Projet de bases de données

420-4GW-BB

**Proposition de travail**

Travail présenté à

El Kharraz, Amal

Par

HO Louis,

RENAUD Gabriel,

AVLAH Love Josué,

HAMDOUNI Youssef

Collège de Bois-de-Boulogne

05 février 2023

Table des matières

[Description du projet 2](#_Toc135000892)

[Description 2](#_Toc135000893)

[Motivations 2](#_Toc135000894)

[Détail des besoins 2](#_Toc135000895)

[Portée 3](#_Toc135000896)

[Exigences 4](#_Toc135000897)

[Exigences fonctionnelles 4](#_Toc135000898)

[Exigences non fonctionnelles 4](#_Toc135000899)

[Modèle de la base de données 5](#_Toc135000900)

[Clé de lecture 5](#_Toc135000901)

[Exemples d’interfaces 6](#_Toc135000902)

[Cas d’utilisation 11](#_Toc135000903)

[Diagramme des cas d’utilisation 11](#_Toc135000904)

[LA MÉTHODOLOGIE DE TRAVAIL EN ÉQUIPE 12](#_Toc135000905)

[OUTILS 12](#_Toc135000906)

[Frontend 12](#_Toc135000907)

[Backend 12](#_Toc135000908)

[Travail en équipe 12](#_Toc135000909)

[RESPONSABILITÉS 12](#_Toc135000910)

[MODE DE TRAVAIL 14](#_Toc135000911)

[Rencontre hebdomadaire 14](#_Toc135000912)

[Priorisation des tâches 14](#_Toc135000913)

[GESTION DES SOURCES 14](#_Toc135000914)

[LES PRINCIPES SOLID, DRY ET KISS 15](#_Toc135000915)

[SOLID (4/5) 15](#_Toc135000916)

[DRY 15](#_Toc135000917)

[KISS 16](#_Toc135000918)

[MÉDIAGRAPHIE 17](#_Toc135000919)

# Description du projet

## Description

Justin, un entrepreneur dans le secteur de la restauration œuvre en partenariat avec des restaurants. Il souhaite à présent créer une plateforme de vente en ligne afin de leur permettre d’effectuer des ventes à des clients qui auront la possibilité de commander et se faire livrer de la nourriture. L’entrepreneur pourra ainsi percevoir des commissions sur les ventes réalisées par ses partenaires.

### Motivations

Les raisons qui ont motivé l’acceptation de la prise en charge de ce projet par notre équipe de développement sont multiples :

* D’une part, l’équipe n’avait jamais eu à développer une application web pour de la restauration, donc elle voyait en ce projet une occasion de vivre de nouvelles expériences en développement.
* D’autre part, l’ensemble de l’équipe est motivé par le thème de la restauration qui chapeaute le projet.

## Détail des besoins

Pour récapituler, le client a donc les besoins suivants :

1. Une plateforme de vente en ligne.
2. Cette plateforme doit exposer les menus proposés par chaque restaurant.
3. Un client doit pouvoir commander de la nourriture.
4. Un client doit pouvoir recevoir sa commande par livraison.

## Portée

Après analyse des besoins du client, le développement d’une application web transactionnelle est la solution retenue pour la nature du projet considéré.

Dans une première version de l’application, voici ce que le système pourra faire :

* **Créer un compte client.** *Un client qui détient déjà un compte, pourra directement rejoindre une page de connexion et s’y connecter, sinon, il aura la possibilité de s’inscrire et de procéder ensuite à la connexion.*
* **Créer un compte administrateur (admin).** *Un admin peut ajouter de nouveaux restaurants partenaires, de nouveaux menus dans les restaurants ainsi que des items dans les menus.*
* **Afficher les restaurants partenaires.**
* **Afficher une carte de localisation des restaurants aux clients.**
* **Afficher les menus de chaque restaurant.** *Le client pourra parcourir les menus, ainsi que les items de chaque menu. Il pourra vérifier leur disponibilité.*
* **Créer un panier d’items.** *Le client peut choisir un item d’un menu et l’ajouter au panier.*
* **Afficher une page de paiement avec le panier, l’adresse de livraison ainsi que la facture.**
* **Montrer les détails de la facture d’une commande.**
* **Confirmer la réception des commandes aux clients.**
* **Procéder aux transactions bancaires avec PayPal.**
* **Envoie de facture par courriel au client.**

Voici les fonctionnalités exclues du système qui seront à développer éventuellement dans une prochaine version :

* **Montrer les états de traitement d’une commande.** (Ex : réception de commande, traitement, fin de traitement, récupération de la commande par le livreur, commande en transit, commande livrée).
* **Affichage des états de la commande d’un client.** *Une version ultérieure de l’application devra montrer au client, les différentes étapes franchies dans le traitement de la commande (réception de commande, traitement de celle-ci, fin du traitement, commande récupérée par le livreur, commande en transit et commande livrée).*

# Exigences

## Exigences fonctionnelles

\* Voir la section ***portée*** plus haut.

