Rendu Milestone 3

Date de rendu : 16/02/25

Équipe de projet : Julien Durieux - Remi Mahieux - Julien Dilly - Antonin Drujon - Paco Pastor

Version	Date	Auteur	Validation	Commentaires	Information de contact	Entreprise
V1	6/02/2025	Antonin DRUJON	Julien DILLY : KO	Brouillon	antonin.drujon@etud.u-picardie.fr	**MS Corp**
V2	10/02/2025	Antonin DRUJON Paco PASTOR	Équipe : KO	- À review - MCD à améliorer	antonin.drujon@etud.u-picardie.fr paco.pastor@etud.u-picardie.fr	**MS Corp**
V3	12/02/2025	Julien DURIEUX Antonin DRUJON Rémi MAHIEUX Paco PASTOR	Equipe : KO	Review des différentes partie du documents	j.durieux@etud.u-picardie.fr antonin.drujon@etud.u-picardie.fr remi.mahieux@etud.u-picardie.fr paco.pastor@etud.u-picardie.fr	**MS Corp**
V4	12/02/2025	Paco PASTOR	Paco : KO	- Review partie 6 OK - Ajout de la partie prise de recul et organisation	paco.pastor@etud.u-picardie.fr	**MS Corp**
V5	14/02/2025	Antonin DRUJON	Antonin, Rémi : KO	- Refonte intro + partie 5 et 6 - Review annexe - Mise à jour partie 3	antonin.drujon@etud.u-picardie.fr	**MS Corp**
V6	15/02/2025	Paco PASTOR	Paco, Julien : KO	- Intro, partie 4 OK - Prise de recul terminée après review	paco.pastor@etud.u-picardie.fr	**MS Corp**
V7	16/02/2025	Antonin DRUJON	Antonin, Julien, Paco : KO	- Dernier nettoyage - Amélioration partie planification	antonin.drujon@etud.u-picardie.fr	**MS Corp**
V8	16/02/2025	Julien DURIEUX Antonin DRUJON Rémi MAHIEUX Paco PASTOR	Equipe : OK	Document OK niveau équipe	j.durieux@etud.u-picardie.fr antonin.drujon@etud.u-picardie.fr remi.mahieux@etud.u-picardie.fr paco.pastor@etud.u-picardie.fr	**MS Corp**

Table des matières

1. Introduction

Dans le cadre de notre projet, nous avons franchi une étape clé en progressant vers la phase de conception. Cette phase est essentielle pour poser des bases solides au développement de notre application, en structurant notre base de données, en définissant une planification détaillée et en mettant en place un environnement de travail collaboratif efficace via GitHub.

Ce projet s'inscrit dans un contexte réel, porté par le restaurant amiénois **"La Manufacture"**, un établissement indépendant géré par deux associés et employant une **quinzaine de personnes**. Avec un service continu de **12h00 à 22h00** et une capacité d'environ **25 à 30 tables**, l'organisation du restaurant repose sur une **prise de commandes fluide et efficace** pour assurer une expérience client optimale.

Cependant, les **méthodes traditionnelles** actuellement utilisées, notamment la prise de commandes sur papier, entraînent des **erreurs fréquentes**, des **pertes de temps** et des **malentendus** entre les serveurs et la cuisine. De plus, la gestion des demandes spécifiques (allergies, préférences alimentaires) reste complexe et source de confusion.

Face à ces défis, le restaurant souhaite **moderniser ses processus** en adoptant une solution numérique permettant de **fluidifier la communication entre les équipes**, **réduire les erreurs**, et **optimiser la gestion des stocks en temps réel**. L'objectif est de développer une **application mobile intuitive**, offrant une saisie rapide des commandes avec personnalisation, un affichage organisé en cuisine et une intégration de la gestion des stocks.

Notre approche repose sur une **méthodologie itérative et progressive**, nous permettant d'affiner notre travail tout au long du développement en anticipant les contraintes techniques et d'organisation. Actuellement, nous concentrons nos efforts sur la **modélisation des données**, la **planification des tâches** et l'**organisation du projet sur GitHub** afin d'assurer une base solide pour la suite du développement.

