Milestone 2

Master 2 MIAGE - UPJV Amiens - 2024/2025
Louiza ACHIR - Damien FROIDURE - Lylia ADANE - Lilian GANDEBOEUF - Bastien DREUX

Sommaire

Introduction

Cette phase du projet, correspondant au Milestone 2, marque une étape essentielle dans l'avancement de notre application. Elle a été l'occasion pour notre équipe de structurer nos idées, d'organiser nos actions et de poser des bases solides pour la suite du projet dans le but de faire un point clair et accessible sur ce que nous avons accompli jusqu'à présent et sur les choix qui ont guidé nos décisions.

L'idée est de présenter de manière claire et concise les bases sur lesquelles nous nous appuyons pour avancer dans ce projet. Vous y trouverez :

- Une vue d'ensemble de notre organisation et de notre planification ;
- Les raisons qui ont guidé nos choix technologiques ;
- Un aperçu de la manière dont nous collaborons et gérons les versions grâce à GitHub
- Les réussites et les difficultés rencontrées

L'objectif est d'assurer une transparence totale sur la progression et la gestion du projet, tout en posant les bases nécessaires pour les étapes futures, conformément aux attentes des parties prenantes et aux exigences académiques.

Ce rapport constitue ainsi une synthèse complète des travaux réalisés dans le cadre du Milestone 2 et reflète notre engagement envers une approche rigoureuse et collaborative, garantissant le succès final du projet.

Contexte (Rappel du sujet):

Ce projet a pour objectif de concevoir et de développer un site web e-commerce pour un mini-supermarché qui ne dispose actuellement pas de plateforme en ligne.
Ce site a pour vocation de répondre aux nouvelles attentes des clients, qui recherchent des solutions modernes, rapides et pratiques pour effectuer leurs achats. En permettant aux clients de consulter les produits disponibles, de créer un panier, de payer en ligne via carte bancaire, et de récupérer leurs commandes directement en magasin, ce projet vise à améliorer l'expérience client et à renforcer la compétitivité de l'enseigne sur le marché local.

Outre les fonctionnalités dédiées aux clients, le site inclura également des outils de gestion destinés aux préparateurs de commandes et à l'administrateur. Les préparateurs de commandes auront la possibilité de suivre l'état des commandes, de signaler les éventuelles ruptures de stock et de mettre à jour leur statut. De leur côté, les administrateurs disposeront d'un tableau de bord leur permettant de gérer efficacement le catalogue de produits, les stocks, les commandes, ainsi que les comptes des utilisateurs internes.

Ce projet s'inscrit donc dans une démarche globale de digitalisation, visant à moderniser les services du supermarché tout en simplifiant la gestion quotidienne des opérations.

Présentation de la conduite de projet

Nous avons structuré notre projet en plusieurs phases clés réparties sur le calendrier, en fonction des échéances des Milestones et des tâches critiques. Le diagramme de Gantt reflète cette planification détaillée.

Dans le cadre de la gestion de notre projet, nous avons adopté une structure en plusieurs phases, chacune correspondant à des jalons clés. Ces phases ont été planifiées en tenant compte des échéances importantes et des tâches critiques à accomplir. Le processus de gestion du projet suit une méthodologie en cascade pour la phase initiale, puis une approche agile pour les phases suivantes afin de garantir la flexibilité et l'adaptabilité face aux imprévus et aux retours du client.

1. Méthodologies adoptées

1.1. Méthodologie en cascade pour le Milestone 1

Pour le Milestone 1, qui correspond à la rédaction du cahier des charges et à la validation des besoins du client, nous avons choisi une méthodologie en cascade. Cette approche est adaptée pour cette première étape, car elle permet de structurer clairement la collecte des besoins avant d'entamer les développements. Durant cette phase, nous avons organisé des réunions internes régulières entre les membres de l'équipe pour faire le point sur l'avancement et assurer la bonne compréhension des attentes du client. Cela nous a permis de centraliser les informations, d'échanger sur les besoins spécifiques et de nous assurer que chaque membre était aligné sur la direction à suivre.
En parallèle, une réunion avec le client a été programmée pour valider les besoins recueillis et confirmer que la vision du client était bien représentée dans le cahier des charges. Cette réunion a été cruciale pour valider les fonctionnalités à intégrer, les priorités et la direction générale du projet. Le cahier des charges ainsi rédigé constitue la base de toutes les décisions suivantes, garantissant que le projet répondra aux attentes du client.

1.2. Méthodologie agile pour la phase de développement

Au Milestone 2, nous avons fait un pivot méthodologique vers une approche agile. Cette transition permet de passer d'une planification rigide à une gestion plus souple, permettant une évolution continue du projet. L'agilité nous permet d'être réactifs aux retours du client et d'ajuster nos développements au fur et à mesure.
Dans cette phase, des réunions mensuelles avec le client seront mises en place. Ces réunions servent à valider les fonctionnalités livrées, recueillir des retours détaillés et ajuster les priorités en fonction des nouvelles exigences ou des améliorations nécessaires. Cela garantit que le projet reste aligné avec les besoins du client et permet d'intégrer les retours de manière continue, sans perturber le rythme du développement.

2. Répartition des tâches

La répartition des tâches pour ce milestone a été répartie efficacement au sein de l'équipe. Bastien DREUX a pris en charge la rédaction des documents techniques, incluant la conduite de projet et la configuration de GitHub. Lilian GANDEBOEUF s'est occupé de l'ajout de fonctionnalités dans GitHub, notamment la gestion du Gantt, tandis que Louiza ACHIR a travaillé sur la finalisation de la maquette Figma. Damien FROIDURE a configuré GitHub en expliquant la stratégie mise en place, et Lylia ADANE a assuré la mise en place de l'environnement sur Git et la gestion de la planification en réalisant le diagramme de Gantt.
Au fil des semaines, bien que certaines tâches aient pris du retard, l'équipe a su maintenir une bonne coordination. La prise de recul et la révision des documents ont été partagées entre tous les membres, et la mise en place du projet Git a facilité la collaboration. L'équipe a ainsi avancé efficacement vers la préparation du dépôt final et de la soutenance prévue.