## Exigences non fonctionnelles

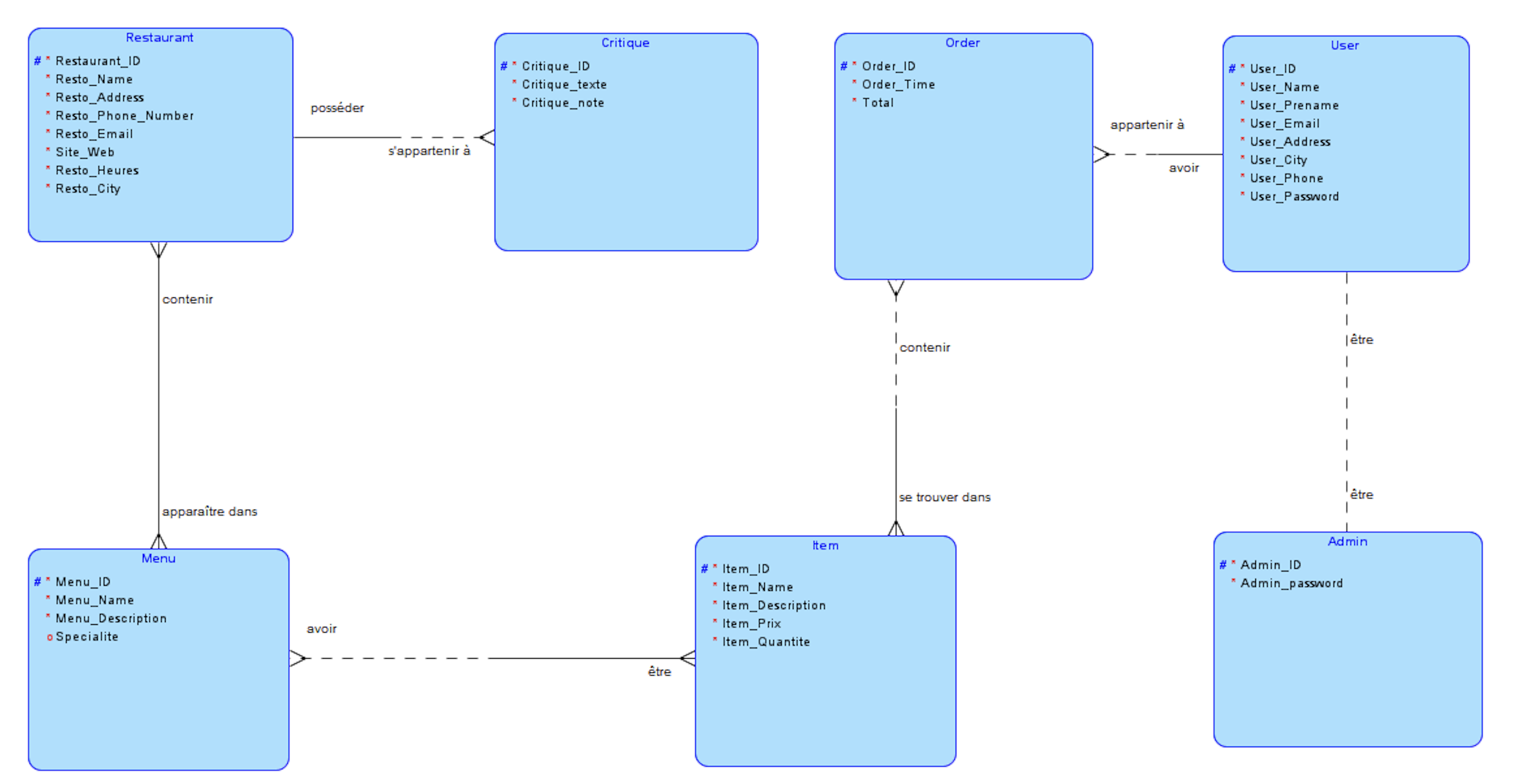
\* Voir la sous-section ***outils*** dans la section ***Méthodologie de travail en équipe*** *plus bas*.

# Modèle de la base de données

Voici un modèle de la base de données, réalisé à l’aide de la notation standard d’Oracle.

Ce modèle tente de couvrir les concepts essentiels du projet BDEBUBER-Eats, d’en couvrir les grandes lignes. Ces différents concepts sont, pour nous, également la façon dont nous stockerons les données du programme, à l’aide d’une base de données.