Ce document vise à détailler notre démarche de conception, en présentant les fondements du projet, les besoins identifiés et les solutions techniques et organisationnelles envisagées pour répondre aux attentes du client.

2. État d'Avancement du Projet

À ce stade du projet, nous avons réalisé des avancées significatives dans plusieurs domaines clés. La phase de conception de la base de données est bien avancée, avec un modèle conceptuel établi et validé. Nous avons également construit un dictionnaire des données détaillant les différents attributs et relations, garantissant ainsi une structure robuste pour la gestion des informations.

En parallèle, la planification des tâches a été effectuée en s'appuyant sur un diagramme de Gantt. Cette planification nous permet de structurer le travail en identifiant les fonctionnalités prioritaires à développer et en répartissant les responsabilités entre les membres de l'équipe. Les différentes tâches ont été intégrées dans notre espace GitHub sous forme d'issues, ce qui facilite leur suivi et leur gestion.

Sur le plan organisationnel, l'environnement de développement est désormais fonctionnel. L'organisation des branches Git a été clairement définie, et chaque fonctionnalité est développée dans une branche dédiée avant d'être intégrée à la branche principale via des Pull Requests soumises à des revues de code rigoureuses. Cela nous permet d'assurer une qualité de code optimale et d'éviter les conflits lors des fusions.

Enfin, l'équipe est bien engagée dans la dynamique du projet. Chaque membre connaît son rôle et ses responsabilités, et un suivi régulier est assuré par le responsable du pilotage pour garantir que tout se déroule conformément aux attentes et aux délais fixés.

3. Conception de la Base de Données

Nous avons élaboré un dictionnaire des données détaillé recensant l'ensemble des informations essentielles à la structuration de notre base de données. Chaque champ y est défini de manière précise avec son nom, son type de données, ses contraintes d'unicité et ses relations avec les autres entités. Cette étape nous garantit une organisation cohérente et performante des données.

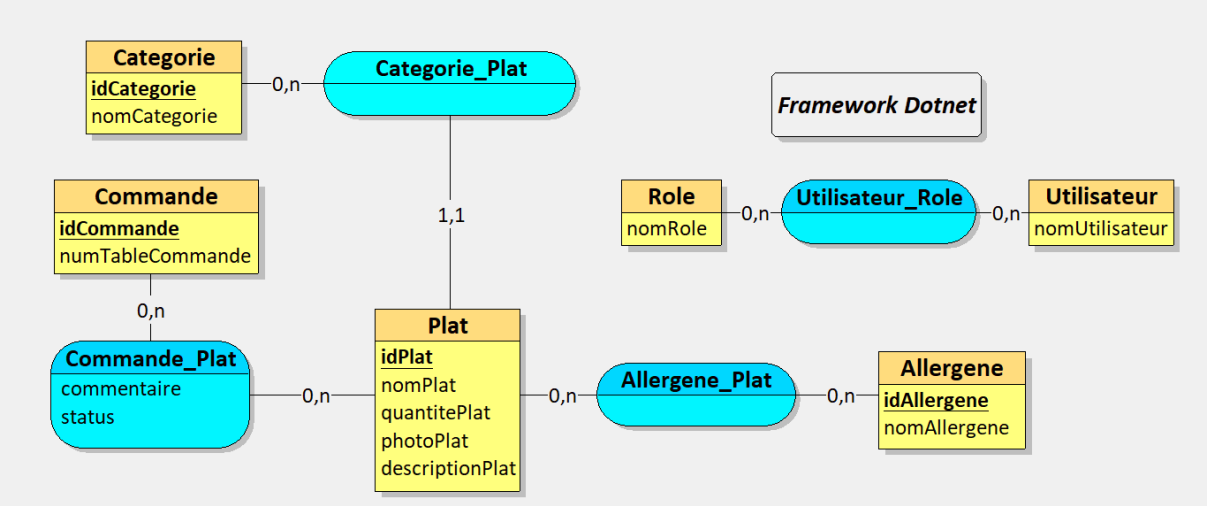
Nom de la table	Nom de l'attribut	Type de donnée	Contraintes	Description
Commande	idCommande	INT	PRIMARY KEY	Identifiant unique de la commande
	numTableCommande	INT	NOT NULL	Numéro de la table liée à la commande
Plat	idPlat	INT	PRIMARY KEY	Identifiant unique du plat
	nomPlat	VARCHAR(255)	NOT NULL	Nom du plat
	quantitePlat	INT	NOT NULL	Quantité disponible en stock
	photoPlat	TEXT		Lien ou stockage de l'image du plat
	descriptionPlat	TEXT		Description du plat
	idCategorie	INT	FOREIGN KEY	Référence la table `Categorie.idCategorie`
Categorie	idCategorie	INT	PRIMARY KEY	Identifiant unique de la catégorie
	nomCategorie	VARCHAR(255)	NOT NULL	Nom de la catégorie
Allergene	idAllergene	INT	PRIMARY KEY	Identifiant unique de l'allergène
	nomAllergene	VARCHAR(255)	NOT NUL	Nom de l'allergène
Commande_Plat	commentaire	TEXT		Commentaire ajouté à l'association entre commande et plat
	status	VARCHAR(50)	NOT NULL	Statut du plat dans la commande (ex: en cours, servi, annulé)
	idCommande	INT	FOREIGN KEY	Référence la table `Commande.idCommande`
	idPlat	INT	FOREIGN KEY	Référence la table `Plat.idPlat`
Allergene_Plat	idPlat	INT	FOREIGN KEY	Référence la table `Plat.idPlat`
	idAllergene	INT	FOREIGN KEY	Référence la table `Allergene.idAllergene`

