Université Catholique de Lille 1875

2024/2025

Project Master Cyber

Livrable N°3

Fait par

Aftisse Amel

Encadreurs

Nicolas Beaussart

Charles Yaacoub

Belhassen Zouari

Table des matières

Rappel du contexte

Dans le cadre de ce projet de master, l'objectif était de concevoir et de développer un site web de gestion de stock et de livraisons, destiné à une petite entreprise familiale spécialisée dans la fabrication et la distribution de matériel pour les professionnels de la pâtisserie.

L'idée du projet est née à la suite d'un voyage en Algérie, durant lequel j'ai constaté que cette entreprise gérait l'ensemble de ses activités de façon entièrement manuelle : gestion des stocks, organisation des livraisons, suivi client, et paiements. Cette méthode traditionnelle engendrait un manque de visibilité, un risque important d'erreurs, ainsi que des pertes de temps conséquentes, sans justificatifs clairs.

Pour répondre à ces problématiques, j'ai proposé la mise en place d'une solution numérique, sous la forme d'un site web responsive, capable de centraliser et d'automatiser ces processus.

Les objectifs principaux du site web sont :
— Suivre en temps réel les niveaux de stock des matériaux disponibles.
— Planifier et suivre les livraisons réalisées par plusieurs livreurs.
— Améliorer la traçabilité des opérations internes.
— Gérer les paiements et générer automatiquement des justificatifs.

L'idée est donc de développer une sorte de mini-ERP, spécifiquement adapté aux besoins de mon client, afin de centraliser les informations essentielles de manière simple et efficace. Le nom du site web sera « maylisse-app » comme demandé par le client.

Même si des ERP bien connus comme Odoo existent déjà sur le marché, ils sont souvent coûteux, difficiles à prendre en main, et pensés pour des structures de grande taille. Leur interface est souvent surchargée de fonctionnalités, dont une bonne partie s'avère inutile pour une petite entreprise, rendant leur utilisation complexe sans formation préalable. Par ailleurs, la gestion des livraisons y est généralement conçue pour des systèmes logistiques industriels, loin des besoins concrets d'une entreprise artisanale.

À l'inverse, mon site web a été conçu sur mesure, avec une approche centrée sur la simplicité, l'efficacité et l'accessibilité. Il permet une gestion claire des stocks, un suivi précis des livraisons par livreur, et propose une interface intuitive utilisable directement depuis un ordinateur, sans compétences techniques particulières.

Rappel du besoin et des fonctionnalités

2.1 Objectifs

Ce projet vise à concevoir et développer un site web de gestion de stockage et de livraison pour une petite entreprise artisanale. L'objectif est de centraliser les opérations internes en offrant une solution numérique simple, intuitive et adaptée aux besoins réels du terrain.
Cette solution permet ainsi de remédier aux difficultés identifiées, telles que le manque de visibilité sur les stocks, les erreurs dans la gestion des livraisons, l'absence de suivi client structuré et de justificatifs de paiement, ainsi que les pertes de temps liées aux méthodes de gestion manuelle.

Les objectifs principaux sont :
— Surveiller les niveaux de stock en temps réel.
— Gérer et suivre les livraisons assurées par plusieurs livreurs.
— Justifier les modifications de quantités lors des livraisons.
— Gérer les paiements avec génération automatique de justificatifs.

Périmètre

Le projet couvre les fonctionnalités suivantes :
— Gestion des stocks : quantités, prix, historique.
— Suivi des livraisons : affectation, confirmation, historique.
— Prise en compte des besoins clients et préparation des commandes.
— Gestion des paiements et génération de factures.

Hors périmètre

— Gestion des fournisseurs et achats de matériel.
— Interface client pour passer commande.
— Application mobile dédiée (interface web responsive uniquement).

2.1.1 Fonctionnalités principales

Le nom de l'entreprise est Maylisse, j'ai donc nommé le site web « maylisse-app ».

1. Interfaces utilisateurs :
— *Administrateur* : accès complet à la gestion des stocks, livraisons, besoins clients, facturation et paiements.
— *Livreur* : accès restreint aux livraisons et tâches assignées.

2. Gestion de stock :
— Ajout, modification et suppression de matériel.
— Consultation de l'historique des modifications.
— Préparation de cartons client avec déduction automatique du stock.

3. Gestion des livraisons :
— Affectation et validation de livraisons.
— Confirmation avec signature du réceptionniste.
— Historique et suivi des livraisons (début/fin, retours, imprévus).

