nous pasons maintenant à la gestion des composant. Dans ce chapitre j'aborde les composant liés à un script permettant à un contenue dynamique à remonter dans le DOM sans rechargement de la page.

```
1 <?php
2     $input = json_decode(file_get_contents('php://input'), true);
3
4     require '../models/database.php';
5     $Database = new Database;
6
7
8     if(isset($input['value']))
9     {
10         $valueName = htmlentities($input['value']);
11
12         $resultBegin = $Database->selectResultSearchBar($valueName, 'begin');
13         $resultUnder = $Database->selectResultSearchBar($valueName, 'under');
14
15
16         foreach ($resultBegin as $key => $value) {
17
18             for ($i=0; $i < count($resultUnder); $i++) {
19
20                 if(array_key_exists($i,$resultUnder) && $value['nom'] == $resultUnder[$i]['nom'])
21                 {
22                     unset($resultUnder[$i]);
23                 }
24             }
25         }
26
27         $result = ['begin' => $resultBegin, 'under' => array_values($resultUnder)];
28         $return = json_encode($result);
29         echo $return;
30     }
31     else{
32         echo 'pas ok';
33     }
34
35 ?>
```

Dans l'exemple ci dessus nous pouvons voir une partie de scripte destiné à récupérer les informations retourné qui sera retourné par l'autocompletion

Script js permettant de récupérer la valeur de l'input

```
1 window.addEventListener("DOMContentLoaded", (event)=>
2 {
3     /**
4      * Définition du concepte d'application de l'autocompletion sur la search bar
5      * 1/ Le formulaire doit avoir une role définit comme "search"
6      *     - Le formulaire ne comporte pas de submit ou de boutton
7      *     - Le formulaire comporte un input type search
8      * 2/ il doit comprendre un input search portant le nom "recherche"
9      * 3/ Une div vide doit être placer directement en dessous de la bar de recherche
10     *
11     * /!\ Aucune Classe ni ID n'est à définir cependant pour l'accessibilité il peut être défini sur le container adjacent /!\
12     */
13     let inputSearch = document.querySelector('input[name = recherche]')
14     let containeResultSearch = document.querySelector('#searchResult');
15     let indexSelect = -1;
16     let interruptSearch = false
17
18     /**
19      * Change la casse en capitale du premier caractère sur une chène donnée
20      * @param {*} a chaîne de caractère
21      * @returns la chaîne de caractère donnée avec un capitale
22      */
23     function strUcFirst(a)
24     {
25         return (a+'').charAt(0).toUpperCase()+a.substr(1);
26     }
27
28     /**
29      * Fonction spécifique
30      * Crée une liste ordonnée des nom des atomes selon des data définies est l'insert dans un conteneur parent choisi
31      * @param {*} data data sous forme json / Doit comporté le 'nom' & 'id' de l'élément
32      * @param {string} position Défini le comportement le comportement de la casse selon sa position dans la recherche :
33      *     - 'under'= 'le champ de recherche s'applique à l'intérieur du nom'
34      *     - 'begin'= 'le champ de recherche s'applique au début du nom'
35      * @param {*} kinship Défini le parent qui contiendra la liste des éléments
36      */
37     function creatNewListName(data, position, kinship)
38     {
39         ulResultSearch = document.createElement('ul')
40         let icrementTab = 0;
41         data.forEach(atome => {
42             let li = document.createElement('li');
```

SCSS & 7.1

Nous pensons maintenant à la gestion des composant. Dans ce chapitre j'aborde les composant liés à un script permettant à un contenu dynamique à remonter dans le DOM sans rechargement de la page.

Comme nous pouvons le voir, tout est dans un gros répertoire sass . Au niveau supérieur du répertoire se trouve un seul fichier main.scss (il s'agit du fichier d'entrée) et sept sous-répertoires :

abstracts ne contient pas de styles réels , juste des mécanismes Sass qui aident à définir des styles dans d'autres répertoires (parfois appelés «helpers»). Cela inclut des éléments tels que les variables globales, les fonctions et les mixins. Chacune de ces catégories doit être placée dans son propre fichier partiel nommé de manière appropriée, comme indiqué ci-dessus.

vendors contient toutes les feuilles de style tierces utilisées par un projet. Par exemple, si nous voulions utiliser Bootstrap avec notre propre Sass personnalisé dans un projet, nous téléchargeions la feuille de style Bootstrap et la placions ici.

la base contient le passe-partout utilisé sur l'ensemble d'un site. Cela inclut les styles de typographie à l'échelle du projet et les feuilles de style qui réinitialisent ou normalisent universellement le CSS par défaut.

layout contient des styles pour différents aspects de la mise en page structurelle du site (pensez à des zones telles que les barres de navigation, les en-têtes, les pieds de page, etc.)