Par la suite, dans le cadre de la phase de développement afin d'assurer une gestion efficace du projet, les tâches ont été réparties entre les membres de l'équipe en fonction de leurs compétences et de leurs disponibilités. Cette répartition est visible dans le diagramme de Gantt, qui permet de suivre l'avancement de chaque tâche et de garantir que chaque membre de l'équipe est responsable d'un ensemble d'objectifs spécifiques.
Nous avons veillé à répartir les tâches de manière équitable afin que chaque membre de l'équipe consacre un temps similaire au projet. Cependant, cette répartition n'a pas été effectuée de manière aléatoire, mais en prenant en compte les **compétences de chacun** en développement web ainsi que la **complexité des tâches** et des fonctionnalités à implémenter. Par exemple, certains membres, ayant une expertise plus marquée dans certains domaines comme la gestion des bases de données ou l'implémentation de fonctionnalités spécifiques, ont été assignés à des tâches plus techniques ou plus complexes. De plus, la **vélocité** de chaque membre a été prise en compte, ce qui signifie que nous avons évalué le temps que chacun pouvait consacrer en fonction de son emploi du temps et de ses autres engagements. Cette approche nous a permis de distribuer les tâches de manière à ce que personne ne soit surchargé tout en maximisant l'efficacité du groupe.
Afin de garantir une répartition optimale, nous avons organisé une réunion au début du projet pour discuter des tâches et des responsabilités. Cette réunion nous a permis de clarifier les compétences de chaque membre et d'évaluer la difficulté de chaque fonctionnalité. Cela a conduit à une répartition équilibrée qui tient compte à la fois des compétences, du temps disponible et de la complexité des tâches.

Voici un résumé des tâches réparties :

Membres	Tâches

Lilian	• CRUD Compte (avec gestion des rôles)
	  ○ Ajouter un compte
	  ○ Modifier un compte
	  ○ Supprimer un compte
	  ○ Consulter un compte
	• Connexion
	• Réinitialiser le mot de passe
	• Notification de limite de stock atteint
	• Plan d'accès au magasin
	• Visualiser les commentaires google

Louiza	• Rechercher un produit
	• Filtrer les produits
	• Trier les produits
	• CRUD panier
	  ○ Ajouter au panier
	  ○ Modifier le panier
	  ○ Supprimer le panier
	  ○ Consulter le panier
	• CRUD Promos
	  ○ Ajouter une promotion
	  ○ Modifier une promotion
	  ○ Supprimer une promotion
	  ○ Consulter une promotion

Damien	• CRUD Catégories
	  ○ Ajouter une catégorie
	  ○ Modifier une catégorie
	  ○ Supprimer une catégorie
	  ○ Consulter une catégorie
	• Visualiser la politique de confidentialité
	• Mettre à jour la politique de confidentialité
	• Visualiser CGV
	• Mettre à jour CGV
	• Page d'accueil du site (+header, + footer)

Lylia	• Mise en place de l'environnement de développement
	• Modéliser la base de données
	• CRUD Commandes
	  ○ Ajouter une commande
	  ○ Modifier une commande
	  ○ Supprimer une commande

Bastien	○ Consulter une commande ● Effectuer un paiement en ligne ● Mettre à jour l'état d'une commande
	● CRUD Produits ○ Ajouter un produit ○ Modifier un produit ○ Supprimer un produit ○ Consulter un produit ● Suivre l'état d'une commande ● Notification de validation de commande ● Gestion des stocks (déduire lors d'une commande)
Toute l'équipe	● Préparation du rendu Milestone 2 ● Réunion hebdomadaire d'équipe ● Réunion mensuelle avec le client

3. Planification

La répartition des tâches mentionnée précédemment repose sur un **diagramme de Gantt** utilisé comme base pour la planification initiale. Ce diagramme détaille les différentes étapes et tâches à réaliser, en les attribuant aux membres de l'équipe selon les fonctionnalités à développer. Il comprend également des créneaux réservés aux tâches communes, telles que la rédaction des documents, la préparation des rendus écrits et les soutenances orales. Chaque membre de l'équipe est identifié par une couleur spécifique, tandis que les tâches collectives sont indiquées en **violet**, offrant ainsi une vue d'ensemble de la répartition du travail.
Cependant, cette planification initiale ne constitue qu'un **point de départ**. En effet, nous avons opté pour une approche **agile**, qui privilégie la flexibilité et l'adaptabilité tout au long du projet. L'objectif de cette méthodologie est de permettre une approche **itérative**, dans laquelle la planification peut être ajustée en fonction des retours et des imprévus. Dans cette optique, le diagramme de Gantt joue un rôle fondamental en posant les bases du projet et en offrant une vision globale initiale. Toutefois, il nécessite d'être régulièrement révisé afin de s'ajuster aux évolutions du travail, aux obstacles rencontrés et aux ajustements nécessaires pour garantir la réussite du projet.
La méthodologie agile encourage également une **communication constante** et un suivi rapproché de l'avancement. Chaque semaine, l'équipe se réunit pour faire le point sur le travail accompli, identifier les éventuels blocages et ajuster les priorités en fonction des besoins. Ces réunions permettent de maintenir une dynamique collaborative forte et assurent une réactivité rapide face aux défis qui peuvent surgir. Par ailleurs, des **réunions mensuelles avec le client** sont organisées pour lui présenter les progrès réalisés, recueillir ses retours et s'assurer que le projet reste aligné avec ses attentes, tout en intégrant de potentielles modifications ou nouvelles demandes.

Un des piliers de notre méthodologie agile est l'utilisation de **Git**, un outil permettant de suivre la progression du travail de manière transparente. Les tâches définies dans la planification sont transformées en **"issues"** sur notre dépôt Git. Chaque membre de l'équipe est responsable de fermer les issues correspondant aux tâches qui lui sont attribuées, une fois celles-ci terminées. Ce système non seulement permet de suivre précisément l'avancement du projet, mais aussi de maintenir une gestion de tâches dynamique et collaborative. Les **issues Git** servent ainsi de base à la gestion du projet, en étant mises à jour en temps réel, ce qui reflète les ajustements effectués en fonction de l'évolution du travail et des retours des membres de l'équipe et du client.
Ainsi, bien que la planification initiale soit essentielle pour amorcer le projet et structurer son organisation, elle n'est en aucun cas figée. Grâce à la méthodologie agile, la planification devient un processus évolutif, permettant de répondre aux défis du projet de manière flexible et réactive. Le diagramme de Gantt, loin de se limiter à un outil rigide, se transforme en un guide qui oriente le projet tout en restant adaptable à chaque étape du développement. Ce processus dynamique, soutenu par les **issues Git**, assure une gestion optimale de l'avancement et de la collaboration tout au long du projet.