Normalement, tout concept important pour le déroulement du projet devrait se retrouver ici, ce qui vous permet d’avoir une vision d’ensemble en un coup d’œil rapide sur notre compréhension des différents aspects nécessaires au projet.



Dans la deuxième partie du projet, nous sommes passés sur une base de données non relationnelle (NoSQL) à savoir MongoDB.

## Clé de lecture

Un restaurant peut posséder une ou plusieurs critiques.

Une ou plusieurs critiques doivent appartenir à un restaurant.

Un ou plusieurs restaurants doivent contenir un ou plusieurs menus.

Un ou plusieurs menus doivent apparaître dans un ou plusieurs restaurants.

Un ou plusieurs menus doivent avoir un ou plusieurs items.

Un ou plusieurs items peuvent être dans un ou plusieurs menus.

Un ou plusieurs « orders » doivent contenir un ou plusieurs items.

Un ou plusieurs items peuvent se trouver dans un ou plusieurs « orders ».

Un ou plusieurs « orders » doivent appartenir à un ou plusieurs « users ».

Un « user » peut avoir un ou plusieurs « orders ».

Un « user » peut être un « admin ».

Un admin peut être un « user ».

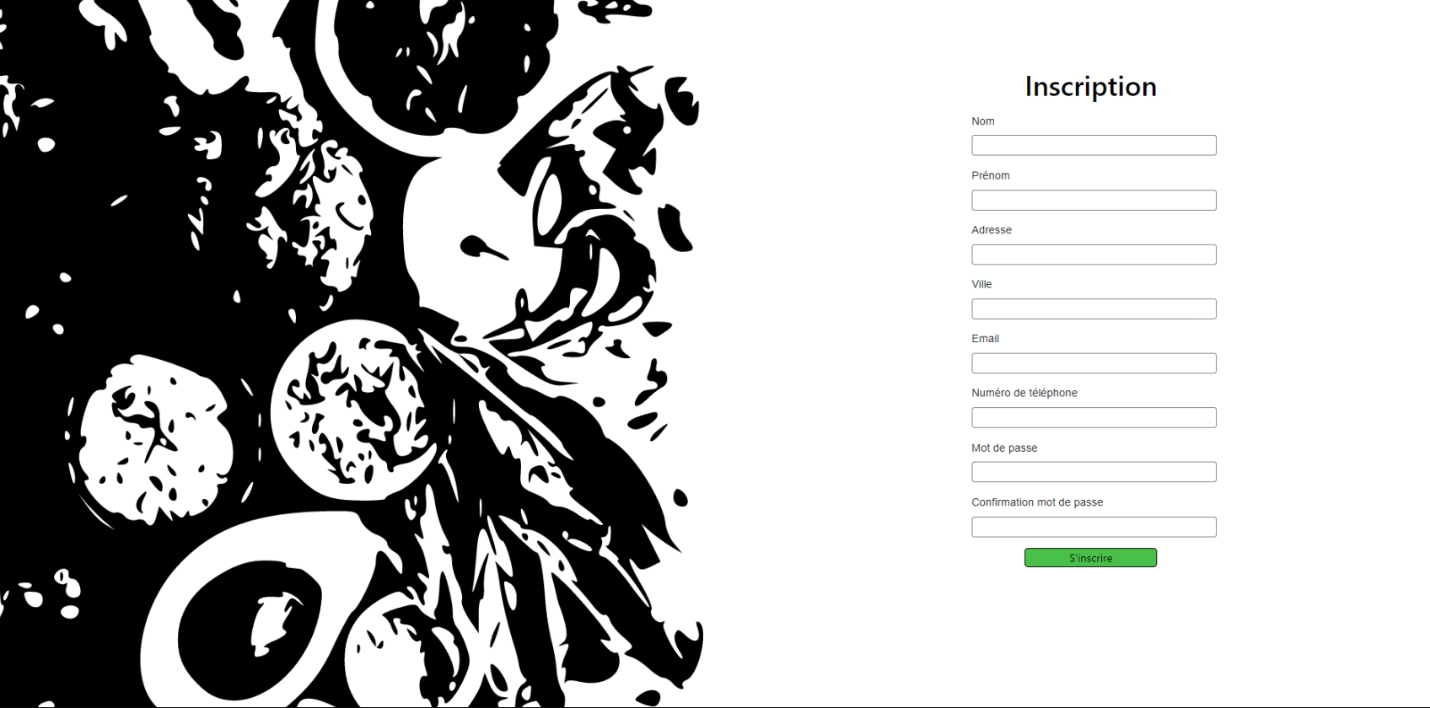
# Exemples d’interfaces

Voici quelques exemples de ce à quoi pourrait ressembler l’interface d’utilisation du nouveau module.