Le Modèle Conceptuel de Données (MCD) ci-dessous illustre les relations entre les différentes entités de notre base de données. Il met en évidence les cardinalités ainsi que les associations entre les tables principales :

📌 **Explication des Relations :**

- **Commande ↔ Commande_Plat ↔ Plat** : Une commande peut contenir plusieurs plats, et un plat peut être associé à plusieurs commandes. L'entité **Commande_Plat** permet de gérer cette relation en ajoutant des informations comme le statut et les commentaires.
- **Plat ↔ Categorie** : Chaque plat appartient à une seule catégorie, mais une catégorie peut regrouper plusieurs plats.
- **Plat ↔ Allergene_Plat ↔ Allergene** : Un plat peut contenir plusieurs allergènes, et un allergène peut être présent dans plusieurs plats. L'entité **Allergene_Plat** gère cette relation.

L'outil **Looping** a été utilisé pour concevoir ce modèle, assurant une représentation visuelle claire des entités et de leurs interactions.



Modèle Conceptuel de Données (MCD) – Structure et Relations de la Base de Données

4. Planification des Tâches

La planification est une étape essentielle pour garantir une gestion efficace du projet. Nous avons utilisé un diagramme de Gantt pour organiser et visualiser notre avancement, ce qui nous a permis de définir clairement les délais et les étapes du projet. À partir du cahier des charges, nous avons choisi de sélectionner toutes les fonctionnalités décrites dans l'analyse fonctionnelle, car nous pensons pouvoir tout réaliser en temps et en heure. Chaque fonctionnalité a ensuite été minutieusement détaillée en tâches spécifiques, facilitant ainsi leur exécution.