Lien du Gantt :
https://docs.google.com/spreadsheets/d/1eLlswQkPx4B55V_3aDrUHNUNIWk7E65coZOR-VGe1_A/edit?usp=sharing

4. Jalons Clés

La réalisation de ces travaux est soumise à des **dates limites** strictes, ce qui implique que la planification des rendus doit être ajustée en fonction de ces délais.
La planification de la phase de développement a également été mise en place de manière à **respecter ces échéances**. Cela signifie que chaque étape du développement est organisée de manière à permettre des **marges de manœuvre suffisantes** pour la révision, les tests et la validation avant les dates de rendu. Ce processus nous assure que nous serons prêts à livrer le produit dans les délais impartis tout en maintenant la qualité du travail réalisé.

Milestone 2 (22 janvier 2025) :

- Livraison des scénarios d'utilisation et des maquettes.
- Préparation de la présentation de la conduite de projet et justification des choix techniques.
- Mise en place des environnements de développement.

Milestone 3 (30 juin 2025) :

- Démonstration complète du produit.
- Tests fonctionnels et techniques.
- Mise en production.

5. Outils utilisés

GitHub Projects : Utilisé pour suivre les issues et l'avancement des tâches tout au long du projet.

Git : Outil de versionning et de partage du code permettant de travailler de manière collaborative sur le projet.

JIRA (anciennement) : Utilisé au début du projet pour le suivi des tâches, avant de migrer vers GitHub Projects pour une gestion plus fluide.

Figma : Employé pour la création et le design des maquettes du projet.

Google Drive :Utilisé pour le partage des documents de travail et la collaboration en ligne. Sa suite d'outils nous a permis de créer et gérer nos documents efficacement :

Google Doc : Pour la création de document écrit
Google Sheets : Pour la réalisation de tableur ou de gantt notamment
Google Slides : Pour toutes création de présentation

Présentation des choix techniques

Dans le cadre du développement de notre application, nous avons dû faire des choix techniques en fonction des besoins du projet, de nos compétences et des spécificités de chaque technologie.

1. *Framework de Développement*

Pour le développement de l'application, plusieurs frameworks PHP et technologies front-end ont été envisagés :

Critères	Symfony	Laravel	React + API REST
Performance	**++** Excellente gestion des requêtes et optimisation des performances.	**+** Bonne performance, mais moins optimisé pour les projets complexes.	**+** Nécessite une configuration API REST poussée.
Extensibilité	**++** Grande modularité grâce aux bundles Symfony.	**+** Bonne extensibilité via les packages Laravel.	**++** Extensible, mais complexité accrue pour les API.
Sécurité	**++** Fonctionnalités natives pour la sécurité (XSS, CSRF).	**+** Sécurité intégrée mais moins flexible que Symfony.	**-** Nécessite des solutions complémentaires pour sécuriser les APIs.
Coût humain	**++** Familiarité de l'équipe avec Symfony.	**+** Facilité de prise en main, mais courbe d'apprentissage nécessaire.	**-** API REST complexe pour notre équipe.
Documentation	**++** Documentation complète et une large communauté.	**++** Documentation robuste.	**+** Bonne documentation, mais plus dispersée.

Choix retenu : Symfony.
Symfony est un framework PHP full-stack particulièrement adapté pour développer des sites web structurés et évolutifs. Il se distingue par sa modularité, sa flexibilité et son

architecture MVC (Modèle-Vue-Contrôleur), qui facilite la séparation des responsabilités et améliore ainsi la maintenabilité du code à long terme. Ce framework offre également des fonctionnalités robustes pour la gestion des bases de données, la mise en cache, le débogage et la gestion des erreurs, ce qui en fait un choix idéal pour un projet web à grande échelle.

Un des principaux critères de choix de Symfony est la familiarité de notre équipe avec ce framework. En effet, plusieurs membres de l'équipe possédaient déjà une expérience avec Symfony, ce qui nous a permis de gagner un temps précieux dans la phase de développement, sans avoir à investir dans une formation technique complexe pour adopter une nouvelle technologie. De plus, l'utilisation d'un autre framework, tel que **React** avec une API REST, aurait impliqué une complexité accrue et une courbe d'apprentissage plus longue, notamment en raison de la gestion d'une API, qui n'était pas maîtrisée par l'équipe. Ce choix de Symfony garantit ainsi une transition fluide et une meilleure productivité.

2. Système de Gestion de Base de Données

Pour la gestion des données, deux principales solutions ont été comparées :

Critères	PostgreSQL	MySQL
Performance	**++** Performant pour des applications complexes, mais moins optimisé pour des projets légers.	**+** Adapté aux applications légères à moyennes, rapide et efficace pour des requêtes simples.
Extensibilité	**+** Supporte des données complexes et des extensions, mais plus complexe à configurer.	**++** Facilement extensible avec des solutions adaptées pour des projets de taille moyenne.
Sécurité	**+** Fonctionnalités de sécurité avancées, mais souvent plus difficiles à configurer et à gérer.	**++** Sécurité de base suffisante pour des projets standards, avec des options de sécurité avancées.
Compatibilité	**++** Support multi-plateforme et compatibilité avec Symfony.	**++** Compatible avec Symfony et bien documenté.

