

Compte rendu WEB

Site marchand - PHP, Twig

ISIWEB4SHOP Boissons Biscuits Fruits secs



Bienvenue !

Vous voilà sur le site ISIWEB pour acheter du café.



Boissons

Profitez de notre large gamme de boissons !



Biscuits

Commandez nos biscuits cuisinés par nos soins !



Fruits secs

Découvrez notre rayon fruits secs !

TOUPENCE Tom - PETEL Noa



Année Universitaire 2023 - 2024

I. Organisation et répartition des tâches	3
1. Répartition des tâches:	3
2. Organisation Git	3
II. Réalisation du projet	3
1. Architecture du projet	3
2. Fonctionnalités principales	3
3. Difficultés rencontrées	3
4. Amélioration possible	3

I. Organisation et répartition des tâches

1. Répartition des tâches:

Étant donné que ce projet est notre premier en PHP, l'organisation initiale a été brouillon. Notre organisation GIT a été difficile jusqu'à ce que nous décidions d'appliquer la méthode **Git Flow**. 2

Pour vous résumer la répartition du tâche, nous allons donc résumer les différentes fonctionnalités que nous nous sommes répartis dans un tableau où on fera la différence entre la période de début de projet, où notre travail n'a pas été très indépendants, et l'instant où nous avons mis en place Git Flow et que nous avons séparés distinctement les tâches.

	Noa	Tom
Début du projet	Routeur	Création du front: layout, page principale, bande latérale
Git Flow	Panier Commande Facture Paiement	Inscription Connexion/Déconnexion Admin (panneau de commande) Gestion du layout

Pour préciser le tableau, il est important de préciser que chaque fonctionnalité nécessite généralement la création d'une vue via Twig, d'un contrôleur ainsi que d'un accès à la BD.

2. Organisation Git:

Pour ce projet, nous avons fait le choix d'utiliser l'outil de gestion de version GIT et notamment la méthode Git Flow de manière simplifiée (notre projet n'étant pas très conséquent, certaines branches non pas été utiles).

La méthode Git Flow est une approche de gestion de versions basée sur Git qui simplifie le développement collaboratif et la gestion des versions du projet. Elle

repose sur des branches principales, notamment **`master`** pour les versions stables et **`develop`** pour le développement en cours.

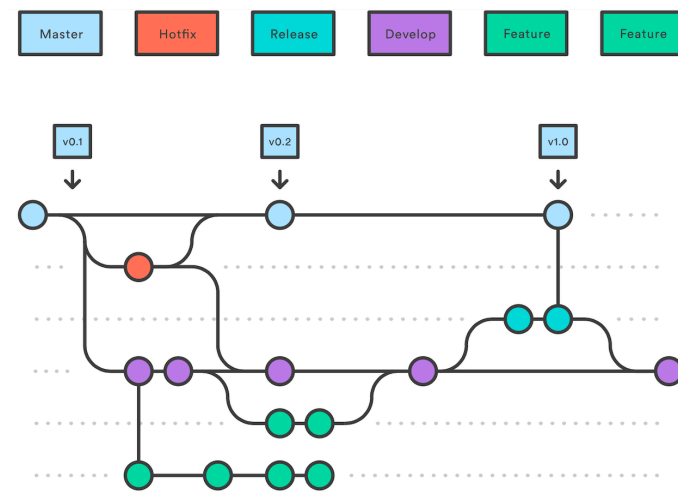
Les branches secondaires sont les branches **`features`** (pdf, connexion, panier sont des branches features sur notre git). Ces branches isolent le développement des fonctionnalités et permettent donc leur développement simultanés en parallèle. Une fois implémentées, celles-ci sont merge sur la branche **`develop`** où les potentiels conflits sont résolus.

Grâce à cette méthode, la collaboration est favorisée et les interférences potentielles sont minimisées afin d'éviter des debug inutiles.

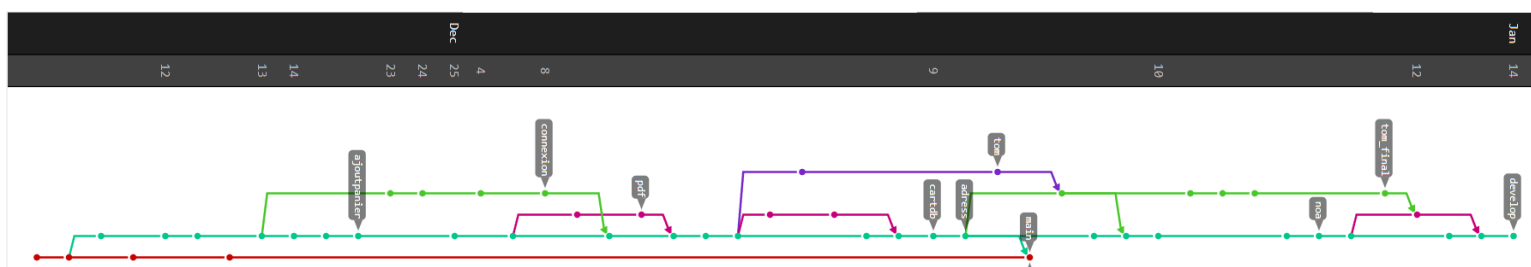
Nous avons fais le choix de ne pas utiliser les branches **`hotfix`** et **`release`** car notre projet n'était pas assez conséquent pour en voir une utilité directe.