Fonctionnalités à développer :

- **Gestion des commandes**
  - Créer une commande
  - Éditer une commande
  - Envoyer une commande en cuisine
  - Envoyer les plats d'une commande en salle
  - Consulter la liste des commandes
  - Récupérer une commande
  - Clôturer une commande
- **Gestion des plats**
  - Ajouter un plat
  - Éditer un plat
  - Supprimer un plat
  - Consulter la liste des plats
  - Éditer le stock de plats
- **Sécurité**
  - Saisir un code PIN
  - Éditer le code PIN
- **Communication**
  - Appeler un serveur

Afin de répartir les tâches de manière optimale, nous avons attribué à chaque tâche une estimation de difficulté en utilisant la méthode de **difficulté Fibonacci**, un système d'évaluation qui nous a permis d'identifier la charge de travail pour chaque tâche. Ce système repose sur la suite de Fibonacci (1, 2, 3, 5, 8, 13, 21...), où chaque nombre représente un niveau de complexité croissant. Pour attribuer ces niveaux de difficulté, nous avons discuté en équipe de la charge de travail associée à chaque tâche, en prenant en compte la complexité technique, le temps nécessaire et les compétences spécifiques requises.

Exemple de répartition des tâches et discussion sur la difficulté :

- **Tâche : Implémentation du menu latéral (difficulté : 2 - Facile 🟢)**

  La mise en place du menu latéral a été estimée comme **facile**, car elle repose principalement sur l'utilisation de composants standards de Vue.js et/ou de bibliothèques CSS existantes (comme Tailwind ou Bootstrap). Cependant, une attention particulière doit être portée à l'ergonomie et à la navigation pour assurer une intégration fluide et intuitive dans l'application.

- **Tâche : Mise en place du projet Vue.js (difficulté : 3 - Modérée 🟡)**

  La mise en place du projet a été également estimée à 3, en raison de la configuration des dépendances nécessaires et de l'organisation du projet pour garantir qu'il soit opérationnel.

- **Tâche : Implémentation de la barre de recherche (difficulté : 5 - Difficile 🔴)**

  L'ajout d'une **barre de recherche** a été évalué comme **difficile**, car elle implique plusieurs aspects techniques avancés. Il est nécessaire d'implémenter une **recherche efficace et réactive**, potentiellement avec une **mise à jour en temps réel** des résultats via la gestion de l'état dans Vue.js.

Barème de temps par difficulté :

Afin d'estimer la charge de travail et d'optimiser la répartition des tâches, nous avons défini un barème basé sur le nombre de jours nécessaires par développeur en fonction du niveau de difficulté. Ce barème nous permet d'avoir une vision claire du temps alloué à chaque tâche et d'assurer une gestion efficace des ressources humaines. Voici notre classification :

- **Très Facile** : **1 jour / homme** → Tâches élémentaires demandant peu d'effort et de réflexion.
- **Facile** : **2 jours / homme** → Tâches simples nécessitant une implémentation de base sans complexité majeure.
- **Modérée** : **3 jours / homme** → Tâches impliquant une réflexion plus approfondie, avec une légère complexité technique.
- **Moyennement Difficile** : **5 jours / homme** → Tâches demandant une bonne compréhension du projet et une maîtrise des outils utilisés.
- **Difficile** : **7 jours / homme** → Tâches complexes nécessitant une expertise avancée et une forte implication.

Par la suite, nous avons attribué les tâches en fonction des compétences et des tâches critiques du projet.

Vous pouvez voir ci-dessous le diagramme de Gantt pour notre planification avec :

- **Antonin Drujon** en vert 🟢
- **Julien Durieux** en bleu 🔵
- **Julien Dilly** en jaune 🟡
- **Paco Pastor** en violet 🟣
- **Rémi Mahieux** en rouge 🔴