Choix retenu : MySQL avec PhpMyAdmin
Nous avons choisi **MySQL** en raison de sa simplicité, de sa large adoption et de sa bonne compatibilité avec **Symfony**, facilitant ainsi le développement de l'application. Bien que **PostgreSQL** soit plus adapté aux applications complexes, **MySQL** répond parfaitement aux besoins du projet tout en offrant une bonne extensibilité et une sécurité de base suffisante pour le contexte.
Pour l'administration de la base de données, nous avons opté pour **phpMyAdmin**, qui permet une gestion facile et intuitive des bases de données MySQL. **phpMyAdmin** offre une interface web simple pour effectuer des tâches d'administration courantes, telles que la gestion des utilisateurs, des tables et des requêtes SQL, ce qui facilite le travail de l'équipe technique.

3. Paiement en ligne

Pour la mise en place des paiements sécurisés, deux principales solutions ont été comparées :

Critères	Stripe	PayPal
Facile d'intégration	**++** Compatibilité directe avec Symfony via son SDK.	**+** Nécessite une configuration plus complexe.
Sécurité	**++** Conformité PCI DSS et protection des données.	**++** Sécurité assurée, conforme PCI DSS.
Fonctionnalités	**++** Support des paiements récurrents, Apple Pay et Google Pay.	**+** Fonctionnalités de base pour les paiements.
Coût	**++** Frais compétitifs.	**++** Frais variables en fonction des volumes.

Choix retenu : Stripe
Stripe se distingue par sa facilité d'intégration grâce à un SDK PHP et des outils d'intégration adaptés à Symfony. Il propose une sécurité optimale conforme aux normes PCI DSS, garantissant que les informations sensibles des utilisateurs sont protégées tout au long du processus de paiement. Stripe offre aussi des fonctionnalités avancées telles que le support des paiements récurrents, Apple Pay, Google Pay, et un tableau de bord pour suivre et gérer facilement les transactions. Ces fonctionnalités permettent une grande flexibilité et une expérience utilisateur fluide. De plus, l'intégration de Stripe dans notre projet Symfony est simple, ce qui permet de gagner du temps sur la configuration et le développement.

4. Hébergement web

L'hébergement est un élément critique pour garantir la disponibilité et la performance du site. Pour la mise en place de l'hébergement, trois principales solutions ont été comparées :

Critères	IONOS	AVH	Azure
Performance	**++** Bonne performance pour les projets de taille moyenne.	**++** Performance stable et fiable pour des projets moyens.	**++** Performance solide avec une courbe d'apprentissage élevée.
Coût	**++** Offre à partir de 1 €/mois.	**+** Coût abordable pour des services de base.	**-** Coûts élevés pour des petites équipes.
Facilité d'utilisation	**++** Interface intuitive et facile à configurer.	**+** Interface simple et gestion facile des ressources	**-** Courbe d'apprentissage importante.
Sécurité	**++** Certificats SSL et protection DDoS inclus.	**++** Sécurité de base avec options supplémentaires.	**++** Sécurité renforcée.

Choix retenu : IONOS

Bien que le client nous ait manifesté sa familiarité avec **AVH**, qu'il utilise pour des noms de domaine et des adresses emails, nous avons finalement décidé de retenir **IONOS** pour plusieurs raisons pratiques et techniques.

IONOS offre un excellent rapport qualité-prix, avec des tarifs très compétitifs, surtout pour des projets de taille moyenne, ce qui correspond parfaitement à notre budget et à la structure du projet. De plus, son interface intuitive permet une gestion facile et rapide des services, contrairement à **AWS** et **Azure**, qui nécessitent une courbe d'apprentissage plus importante.

Sur le plan de la sécurité, **IONOS** fournit des fonctionnalités de base comme les certificats SSL et la protection contre les attaques DDoS, ce qui est suffisant pour assurer la sécurité de notre site tout en restant dans une gamme de prix abordable.

En outre, même si **AVH** pourrait être une option intéressante pour certains besoins spécifiques (comme les noms de domaine), la solution **IONOS** est plus adaptée à l'ensemble de nos besoins d'hébergement et garantit une meilleure stabilité à long terme, avec une gestion plus simple et un support efficace.

5. Gestion du code source

Pour la gestion collaborative du code source, nous avons utilisé **GitHub**, une plateforme de gestion de code basée sur Git, qui facilite la collaboration entre les membres de l'équipe.
Sur GitHub, nous avons suivi une approche basée sur **GitFlow**, où chaque fonctionnalité est développée sur une branche dédiée. Une fois une fonctionnalité terminée, une demande de pull (pull request) est créée pour fusionner la branche avec la branche principale. Ce processus permet de revoir le code, de s'assurer de sa qualité et de résoudre d'éventuels conflits avant la fusion. Cette méthode a

6. Résumés choix techniques

Eléments	Technologie choisie	Justification
Framework de développement	**Symfony**	Robustesse, modularité, expertise de l'équipe.
Base de données	**PostgreSQL**	Performance, compatibilité avec Symfony.
Paiement en ligne	**Stripe**	Sécurité, facilité d'intégration, modernité.
Hébergement	**IONOS**	Coût abordable, fiabilité, simplicité.
Gestion du code source	**GitHub**	Collaboration facilitée via GitFlow.

Maquette

Pour la partie Maquette, nous avons l'avons réalisé sur le logiciel Figma afin d'illustrer le visuel final de notre application. Elle offre ainsi une visualisation des fonctionnalités et de l'interface, facilitant ainsi la compréhension du design et la validation des choix effectués avant la phase de développement.

Lien maquette :
https://www.figma.com/design/lRB8myxTtQnMWSeChRsSAN/PresShop?node-id=0-1&t=IobmHpTzsbNK1vrP-1

Lien prototypage de la maquette :

PresShop

Présentation de la configuration GitHub

1. Workflow GitFlow

Le **GitFlow** est une méthodologie de gestion des branches qui structure le développement de l'application de manière claire et collaborative. Il repose sur l'utilisation de plusieurs branches pour organiser le travail.