4. Justification des modifications de quantités :
— Motif, impact sur le stock, informations du réceptionniste (nom, entreprise, signature).

5. Paiements et facturation :
— Suivi des commandes validées.
— Génération automatique de factures complètes.

Méthodologie et organisation du travail

Pour mener à bien la réalisation de ce site web, j'ai utilisé un outil appelé Notion.so. Cet outil m'a permis de gérer efficacement les différentes tâches du projet, de centraliser les idées que j'ai eues, de documenter les pages que j'ai conçues, de conserver mes notes issues des échanges avec le client, ainsi que de planifier le temps nécessaire à chaque étape de développement. Chaque jour, j'avançais sur une partie du code, afin d'éviter toute surcharge et de coder avec plaisir et efficacité.

Voici un exemple concret de la manière dont j'ai utilisé cet outil :

creation de la base de donnée	Terminé	8 avril 2025 11:28
remplir la base de donnée	Terminé	14 avril 2025 11:10
creation page article	Terminé	20 avril 2025 23:26
créer les roles : admin et livreurs	Terminé	3 mai 2025 17:21
créer la page creer un article	Terminé	3 mai 2025 17:21
créer la page chercher commande	Terminé	3 mai 2025 17:21
creer la page creer livreur et client	Terminé	5 mai 2025 20:59
discussion avec le client	Terminé	6 mai 2025 13:16

Figure 3.1 Tâches réalisées avec notion.so

J'ai d'abord commencé par la création de la base de données, afin d'avoir les tables correspondant aux besoins et de pouvoir manipuler les données dans les différentes pages. Chaque page a été créée indépendamment, c'est-à-dire que je ne passais pas à une nouvelle page sans avoir terminé celle en cours de développement. Les idées de pages me venaient au fur et à mesure du développement, comme une suite logique permettant de finaliser l'ensemble des pages nécessaires. J'ai également modifié la base de données à plusieurs reprises, afin de l'adapter aux besoins exprimés et aux nouvelles remarques formulées par le client.

Chaque semaine, je discutais avec mon client pour lui présenter l'état d'avancement du projet, vérifier si cela lui convenait, et identifier les éventuelles modifications à apporter.

L'objectif principal de ce site est d'être user friendly (convivial pour l'utilisateur). Le style de chaque page a été défini à la fin du développement, car il était essentiel de valider d'abord les fonctionnalités et de s'assurer de leur bon fonctionnement avant de se concentrer sur l'aspect esthétique.

Outils et technologies

4.1 Supabase

Pour la partie backend et la gestion de la base de données, j'ai choisi d'utiliser **Supabase**. Il s'agit d'une plateforme open source qui propose une alternative moderne à Firebase, tout en s'appuyant sur PostgreSQL, une base de données relationnelle robuste et bien adaptée aux applications de gestion.

Supabase offre une intégration facile entre la base de données, l'authentification des utilisateurs, ce qui m'a permis de gagner un temps considérable dans le développement du projet.

Un autre avantage majeur de Supabase réside dans son système d'authentification basé sur des **tokens sécurisés**. Chaque utilisateur se voit attribuer un token JWT lors de la connexion, ce qui permet d'assurer la sécurité des échanges et de restreindre l'accès aux données sensibles en fonction du rôle de l'utilisateur (administrateur ou livreur).

Grâce à cette architecture, j'ai pu mettre en place un système simple, efficace et sécurisé, tout en limitant le temps passé sur la configuration serveur ou la gestion manuelle des connexions à la base de données.

4.2 Frontend, Styling et Tests

Le site a été développé avec **Next.js** et **TypeScript**. Next.js est un framework basé sur React qui offre une gestion efficace du rendu côté serveur (**SSR**) et côté client (**CSR**). Cela permet un affichage rapide des pages importantes, tout en conservant des interfaces dynamiques et interactives. Ce double rendu améliore à la fois la performance et l'expérience utilisateur.

L'organisation du projet repose sur la structure de fichiers de Next.js, avec des dossiers comme `admin` ou des routes dynamiques comme `[id]`. Cette architecture simplifie la gestion des pages et des rôles utilisateurs.

Pour le style, j'ai utilisé **Tailwind CSS**, un framework utilitaire qui permet de construire rapidement des interfaces propres, réactives et cohérentes, sans avoir à écrire de CSS classique.

Enfin, des tests unitaires ont été réalisés avec **Jest**, afin de valider les composants et les fonctions critiques du projet.

Cette combinaison d'outils permet un développement moderne, fiable et rapide.