les composants sont comme des «mini» mises en page. Les styles pour les petites parties réutilisables du site doivent résider ici (pensez aux boutons, aux formulaires, aux images de profil, etc.)

pages est l'endroit où résident les styles spécifiques à la page. Par exemple, si un projet contenait plusieurs règles de style qui ne sont jamais utilisées que sur la page «Contactez-nous» , elles vivraient ici dans un fichier _contact.scss , comme indiqué ci-dessus.

thèmes est utilisé chaque fois qu'un site a plusieurs thèmes. Par exemple, l'exemple de projet ci-dessus inclut à la fois les thèmes d'administration et par défaut. Nous pouvons donc supposer que cet exemple de site a un style entièrement différent pour les administrateurs connectés. Peut-être pour mieux présenter et intégrer les fonctionnalités supplémentaires dont dispose un administrateur. Certains sites Web proposent également un «mode nuit», où l'arrière-plan du site est plus sombre avec du texte de couleur plus claire pour une lecture plus facile dans des environnements peu éclairés. Une option comme celle-ci serait également représentée dans son propre fichier de thème.

Nous avons adapté ce système de hiérarchie à nos besoins :

..		
base	fix footer	
components	fix front	
layout	fix header nav wireframe	
pages	fix msg erreur page produit commentaire	
utils	front ok sur admin : cat, comment, tag, mise en place burger coulissant	
main.css	fix header nav wireframe	
main.css.map	fix header nav wireframe	
main.scss	Customisation de la page 404	

Nous pouvons voir dans le dossier composants que ceux-ci ont un fichier par éléments :

..		
README.md	mise en place design pattern 7-1	4 months ago
_button.scss	fix front	3 months ago
_card.scss	Modification du hover sur les cards	3 months ago
_carrousel.scss	page home 80% done	3 months ago
_errorPopup.scss	fix gestion erreur couleur	3 months ago

```
190
191 @media only screen and (min-width: 1280px) {
192     .product {
193         // background-color: red;
194         display: flex;
195     }
196     .test {
197         // background-color: blue;
198         padding: 0 48px;
199         flex: 3;
200     }
201     &__image {
202         height: auto;
203         width: 100%;
204     }
205     .payment {
206         display: flex;
207         align-items: center;
208         justify-content: space-between;
209         margin: auto;
210         width: 100%;
211     }
212 }
213
214 .test {
215     fill: #135e4c;
216 }
```

Responsiv design & médiaQuerys

Nous avons choisi de concevoir l'interface avec une approche mobile first selon le RWD(responsive web design). Afin d'accroître la fluidité de l'utilisateur j'ai choisi d'adapter le placement des éléments selon leurs utilisations. Ainsi les éléments les plus utilisés se trouvent dans la zone de confort en mobile et se replacent selon la mise en page communément admise lors de l'utilisation sur Desktop.

Pour cela, nous avons opté pour une approche avec 3 Breakpoint, l'idée étant de permettre au site une expérience fluide couvrant l'ensemble des Devices :

inférieur à 720px
de 720px à 1024px
supérieur à 1024px

image 1 : capture d'écran de la page boutique de kawa sur display iphon SE

image 2: capture d'écran de la même page sur display un laptop

Nous pouvons constater que la sidebarre permettant le tri par filtre est en menu déroulant sur les Device inférieur à 1024px pour se déployer en position fixe lors de la navigation sur Desktop.

De la même manière, nous retrouvons la barre de navigation fixée au bottom pour la positionner dans un zone accessible au pouce alors qu'en Desktop elle va retrouver sa place habituelle au niveau du Header.

Nous avons choisi de travailler sur une architecture SCSS en 7.1 afin de coordonner plus facilement les fichiers de travail et de s'assurer aucun conflit de Merge lors de la mise en production. Nous avons donc travaillé selon les principes de l'atomic design en s'assurant de définir tous les états de chaque composant.

L'utilisation de wrapper nous a permis de flexibiliser la mise en page à notre guise tout en conservant des notions d'accessibilité. Nous avons notamment suivi les recommandations de la W3C et utilisé unités syntaxiques appropriées à leurs sémantiques. L'usage correct de ces balises permettant un meilleur référencement naturel sur la SERP (Search Engine Result Page).

```
190
191 @media only screen and (min-width: 1280px) {
192     .product {
193         // background-color: red;
194         display: flex;
195     }
196     .test {
197         // background-color: blue;
198         padding: 0 48px;
199         flex: 3;
200     }
201     &__image {
202         height: auto;
203         width: 100%;
204     }
205     .payment {
206         display: flex;
207         align-items: center;
208         justify-content: space-between;
209         margin: auto;
210         width: 100%;
211     }
212 }
213
214 .test {
215     fill: #135e4c;
216 }
```

exemple de médiaquery