1.1. Structure des Branches

main :

- Branche principale contenant la version stable du projet, prête pour la livraison ou la mise en production.
- Aucun développement direct n'est effectué sur **main**.

develop :

- Branche d'intégration où les nouvelles fonctionnalités sont fusionnées une fois validées.
- Elle représente la version en cours de développement du projet.

feature/ :

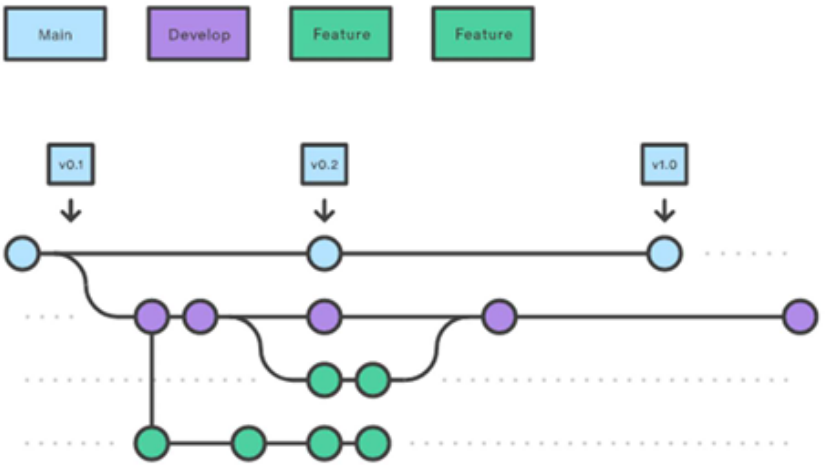
- Pour chaque nouvelle fonctionnalité, une branche spécifique est créée à partir de **develop**.
- Exemple : **feature/gestion-utilisateur** ou **feature/integration-stripe**.
- Cette organisation permet de travailler isolément sur des tâches spécifiques sans impacter les autres parties du code.

hotfix/ :

- Utilisée pour corriger des bugs critiques directement dans la branche **main**.
- Exemple : **hotfix/correction-paiement**.

2. Processus de travail

2.1. Schéma de fonctionnement de notre GitFlow



Le schéma ci-dessus illustre le processus GitFlow simplifié que nous avons adopté pour la gestion du développement de la fonctionnalité. Il montre clairement les relations entre les branches main, develop, et feature, ainsi que le rôle des Pull Requests pour garantir un code stable et validé.

Résumé des étapes :

1. Récupération de la branche main pour partir du code stable.
2. Création de la branche develop pour isoler le développement en cours.
3. Création d'une branche **feature/numFonction-NomFonction** (exemple : feature/1-Base-Projet) pour développer la fonctionnalité.
4. Fusion de la branche **feature** dans **develop** après revue et validation.
5. Fusion de la branche **develop** dans main pour intégrer la version stable du projet.

Ce processus assure une gestion structurée du développement, limite les risques d'erreurs et facilite la collaboration entre les membres de l'équipe. Grâce à l'utilisation de GitHub, le code source reste organisé, versionné et prêt à être déployé dans des délais maîtrisés.

2.2. Développement avec GitHub Desktop

2.2.1. Récupération du code source (branche main)

- **Ouvrir GitHub Desktop** et sélectionner le dépôt du projet.
- S'assurer que la branche **main** est sélectionnée dans la liste des branches.
- Cliquer sur "Fetch origin" pour récupérer les dernières mises à jour de la branche **main**.

2.2.2. Création de la branche develop

- Depuis GitHub Desktop, cliquer sur le menu **"Branch"** → **"New Branch"**.
- Nommer la nouvelle branche **develop** et s'assurer qu'elle est créée à partir de main.
- Valider en cliquant sur **"Create Branch"**.

La branche **develop** servira de base pour les fonctionnalités en cours de développement.

2.2.3. Création de la branche feature

Pour travailler spécifiquement sur la fonctionnalité **"Exemple de fonction"** :

- Avec la branche **develop** sélectionnée, aller dans "Branch" → "New Branch".
- Nommer la nouvelle branche **feature/exempleFonction**.
- Cliquer sur **"Create Branch"** pour commencer à travailler isolément sur cette fonctionnalité.

2.2.4. Développement et test dans la branche feature

- Dans la branche **feature/numFonction-exempleFonction**, nous développons et testons la fonctionnalité directement dans notre environnement local.
- À chaque étape importante, nous effectuons un **commit** :
    - Cliquer sur **"Changes"** pour voir les modifications apportées.
    - Rédiger un message de commit clair (ex : *"Ajout de la fonction"*).
    - Cliquer sur **"Commit to feature/exempleFonction"**.

Une fois satisfait du développement local, nous **publions** la branche sur GitHub en cliquant sur **"Publish Branch"**.

2.2.5. Fusion de feature dans develop (Pull Request)

Pour intégrer la fonctionnalité développée dans la branche **develop** :

- **Synchroniser** la branche **feature/numFonction-exempleFonction** sur GitHub Desktop via **"Push Origin"**.
- Aller sur **GitHub.com** et ouvrir une **Pull Request (PR)** pour fusionner **feature/numFonction-exempleFonction** dans **develop**.
- Soumettre la PR pour que les autres membres puissent effectuer une **revue de code**.
- Une fois validée, la PR est fusionnée, et les modifications sont intégrées dans la branche **develop**.

2.2.6. Fusion de develop dans main

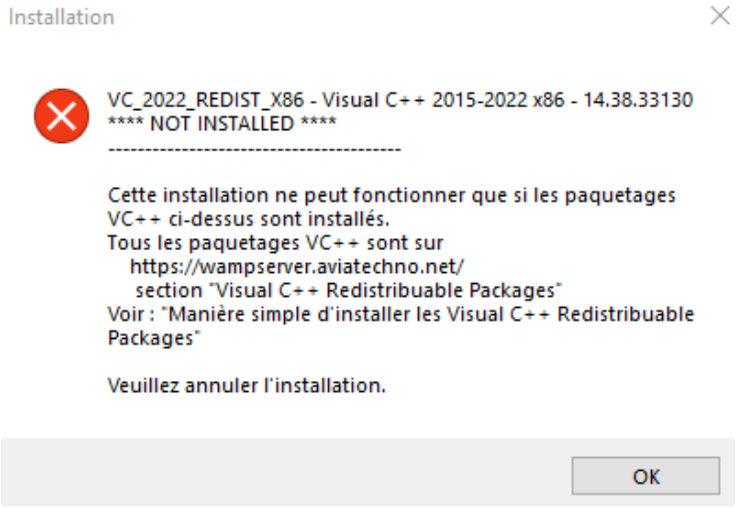
Lorsque la fonctionnalité est stable et que la branche **develop** est prête :

- Publier les dernières modifications de **develop** depuis GitHub Desktop.
- Aller sur **GitHub.com** et créer une **Pull Request** pour fusionner **develop** dans **main**.
- Après validation de la PR par l'équipe, la branche **main** intègre la nouvelle fonctionnalité, et une nouvelle version stable est prête à être déployée.