*À noter que ces images ne sont que des exemples et ne reflètent aucunement la version finale du produit.*

Page d’accueil : Graphical user interface, text, application

Description automatically generated

Page d’inscription : 

Page de connexion : Graphical user interface, application

Description automatically generated

Page choisir Restaurant : Map

Description automatically generated

Page Choisir Item : Company name

Description automatically generated with medium confidence

Page Paiement : Graphical user interface, text, application, email

Description automatically generated

Page de Remerciement/Attente : A screenshot of a computer

Description automatically generated with medium confidence

# Cas d’utilisation

## Diagramme des cas d’utilisation

Diagram

Description automatically generated

Le client commencera par retrouver une page qui lui dirige soit vers la page de création de compte ou soit vers une page login. Si un utilisateur n’a pas de compte existant il devra en créer un. Ensuite, il sera dirigé vers une page de recherche qui indique diverses options de restaurants. Si aucun restaurant n’est disponible soit par manque d’inventaire ou si un restaurant est fermé, l’utilisateur aura alors une page qui lui indique d’essayer à une autre date. Si le restaurant devient disponible, l’utilisateur accédera au menu de ce dernier et les items choisis apparaitront dans le panier de commande. Si l’utilisateur souhaite continuer la commande, il retournera vers la page de recherche, sinon il passera à la page transaction. Si la transaction est effectuée avec succès, il passera à une page de confirmation et recevra ensuite un courriel de reçu.

# LA MÉTHODOLOGIE DE TRAVAIL EN ÉQUIPE

## OUTILS

### Frontend

Le design des pages doivent être responsable.

Le design des pages de notre site est réalisé en langages HTML et CSS. Le code HTML est contenu dans un fichier avec l’extension : « .ejs » Nous avons utilisé la librairie

« Bootstrap » pour la création de la page d’accueil et de la page de connexion.

### Backend

Javascript, NodeJs ont été utilisés pour la création du serveur. Express est le module de choix pour les processus de routage.

MySQL est la base de données utilisée pour assurer le stockage des données des utilisateurs de l’application. Toutes les requêtes sur la base de données sont effectuées en langage SQL.

### Travail en équipe

Pour assurer la gestion des versions de notre application et faciliter le travail en équipe nous avons utilité Git, un incontournable qui répond à ces besoins.

Visual Studio Code est l’environnement de développement que l’équipe a opté pour, en raison de sa facilité d’utilisation ainsi que son interface conviviale.

## RESPONSABILITÉS

De prime abord, chaque membre de l’équipe est responsable de développer une page du site qui lui est assignée. Voici un bref aperçu des responsabilités de chaque équipier :

LOUIS

* Page d’accueil.
* Ajout des restaurants.
* Paiement.

JOSUÉ

* Connexion.
* Ajout des items.
* Remerciement.

YOUSSEF

* Carte de géolocalisation des restaurants.
* Affichage des menus des restaurants.

GABRIEL

* Inscription.
* Accueil d’admin.
* Ajout de menu.

Page d’accueil : cette page contient un bouton vers la page connexion.

Page de connexion : permet aux clients de se connecter.

Page d’inscription : permet aux clients de s’inscrire.

Accueil d’admin : Cette page possède trois bouton « ajout restaurant », « ajout menu » et « ajout item » pour procéder aux différents ajouts sur l’application.

« Ajout des restaurants », « ajout de menu » et « ajout item » : ces trois pages ont le même principe de fonctionnement. L’administrateur dispose de formulaires pour ajouter des données de restaurants, de menus et d’items dans la base de données.

Carte de géolocalisation des restaurants : cette dispose des punaises pour localiser chaque restaurant. Un lien vers le menu des restaurants est disponible aux cliques sur les punaises.

Affichage des menus des restaurants : cette page affiche les menus d’un restaurant et permet de cliquer sur les items pour les commander selon la quantité voulue.

Paiement : cette page permet au client d’entrer son adresse de livraison et de procéder au paiement PayPal.

Remerciement : cette page permet aux clients de recevoir une confirmation de leur achat ainsi qu’un bouton leur permettant de recevoir une facture par courriel.

## MODE DE TRAVAIL

### Rencontre hebdomadaire

Le travail se fait bien souvent chacun de son côté lors des heures de cours, mais essentiellement en dehors de ces dernières. Le travail de chacun est ensuite fusionné à échéance pour obtenir le résultat final désiré. À chaque rencontre de la semaine en cours, l’équipe fait le point sur les tâches réalisées par chaque équipier, le nombre d’heure de travail livré par chacun, l’évaluation des efforts fournis ainsi que la planification des prochaines tâches à effectuer.