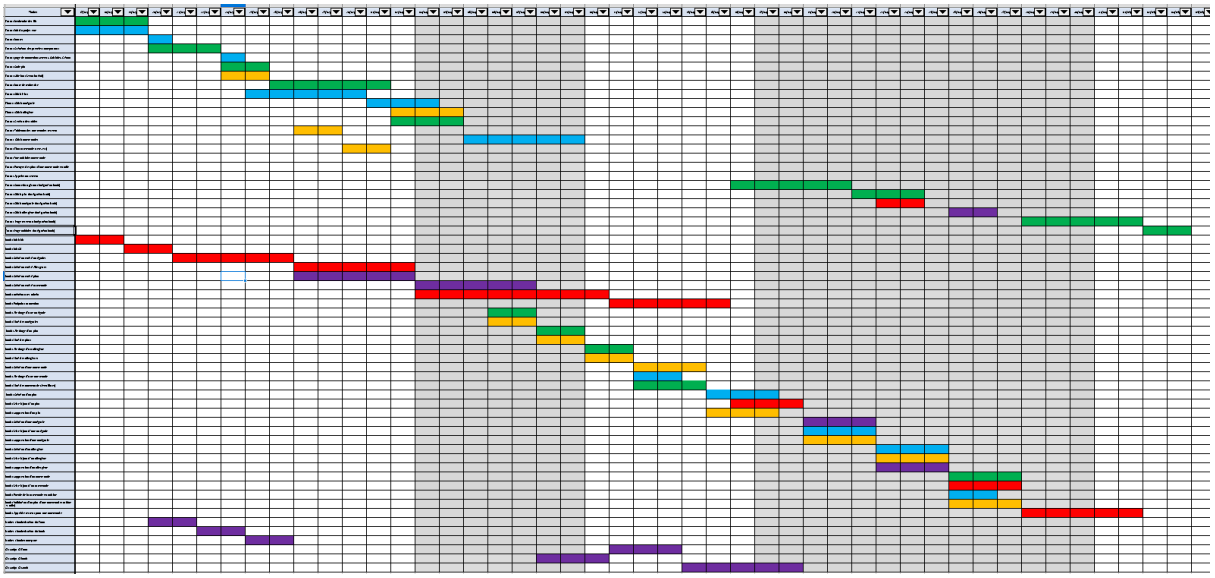
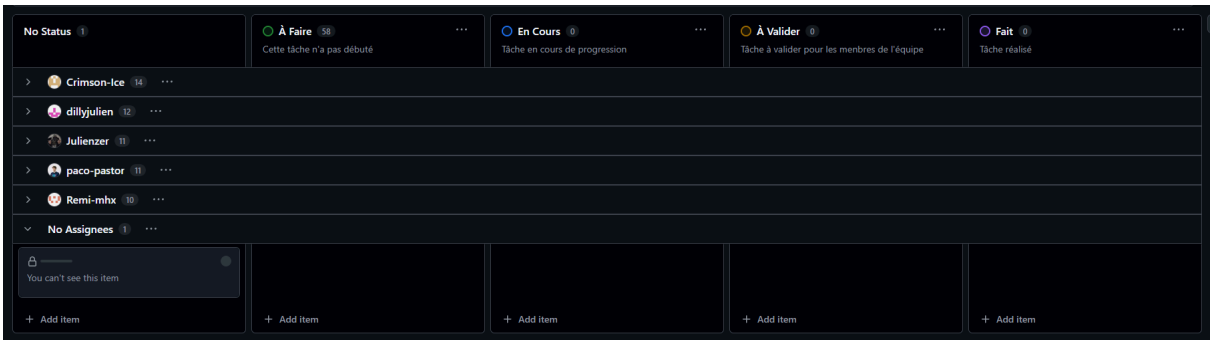


Diagramme de GANTT pour Food Master Solution

https://github.com/miage-amiens-organization/2024_M1_PRO-05_GR5/blob/main/Documents/Milestone_3/Gantt_FMS.xlsx

Pour nous organiser, nous avons choisi de développer en priorité la logique du front sans les routes API. Le développement de l'interface utilisateur étant relativement rapide, les développeurs front disponibles commenceront les tâches backend afin d'accélérer l'avancement global. Antonin Drujon sera principalement chargé d'intégrer les routes backend au frontend.

Les tâches ont ensuite été intégrées sous forme d'issues dans notre repository GitHub, facilitant leur suivi et leur attribution. Ce système de gestion des tâches permet une transparence totale et un contrôle efficace de l'avancement du projet. Enfin, la planification sera ajustée régulièrement en fonction de l'évolution réelle du développement, permettant ainsi d'adapter le travail aux éventuelles difficultés rencontrées en cours de route.