Procédure de la récupération du projet

1. Pré-requis

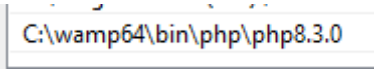
- **Git**
- **WampServer :**
  Se rendre sur : https://www.wampserver.com/
  Lancer le téléchargement.
  Dans le cas ou cette erreur surviendrait lors de l'exécution de l'installer :



  Se rendre sur cette page du support microsoft et télécharger les packages manquants :
  https://learn.microsoft.com/fr-fr/cpp/windows/latest-supported-vc-redist?view=msvc-170
  Ici, le package x86 : https://aka.ms/vs/17/release/vc_redist.x86.exe

Architecture	Lien	Notes
ARM64	https://aka.ms/vs/17/release/vc_redist.arm64.exe	Permalink pour la dernière version d'ARM64 prise en charge
X86	https://aka.ms/vs/17/release/vc_redist.x86.exe	Permalink pour la dernière version x86 prise en charge
X64	https://aka.ms/vs/17/release/vc_redist.x64.exe	Permalink pour la dernière version x64 prise en charge. Le package Redistribuable X64 contient les fichiers binaires ARM64 et X64. Ce package facilite l'installation des fichiers binaires ARM64 Visual C++ requis lorsque le redistribuable X64 est installé sur un appareil ARM64.

- **PHP:** (version recommandée par le projet, souvent >= 8.1 pour Symfony 6)
  S'assurer que le PHP référencé dans le path est bien celui de wamp et que la version de PHP est bien supérieure ou égale à 8.2 :



- **Composer** : Composer est nécessaire à l'installation et à l'utilisation de Symfony.
  https://getcomposer.org/doc/00-intro.md#installation-windows
- **Symfony:** https://symfony.com/download
  Exécuter la commande ci-dessous en ligne de commande :scoop install symfony-cli

2. Création de la BDD

- Démarrer le WampServer et se rendre sur la page suivante : http://localhost/phpmyadmin/
- Créer une nouvelle base de données se nommant “PresShop”.
- Exécuter le script de création de table SQL

3. Récupération du code

- Créer un dossier de travail lié au projet PresShop
  Cloner le repository Git
- Exécuter la commande ci-dessous afin de télécharger les dépendances : `composer install`

Prise de recul

La prise de recul sur ce milestone nous a permis de réaliser l'importance cruciale d'une analyse approfondie lors de la rédaction du cahier des charges. Grâce à cette phase et avec un peu de recul, nous avons pu identifier les points positifs à la réalisation de ce milestone :

1. Points positifs

Collaboration efficace :
Certains éléments omis lors du précédent milestone, notamment en ce qui concerne le dépôt des documents sur git, qui a été sous-estimée initialement.
L'utilisation de GitHub comme outil de collaboration et de gestion des versions témoigne d'un travail d'équipe coordonné et d'un effort pour centraliser les contributions.

Planification stratégique
Nous avons également constaté que la répartition des tâches au sein du groupe et la bonne coordination entre les membres sont essentielles pour le bon déroulement du projet. Par exemple, lors de ce milestone, nous n'étions pas souvent à l'université et les périodes de fêtes et de vacances ont perturbé nos calendriers, rendant encore plus nécessaire une gestion efficace du temps et une coordination fluide. Cela a renforcé notre conviction que le travail en équipe est un véritable atout, où chacun est prêt à aider l'autre, particulièrement dans des moments de difficulté.

Transparence et communication
Enfin, nous avons pris conscience que les échanges réguliers avec le client sont primordiaux, tant pour l'avancement du projet que pour sa qualité. Ces interactions nous permettent de nous adapter en continu à leurs besoins et d'améliorer notre travail tout au long du développement de la plateforme. Cette communication fluide et continue sera, à n'en pas douter, un facteur clé de la réussite pour les prochaines étapes du projet.

2. Difficultés rencontrées

Néanmoins nous avons dû faire face à des contraintes sur :

Complexité des choix technologiques
Les décisions liées aux technologies utilisées ont pu être difficiles à prendre, surtout si elles nécessitent une montée en compétences rapide de certains membres. Nous avons dû prendre en compte les compétences et les connaissances de chacun : Par exemple, certains outils ou frameworks initialement envisagés ont dû être écartés en raison de leur courbe d'apprentissage trop importante ou d'un manque de familiarité de l'équipe. Finalement, cela a conduit certains membres à devoir s'adapter et se former rapidement pour monter en compétences

Conclusion

En conclusion, cette phase du projet a permis de poser les bases solides pour le développement de la plateforme tout en mettant en place les outils nécessaires à leur réalisation.Elle nous a également apporté des compétences précieuses en gestion de projet, en particulier dans la planification et l'organisation des différentes étapes du processus. Nous avons appris à définir des objectifs clairs, à établir des priorités et à gérer les ressources de manière efficace. Cette phase a renforcé notre capacité à anticiper les besoins futurs et à structurer le projet de manière à respecter les délais et les exigences, tout en assurant une communication fluide entre les différentes parties prenantes. Ces compétences seront essentielles pour la réussite des prochaines étapes de développement.

La prochaine étape du projet consistera en le développement de l'application, avec une attention particulière portée à l'amélioration continue de l'interface et des performances. L'accent sera mis sur l'optimisation du système et l'intégration de fonctionnalités avancées, afin d'assurer une progression alignée avec les besoins du projet et des utilisateurs finaux. L'objectif sera de créer une application non seulement performante, mais également parfaitement adaptée aux exigences métier et à un usage quotidien.