Le développement

5.1 La base de données

5.1.1 Description des tables de la base de données

La base de données est structurée pour gérer efficacement les différents aspects du système de gestion de stockage et de livraison. Voici le rôle des principales tables :

- **profiles** : Cette table contient les profils des utilisateurs enregistrés via Supabase Auth. Chaque profil est associé à un `id` unique lié à un utilisateur authentifié et possède un rôle, soit `admin` (administrateur) soit `delivery_man` (livreur). Un trigger automatique crée une entrée dans cette table à la création d'un nouvel utilisateur.
- **article** : Table qui stocke les informations des matériels disponibles, avec leur nom, prix, quantité en stock, description, ainsi que les dates de création et de mise à jour.
- **client** : Contient les données des clients de l'entreprise, incluant nom, email, téléphone et adresse(de son local et pas forcément du lieu de livraison).
- **delivery_man** : Regroupe les informations sur les livreurs, avec leur pseudo unique, nom, prénom, coordonnées, adresse, statut (disponible ou indisponible), et dates de création/mise à jour.
- **order** : Table représentant les commandes passées. Chaque commande a un statut, une adresse de livraison, un livreur affecté (via son pseudo), un client (par son nom), une description de la commande et un commentaire éventuel du livreur à la fin de la livraison.
  Voici l'explication des status qui décrit les différentes étapes de vie d'une commande. Chaque statut correspond à une phase précise du traitement :

Statut	**Description**
initiated	La commande est créée, mais aucun article n'a encore été ajouté. C'est l'état initial lors de la saisie.
preparation	La commande est en cours de préparation, les articles commencent à être ajoutés.
prepared	Tous les articles demandés ont été ajoutés et la commande est finalisée, prête à être livrée.
delivering	Un livreur a pris en charge la commande pour la livraison.
delivered	La commande a été livrée à destination avec succès.
finished	La livraison est terminée ; le livreur doit ajouter un commentaire décrivant le déroulement de la livraison avec le client.
canceled	La commande a été annulée ou supprimée.

Table 5.1 Description des statuts d'une commande dans le système

— **order_article** : Table de liaison entre les commandes et les articles, indiquant la quantité commandée de chaque article, le prix appliqué, ainsi que les dates de création et mise à jour. La clé primaire est la combinaison de l'identifiant de la commande et de l'article.

5.1.2 Authentification avec Next.js et Supabase

Pour sécuriser l'accès au site, j'ai intégré un système d'authentification basé sur **Supabase Auth**, couplé à Next.js. Supabase utilise des **tokens JWT** pour gérer les sessions utilisateur de manière sécurisée et sans état (stateless).

L'intégration a suivi le tutoriel officiel Supabase - Authentification côté serveur avec Next.js, qui détaille la gestion des sessions en combinant les fonctionnalités côté client et côté serveur.

Le principe clé est d'utiliser le **rendu côté serveur (Server-Side Rendering, SSR)** dans Next.js pour vérifier l'authentification de l'utilisateur à chaque requête avant de générer la page HTML. Cela permet de :

— Récupérer et valider le **token JWT** de l'utilisateur à partir des cookies HTTP envoyés par le navigateur.
— Initialiser un client Supabase sécurisé côté serveur avec ce token pour interroger la base de données.
— Restreindre l'accès aux pages protégées en redirigeant les utilisateurs non authentifiés vers la page de connexion.
— Fournir dès le rendu initial les données personnalisées selon l'utilisateur connecté, améliorant la performance et le référencement naturel (SEO).

Côté client, le SDK Supabase gère la connexion, la déconnexion, et la mise à jour automatique des tokens dans le stockage local, assurant une expérience fluide sans rechargement excessif.

Pour garantir la sécurité, les tokens sont stockés dans des **cookies HTTP-only**, ce qui empêche leur lecture par des scripts malveillants (protection contre les attaques XSS). De plus, les rôles utilisateurs (`admin`, `delivery_man`) sont récupérés depuis la base de données et utilisés pour adapter dynamiquement l'interface et les autorisations.

Cette méthode combinée offre un bon équilibre entre performance, sécurité, et expérience utilisateur, tout en exploitant pleinement les capacités modernes de Next.js et Supabase.