Github Issues du projet sur le tableau de bord (Kanban)

Statut des tâches :

Afin d'assurer un suivi rigoureux de l'avancement du projet, nous avons défini quatre statuts principaux pour nos tâches. Ces statuts nous permettent d'identifier clairement l'état d'une tâche et d'assurer une meilleure coordination entre les membres de l'équipe :

- **À faire** : Tâche non commencée, en attente de traitement.
- **En cours** : Tâche actuellement en cours de développement.
- **À valider** : Tâche terminée par le développeur, en attente de validation par un membre de l'équipe.
- **Fait** : Tâche finalisée et validée, considérée comme complétée.

Ce système de suivi est intégré à notre gestion de projet sur GitHub, permettant un suivi en temps réel et une meilleure réactivité en cas de blocage ou d'imprévu

Pour améliorer encore notre organisation, nous avons ajouté des **tags** spécifiques aux tâches, permettant de mieux les catégoriser :

- **Frontend** : Développement de l'interface utilisateur.
- **Backend** : Développement des fonctionnalités serveur et API.
- **Infra** : Configuration et gestion des infrastructures.
- **Fix** : Correction de bugs et améliorations.
- **Feature** : Ajout de nouvelles fonctionnalités.

Chaque tâche possède également une **date de début et de fin** afin de garantir un suivi précis et un respect des délais prévus.

5. Mise en Place de l'Environnement de Développement

Afin de garantir un développement structuré et efficace, nous avons mis en place un environnement de travail organisé et collaboratif. Cette mise en place comprend l'utilisation d'outils adaptés, une gestion rigoureuse du code source via **GitHub**, ainsi qu'une organisation claire de l'arborescence de fichier et des conventions de développement.

Pour garantir la cohérence et la lisibilité du code, nous avons défini des **conventions de nommage** adaptées au **frontend** et au **back-end** :

- **Frontend (Vue.js)** : Utilisation du **camel case** (`nomDeComposant`, `nomDeVariable`).
- **Backend (C#, DotNet)** : Utilisation du **camel case** (`NomDeClasse`, `nomDeVariable`).

Ces conventions assurent **une uniformité du code**, facilitent la collaboration et minimisent les risques d'erreurs.

Pour assurer une organisation claire et efficace du front-end, nous avons adopté une architecture basée sur l'Atomic Design, combinée à une découpe en fonctionnalités distinctes.

L'Atomic Design repose sur une hiérarchie de composants allant du plus simple au plus complexe. À la base, on retrouve les **atomes**, qui représentent les éléments fondamentaux de l'interface, comme les boutons, les champs de saisie ou les icônes. Ces atomes sont ensuite combinés pour former des **molécules**, qui regroupent plusieurs atomes en un seul composant fonctionnel, comme un champ de recherche avec son bouton de validation. À un niveau supérieur, les **organismes** assemblent plusieurs molécules et atomes pour former des sections plus complètes de l'interface, comme un menu latéral ou une barre de navigation. Enfin, les **templates** définissent la structure générale des pages sans contenir de données réelles, tandis que les **pages** instancient ces templates avec du contenu dynamique et des données spécifiques. Cette approche modulaire facilite la maintenance et la réutilisation des composants tout au long du projet.

En parallèle, la structuration du projet repose sur une découpe en **features**, où chaque fonctionnalité est isolée dans un dossier dédié contenant tous les éléments nécessaires à son bon fonctionnement. Une fonctionnalité regroupe ainsi ses propres composants, styles, services et éventuelles ressources, garantissant une organisation claire et limitant les dépendances entre les différentes parties du code. Cette approche permet une meilleure évolutivité et simplifie le travail en équipe, car chaque développeur peut intervenir sur une fonctionnalité spécifique sans impacter le reste de l'application.

Pour le développement du back-end, nous allons utiliser la **Clean Architecture**. Ce modèle nous permet d'assurer une séparation claire des responsabilités, facilitant ainsi la **maintenabilité, l'évolutivité et la testabilité** du projet.