Annexe

Guide de navigation de la maquette

Pour commencer, nous avons un formulaire de connexion, il faudra alors choisir quel rôle vous souhaitez devenir (Client, administrateur ou préparateur de commande)

Pour chaque Rôle :

Vous avez à votre disposition un “header” (menu principal pour naviguer sur le site) avec les onglets suivants :

- Catégories
- Produits
- Commandes
- Comptes
- Une icône de personnage avec un menu déroulant

Le menu déroulant contient à la fois les informations personnelles afin de vérifier les identifiants, mots de passe ainsi que le rôle associé mais aussi une action de déconnexion du compte si jamais dans le cadre de cette présentation vous souhaitez changer de rôle.

Scénario client

1. Contexte utilisateur

Un client souhaite acheter des produits frais, principalement des fruits et légumes, en ligne. Il se rend sur le site pour explorer les produits disponibles, vérifier les promotions, et éventuellement planifier une commande.

1.1. Itinéraire dans la maquette

1. **Page d’accueil :**
    - L’utilisateur arrive sur la page d’accueil, où il est accueilli par une bannière mettant en avant le support 24/7, le paiement sécurisé et le point relais.
    - Juste en dessous de la bannière, une sélection de produits phares (fruits et légumes de saison) s’affiche, avec des boutons correspondants aux différentes catégories des produits.
    - En dessous, nous avons des avis clients avec une note et une description.

2. **Navigation vers le catalogue de produits** (via bouton Produit en haut de page) **:**
    - En haut de la page, un menu avec différentes catégories est visible : **Fruits**, **Légumes**, **Produits locaux**, **Bio**.
    - En cliquant sur l’une des catégories, comme **Fruits**, l’utilisateur est dirigé vers la page listant tous les fruits disponibles.

3. **Exploration des produits** :
    - Sur la page des produits, l'utilisateur peut voir des cartes de chaque article avec une image, un prix, et un bouton ajouter au panier.
    - Une liste des catégories de produits est également disponible à gauche de la page.
    - L'utilisateur souhaite ajouter une banane au panier, il pourra alors se servir de la barre de recherche situé au-dessus des catégories et taper le mot "banane"

4. **Ajout au panier** :
    - L'utilisateur ajoute les bananes à son panier via le bouton "Ajouter au panier".
    - Une petite notification en haut à droite l'informe que le produit a bien été ajouté, avec un accès rapide à son panier pour récapituler ses articles en haut à droite via une icône de caddie.

5. **Consultation du panier** :
    - Depuis cette icône, l'utilisateur clique sur le panier pour voir un résumé de ses articles.
    - Dans le panier, il peut modifier la quantité, supprimer un article ou bien valider le panier pour paiement
    - Le client ici veut ajouter 2 bananes à son panier et valide le panier

6. **Passage en caisse** :
    - Satisfait de sa sélection, l'utilisateur passe à l'étape de paiement.
    - Il suit une page de paiement simple et claire, où il entre ses informations de livraison. (Nom, prenom, mail, date de récupération, commentaires)
    - Avant de confirmer, un récapitulatif de la commande apparaît pour vérification.

7. **Confirmation de commande** :
    - Une fois la commande validée, l'utilisateur reçoit un message de confirmation avec un numéro de suivi et une estimation de la date de livraison.
    - Un bouton lui permet d'aller consulter son historique de commandes, en cas de besoin.

Scénario administrateur

1. Contexte utilisateur

L'administrateur est chargé de gérer le contenu du site PrèsShop, de suivre les commandes, et de veiller à ce que les produits soient bien visibles et mis à jour. Il accède à une interface dédiée pour administrer les fonctionnalités du site.

1.1. Itinéraire dans la maquette

1. **Page d'accueil de l'administration** :
    - L'administrateur se connecte et arrive sur un tableau de bord où s'affichent la liste des produits disponible avec la quantité, le prix ainsi que d'autres informations, l'utilisateur a la possibilité de rechercher le produit qu'il cherche via une barre de recherche situé au-dessus
    - Notre administrateur voudra ajouter un nouveau produit qui est une Pomme Golden

2. **Ajout d'un produit :**
    - Pour cela, notre utilisateur devra appuyer sur le bouton "+" situé à gauche de la barre de recherche
    - Un pop-up s'affiche avec une liste d'informations à renseigner (Image du produit, nom, prix, Unité de vente, catégorie, quantité disponible)
    - Les données que l'utilisateur voudra entrer au sein du formulaire sont les suivantes :
        - Image : "Image d'une pomme"
        - Nom du produit : Pomme Golden
        - Prix : 2,50
        - Unité de vente : Kg
        - Catégorie : Fruits / Légumes
        - Quantité disponible : 15

3. **Ajout d'une catégorie :**
    - L'utilisateur a la possibilité également d'ajouter une catégorie de produit
    - Pour cela, il doit aller dans Catégories situé dans le menu du haut de page.
    - Ensuite , il verra un autre tableau de bord avec la liste des catégories avec pour chacune d'elle une description, la possibilité de modifier et supprimer la catégorie concernée.
    - Ici, nous souhaitons ajouter une catégorie, l'utilisateur appuie sur le bouton à gauche de la barre de recherche et comme pour les produits, un formulaire est affiché avec le nom et la description que nous voulons ajoutés.
    - Voici ce que l'utilisateur devra ajouter :
        - Nom : Céréales
        - Description : La catégorie Céréales regroupe une sélection de produits à base de féculents pour compléter une alimentation saine et équilibrée. Vous y trouverez des produits essentiels comme le riz, les pâtes, le blé, le quinoa, l'avoine et bien d'autres, tous sélectionnés pour leur qualité et leur fraîcheur. Idéale pour constituer une source d'énergie durable, cette catégorie s'adresse à tous ceux qui souhaitent intégrer des féculents variés et nutritifs dans leur quotidien
    - Après avoir créer la catégorie nous pouvons l'apercevoir au sein de la liste
    - Maintenant, nous pouvons vérifier l'intégration de nos ajouts

4. **Intégration des ajouts effectués**
    - Nous pouvons retourner dans les produits via le menu du haut de page

Normalement si vous souhaitez ajouter un nouveau produit, au sein du formulaire dans les catégories vous apercevez la nouvelle catégorie “Céréales”.