Voici une comparaison d'un graphe de dépôt Git Flow théorique et le notre à la fin du projet:

Repository graph Git Flow



Our repository graph











On retrouve les branches suivantes: **Main**, **Develop**, ainsi que les différentes Features

II. Réalisation du projet

1. Architecture du projet:

Nous avons utilisé pour notre projet l'architecture MVC qui permet de diviser une application en trois composants principaux : le modèle, qui gère les données et la logique métier ; la vue, responsable de l'interface utilisateur et de l'affichage des données ; et le contrôleur, qui traite les interactions de l'utilisateur et orchestre la communication entre le modèle et la vue. Cette approche favorise la séparation des différentes couches métier, d'améliorer la fixation des bugs et l'implémentation de nouvelles fonctionnalités.

	controller	14/01/2024 11:56	Dossier de fichiers	
	model	14/01/2024 11:56	Dossier de fichiers	
	public	10/01/2024 18:57	Dossier de fichiers	
	vendor	10/01/2024 18:57	Dossier de fichiers	
	view	14/01/2024 11:56	Dossier de fichiers	
	composer.json	13/12/2023 17:11	Fichier source JSON	1 Ko
	composer.lock	12/12/2023 20:05	Fichier LOCK	9 Ko
	README.md	12/12/2023 20:03	Fichier source Mar...	7 Ko

Voici l'arborescence finale du projet:

On retrouve bien les dossiers de l'architecture MVC, ainsi qu'un dossier public contenant toutes les images, icônes et documents générés ou utilisés et le dossier `vendor` contenant les dépendances du projet.

2. Fonctionnalités principales:

Pour le projet, nous avons dû implémenter les fonctionnalités principales d'un panier, de la gestion d'un utilisateur mais également un routeur.

- *Routeur:*

Afin de router l'utilisateur, nous passons par des variables dans l'URL. Le routeur se charge de récupérer la valeur contenu dans la variable ``page`` et redirige vers le bon contrôleur pour l'affichage de la page.

Par exemple, lorsqu'on clique sur le bouton ``Boissons`` dans le menu. Nous cliquons en réalité sur un `href` qui change l'url et ajoute la variable `"public/?page=boissons"`. Le routeur appelle le *contrôleur* **ProductController** qui se charge d'afficher la page **product.twig** de la *Vue* avec les données renvoyées par le script **data.php** du *Modèle*. Le contrôleur.

- *Panier et utilisateur:*

Dans l'idée, le panier et l'utilisateur se gère pratiquement à l'identique: l'utilisation de la variable `$_SESSION` pour retenir les informations et l'utilisation des différents scripts du modèle pour faire le lien avec la base de données. Pour cela, il suffit d'initialiser avec la fonction `session_start()` qui nous permet d'utiliser les variables de sessions.

Ainsi, lorsque l'on clique sur "Connexion", nous sauvegardons l'identifiant de session de l'utilisateur afin de conserver les informations avec lesquelles il interagit, comme par exemple la sauvegarde du panier à travers les pages.

3. Difficultés rencontrées:

Nous avons évidemment rencontré de nombreuses difficultés durant le projet. Nous allons vous détailler les plus importants et comment nous avons réussi à les surmonter.

- *Le routeur:*

La création du routeur a été l'une des premières difficultés. La question de comment réaliser le routage a été difficile à résoudre et a nécessité de nombreux entretiens avec les professeurs. Nous avons à ce moment pris en main la variable **\$_GET** et l'utilisation de variable dans l'url.

Après coup, il s'avère que notre routeur n'est pas le plus propre et surtout le mieux implémenté. L'idéal aurait été de lire l'URL et de bien séparer les contrôleurs et les actions appelés plutôt que de passer par des variables. C'est donc un point à améliorer dans le futur.

- *La connexion utilisateur*

Créer le système de connexion était une autre difficulté où il fallait prendre connaissance de l'utilisation des variables de sessions. Pouvoir récupérer les données avec lesquelles l'utilisateur interagissait n'était pas facile car il fallait vérifier à chaque fois sur chaque page si le panier se conservait par exemple, ou même si l'on récupérerait bien toutes les informations à partir du profil administrateur.

4. Amélioration possible:

Par la suite, pour améliorer notre projet, nous pourrions implémenter de nouvelles fonctionnalités:

- *La conservation du panier lorsque l'utilisateur se déconnecte et la fusion des paniers entre non connecté et déconnecté.*

Cela permettrait à l'utilisateur de récupérer la commande qu'il souhaitait faire à un autre moment. De plus, si l'utilisateur fait un panier en étant déconnecté et qu'il décide de se connecter, une fonctionnalité devrait être mise en place pour fusionner le panier associé au compte avec le panier créé lors de la session non connectée.

- *Mettre en place un système d'abonnement*

En effet, on aurait pu mettre en place un abonnement qui permettrait à l'utilisateur d'avoir accès à des avantages exclusifs comme la livraison plus rapide, ou alors pouvoir recevoir des codes de promotions.

- *Mise en place d'un reçu de la facture par mail*

Au lieu de télécharger en local sur un PC la facture, il aurait été possible d'envoyer directement un mail avec en pièce jointe la facture.

- *Formulaire de contact*

On aurait pu implémenter un formulaire de contact qui permettrait à l'utilisateur d'envoyer un mail pour poser une question ou alors faire une réclamation.

- *Un espace personnel*

Un espace personnel pourrait permettre à l'utilisateur de modifier ses informations et son mot de passe.

- *Choix de livraison en point relais*

L'utilisateur pourrait avoir le choix de se faire livrer en point relais à l'aide des propositions que nous pourrions lui faire selon la zone où il habite (Besoin d'une DB qui répertorie ces points relais).

Sur le projet actuel, il reste quelques petits détails à régler, notamment sur le frontend. Nous pensons notamment à améliorer la redirection en cas d'erreur lors de la connexion :

- Redirection vers une page de déconnexion
- Créer une vraie page en cas de connexion échouée
- Améliorer la page d'erreur