L'organisation du code repose sur plusieurs couches distinctes, chacune ayant un rôle bien défini :

- **La couche Domaine (Domain)** : Elle contient l'ensemble des entités métier et des interfaces définissant les règles de l'application. Cette couche est totalement indépendante des détails techniques et ne dépend d'aucune autre couche.
- **La couche Application** : Elle regroupe les cas d'utilisation et la logique applicative, en orchestrant les interactions entre les différentes entités métier. Cette couche interagit uniquement avec le domaine via des interfaces.
- **La couche Infrastructure** : Elle implémente les services externes comme les bases de données, les API tierces ou encore la gestion des fichiers. Cette couche est responsable des interactions avec les technologies spécifiques et

est injectée dans l'application via l'inversion de dépendances.

- **La couche Présentation (API)** : C'est le point d'entrée du back-end, exposant les fonctionnalités via des **endpoints**. Elle gère les requêtes des clients et transmet les réponses appropriées en respectant les formats standardisés.

Le repository est organisé en deux parties distinctes pour séparer les responsabilités entre le **frontend** et le **back-end** :

Cette structure permet une **séparation claire des responsabilités**, simplifie la navigation dans le code et assure une meilleure évolutivité du projet.

6. Stratégie d'Utilisation de Git/GitHub

Nous avons adopté une organisation rigoureuse pour garantir une gestion efficace du projet à travers Git et GitHub. Chaque développement, qu'il concerne le frontend, le backend ou une autre partie du projet, suit une méthodologie claire de gestion des branches et de revue de code.

La branche principale, `main`, est strictement réservée aux versions stables et validées du projet. Afin de structurer le développement, chaque nouvelle fonctionnalité est développée sur une branche spécifique, nommée selon le schéma `feature/nom_fonctionnalité`, ce qui permet un travail modulaire et facilite l'intégration progressive des différentes avancées. Lorsqu'un bug nécessite une correction urgente, une branche dédiée au correctif, suivant le modèle `fix/nom_fix`, est créée afin d'apporter rapidement une solution sans perturber l'ensemble du développement.

Aucune modification ne peut être directement intégrée à la branche principale sans passer par une **Pull Request (PR)**. Ce processus garantit une vérification systématique du code avant son ajout définitif. Chaque PR est soumise à une revue rigoureuse effectuée par l'équipe concernée. Les développeurs du frontend, Antonin Drujon, Julien Durieux et Julien Dilly, se chargent de l'examen des contributions liées à l'interface utilisateur et aux interactions utilisateur. De leur côté, Rémi Mahieux et Paco Pastor, en charge du backend, s'assurent de la robustesse et de l'optimisation des fonctionnalités serveur. En cas de désaccords techniques ou de choix à arbitrer, Antonin Drujon pour le frontend et Rémi Mahieux pour le backend ont la responsabilité de trancher et de garantir la cohérence du projet.

Pour renforcer la fiabilité du développement, un système d'intégration continue est mis en place grâce à GitHub Actions. Géré par Paco Pastor, cet outil automatise l'exécution des tests unitaires et d'intégration à chaque nouvelle contribution, permettant ainsi d'identifier rapidement d'éventuelles erreurs. Il inclut également une analyse statique du code afin de détecter des problèmes potentiels et d'améliorer la qualité globale du projet.

L'organisation et le suivi du projet reposent sur une gestion efficace des tâches et des évolutions via GitHub Projects et GitHub Issues. Julien Durieux, chargé du pilotage global, veille à la bonne répartition des tâches et au respect des délais. La planification s'articule autour d'un tableau Kanban détaillant les différentes étapes du projet, avec une catégorisation des tâches entre celles à réaliser, celles en cours, celles en attente de validation et celles finalisées. Cette approche permet de maintenir une vision claire de l'état d'avancement du projet et de réagir rapidement aux ajustements nécessaires.

Grâce à cette organisation méthodique et aux outils mis en place, nous assurons une gestion fluide du développement, une traçabilité efficace des modifications et une qualité optimale du code, tout en favorisant un travail d'équipe collaboratif et structuré.