### Priorisation des tâches

Aucune priorité n’a été établie entre les tâches du projet.

## GESTION DES SOURCES

Comme il a été mentionné plus tôt, Git est utilisé pour gérer les versions du projet. Chaque équipier travaille de son côté, sur sa branche, et fait des ajouts progressifs de nouvelles fonctionnalités ou des parties de tâches terminées dans sa branche. Par exemple, pour les pages du site, Louis a travaillé sur sa branche pour page d’accueil, Josué quant à lui s’est chargé de la page de connexion, Gabriel a travaillé sur la page d’inscription et Youssef, la page concernant les menus associés aux restaurants. Au terme du travail de chacun, toutes les branches ont été fusionnées pour obtenir le résultat final.

## LES PRINCIPES SOLID, DRY ET KISS

SOLID (4/5) :

SOLID est un acronyme pour cinq principes de conception de logiciels orientés objet.

S - Principe de Responsabilité Unique (Single Responsibility Principle) : Une classe ne doit avoir qu'une seule raison de changer.

O - Principe Ouvert/Fermé (Open/Closed Principle) : Les entités logicielles (classes, modules, fonctions, etc.) doivent être ouvertes pour extension mais fermées pour modification.

L - Principe de Substitution de Liskov (Liskov Substitution Principle) : Les sous-classes doivent être substituables à leur classe de base sans altérer le comportement du programme.

I - Principe d'Inversion de Dépendance (Dependency Inversion Principle) : Les modules de haut niveau ne doivent pas dépendre des modules de bas niveau. Les deux doivent dépendre d'abstractions.

D - Principe de Ségrégation d'Interface (Interface Segregation Principle) : Les clients ne doivent pas être forcés de dépendre d'interfaces dont ils n'ont pas besoin.

• Chaque Module créé en TypeScript s’occupe d’une seule fonction unique. Donc chaque module a un seul but dans son existence (SRP)

• Les types personnalisés qu’on a créés avec TypeScript (exemple : ItemData) peuvent être remplacés, par des objets qui ont leurs composantes (LSP)

• Chaque page ou document EJS a les fonctions dont il a besoin. Par exemple, la page de connexion, n’aura pas le script de menu de restaurant attaché à cette dernière. (ISP)

• L’exécution de l’application ne dépend pas d’un ou plusieurs modules TS. Au contraire, chaque module TS offre une fonctionnalité-clé de l’application (DIP)

DRY :

Le principe DRY (Don’t repeat yourself) dicte qu’il ne faut pas répéter la même logique d’exécution sous la forme de code, mais de s’assurer de le factoriser en un seul endroit pour éviter les duplications et rendre le code plus facile à maintenir. Dans le cas de BdebUberEats, le principe s’applique partiellement. En effet, si on prend l’exemple de créer une connexion avec la base de données Mongo Atlas, on peut utiliser la même logique derrière connectToMongo(), qui est un module implémenté dans le code. Cependant, même s’il y’a des interactions similaires dans le code (ex. : ajouter un item dans la BD ou ajouter un menu dans la BD), et que cette logique est similaire, on est quand même passé par de différentes fonctions, même si celle-ci avait le même but : une transaction avec la BD

KISS :

Le principe KISS (Keep it simple stupid) recommende de garder le code simple et facile à comprendre plutôt que de le rendre trop complexe, puisque ça le rend plus facile à débugger, à améliorer et, parfois, est plus performant que le code complexe. Dans le cas de BdebUberEats, le principe s’applique très bien, le code qu’on utilise est très simple. En effet, la majorité des modules décrits par le code a une de deux fonctions : faire une transaction avec des données externes (BD, API, etc.), ou la manipulation/vérification de données et de conditions. Chaque module, d’ailleurs divise ses responsabilités en sous-fonctions. Par exemple, le module d’inscription utilise plusieurs fonctions courtes, mais simples pour ajouter un utilisateur à la base de données Mongo.

# MÉDIAGRAPHIE

*DONT REPEAT YOURSELF*. (s.d.). Récupéré sur WIKIPEDIA: https://en.wikipedia.org/wiki/Don%27t\_repeat\_yourself

*SOLID*. (s.d.). Récupéré sur WIKIPEDIA: https://en.wikipedia.org/wiki/SOLID

WIKIPEDIA. (s.d.). *KISS principle*. Récupéré sur WIKIPEDIA: https://en.wikipedia.org/wiki/SOLID