Scénario du préparateur de commande

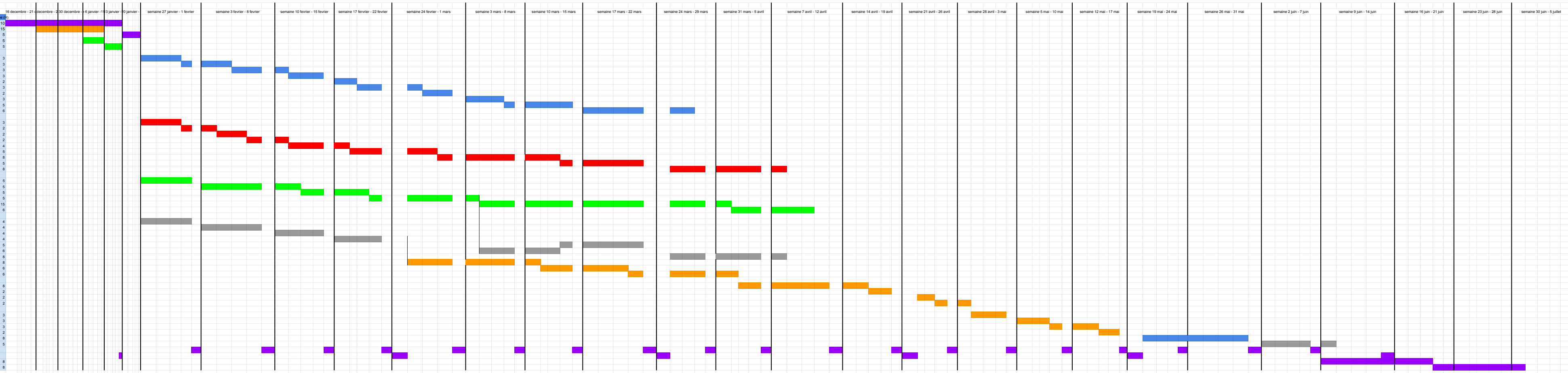
1. Contexte utilisateur

Le préparateur de commande est chargé de rassembler les produits pour les commandes passées en ligne et de préparer ces commandes pour le retrait en magasin. Ce rôle est essentiel dans un service Click & Collect, car il garantit aux clients de récupérer leurs articles rapidement et en parfait état. Le préparateur doit donc disposer d’un accès rapide et fluide aux informations sur les commandes ainsi qu’à des outils pour gérer efficacement leur traitement.

1. **Gestion des commandes**
    - En tant que préparateur des commandes, nous pouvons réaliser la gestion des commandes, pour cela il faut aller à l’onglet Commandes via le menu en haut de page.
    - Nous pouvons à présent apercevoir la liste des commandes avec plusieurs statuts et une couleur de ligne associée : en attente (Rouge), prête (Orange) et récupérée (Vert). Nous pouvons également y retrouver le numéro de commande, le nom du client associé ainsi que la date de récupération de la commande en format JJ/MM/YYYY HH:MM:SS.
    - Nous avons plusieurs actions que nous pouvons effectuer comme la visualisation ou la suppression de la commande. Il existe aussi des actions spécifiques en fonction du statut qui correspondent au changement de statut (En attente ⇒ Prête // Prête ⇒ Récupérée)
    - Pour la visualisation de la commande, nous avons un formulaire récapitulatif de cette dernière avec les informations suivantes :
        - Nom et Prénom du client
        - Son adresse mail
        - Son numéro de téléphone
        - La date de récupération de la commande
        - La liste des articles concerné ainsi que leurs quantités
        - Le montant total de la commande

Deux boutons, une pour fermer le pop-up et l’autre pour changer le statut de la commande.

Tâche	Description	Détails des fonctionnalités	Durée (H)
A'	Milestone 2		10
A"	Maquettage		15
A"'	Soutenance Milestone 2		5
A	Mise en place de l'environnement de développement		5
B	Modéliser base de données		5
C	CRUD Catégories		
C1		Ajouter une catégorie	3
C2		Modifier une catégorie	3
C3		Supprimer une catégorie	3
C4		Consulter une catégorie	3
S	Visualiser la politique de confidentialité		2
T	Mettre à jour la politique de confidentialité		3
U	Visualiser CGV		2
V	Mettre à jour CGV		3
X	Contacter le magasin		5
Z	Page d'accueil du site (+ header + footer)		6
H	CRUD Compte (avec gestion des rôles)		
H1		Ajouter un compte	3
H2		Modifier un compte	2
H3		Supprimer un compte	2
H4		Consulter un compte	2
I	Connexion		4
J	Réinitialiser le mot de passe		5
Q	Notification de limite de stock atteint		8
W	Plan d'accés au magasin		5
Y	Visualiser les commentaires Google		8
K	CRUD Commandes		
K1		Ajouter une commande (pass	5
K2		Modifier une commande (con	5
K3		Supprimer une commande (a	5
K4		Consulter une commande	5
O	Effectuer un paiement en ligne		15
N	Mettre à jour l'état d'une commande		6
D	CRUD Produits		
D1		Ajouter un produit	4
D2		Modifier un produit	4
D3		Supprimer un produit	4
D4		Consulter un produit	4
L	Suivre l'état d'une commande		5
M	Notification de validation de commande		6
P	Gestion des stocks (deduire lors d'une commande)		8
E	Rechercher un produit		8
F	Filtrer les produits		6
G	Trier les produits		6
R	CRUD Panier		
R1		Ajouter au panier	8
R2		Modifier le panier	2
R3		Supprimer du panier	2
R4		Consulter le panier	2
AA	CRUD Promos		
AA1		Ajouter une promotion	3
AA2		Modifier une promotion	3
AA3		Supprimer une promotion	3
AA4		Consulter une promotion	2
AB	Tests		8
AC	Mise en prod		5
AD	Réunions hebdomadaires entre les membres de l'équipe		
AE	Réunions mensuelles avec le client		
AF	Milestone 3		8
AF'	Soutenance Milestone 3		8



1 semaine = 5H | 1 case = 1H

Code couleur	
Lillian	rouge
louiza	orange
Damien	bleu
Lylia	vert
Bastien	gris
Toute l'équipe	violet