Annexe : Organisation du projet et prise de recul

Répartition des tâches

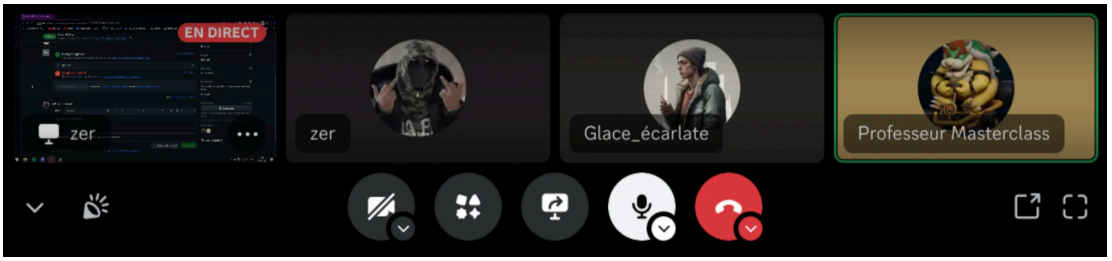
La répartition des tâches s'est faite de manière naturelle, connaissant de mieux en mieux les forces et faiblesses de chacun. Voici une explication des différents rôles au sein de l'équipe :

Julien DURIEUX	🎯 Pilotage du projet 🪵 Planification 🔪 Review globale du projet	Notamment grâce à ses compétences de chef de projet en alternance, Julien est un élément clé pour le pilotage de ce projet.
Paco PASTOR	✏️ Fautes de syntaxe ⚙️ Reviews parties	Paco possède une préférence sur le développement et a donc été amené à travailler sur les parties techniques.

	techniques 🧠 Prise de recul	
Julien DILLY	🔪 Rédaction 📊 Reviews parties organisation	Julien a travaillé sur la rédaction du projet et s'est principalement investi dans les parties concernant l'organisation globale du projet.
Rémi MAHIEUX	⚙️ Reviews parties techniques 📄 Mise en page	De par son expérience en Back-End, Rémi fut tout choisi pour travailler sur la séparation des tâches et les parties techniques.
Antonin DRUJON	🔪 Rédaction 📔 Planification	Antonin a participé à la planification avec Julien mais a aussi rédigé le squelette global du document.

Réunions et communication

Nous avons organisé des réunions hebdomadaires pour suivre l'avancement du projet. Selon la période, ces réunions se tenaient soit en présentiel à l'université, soit à distance via des appels Discord. Ces rencontres régulières étaient l'occasion de discuter des progrès réalisés, de résoudre les problèmes rencontrés, et de redéfinir les priorités.



Configuration GitHub lors d'une réunion Discord

Prise de recul

Au cours de cette nouvelle phase du projet, nous avons su mieux gérer notre temps et travailler de manière plus régulière. Contrairement au Milestone 2, où notre planification initiale était insuffisante et où nous travaillions par périodes de forte intensité suivies de relâchements, nous avons cette fois adopté une approche plus équilibrée. Nous avons réparti nos tâches sur toute la durée du Milestone, ce qui nous a permis de réduire la pression et d'assurer une progression plus fluide et maîtrisée.

Nous avons été confrontés à une difficulté imprévue : notre modèle conceptuel de données (MCD) s'est avéré trop limité par rapport aux exigences du rendu, ce qui a perturbé notre organisation et nous a fait perdre du temps. Nos tâches étaient également trop séparées par manque de temps dû aux nombreux rendus à faire durant cette période, et nous n'avons pas pu couvrir chacun l'entièreté du Milestone, ce qui signifie que des sujets ont été mieux compris par certains que par d'autres.

Grâce à ces améliorations et aux leçons tirées de nos difficultés, nous avons progressé en termes d'organisation et de structuration de notre travail. Nous avons pris conscience de l'importance d'une planification rigoureuse et d'une réflexion approfondie dès les premières étapes afin d'éviter des corrections lourdes en fin de parcours. Cette démarche nous servira pour le développement de l'application.