5.1.3 Schéma de la base de données

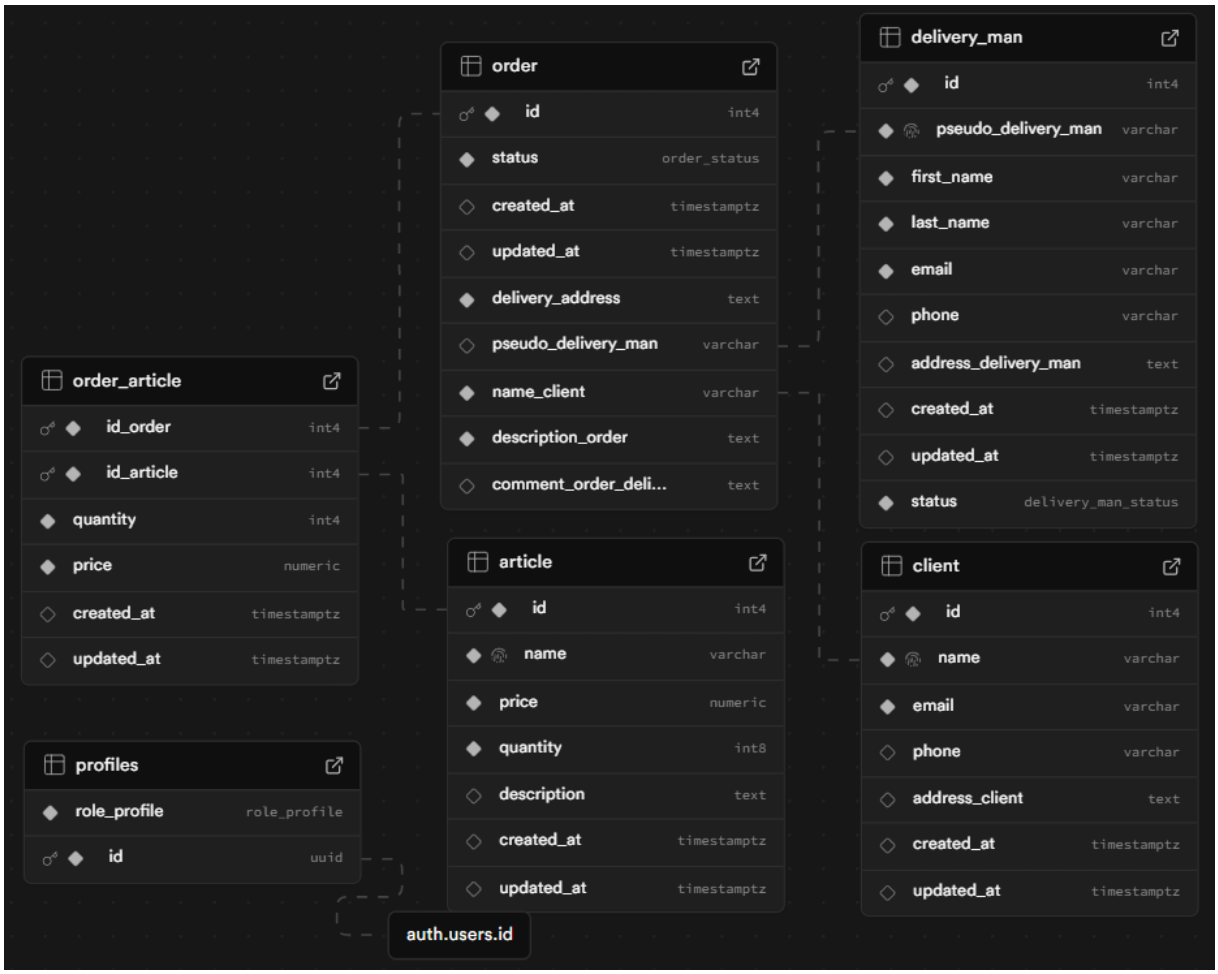

FIGURE 5.1 schéma de la base de donnée

5.2 Le code

Le projet a été développé sur vscode et a été lié à ce github : github où un README pour une explication détaillée de l'installation a été ajouté.

5.2.1 Architecture du projet

Dans cette section, une description succincte de la structure des dossiers du projet est fournie.

Avec Next.js, chaque page correspond à un fichier `page.tsx` situé dans un dossier portant le nom de la route souhaitée. Par exemple, pour afficher la page des articles, le fichier doit se trouver dans `article/page.tsx`.
Pour gérer les différents rôles utilisateurs, notamment les rôles `admin` et `livreur`, un dossier dédié `admin` contient toutes les pages réservées aux administrateurs. Les pages accessibles uniquement aux administrateurs sont regroupées dans ce dossier, tandis que les pages communes peuvent se situer en dehors. Grâce à la gestion des permissions basée sur les rôles, les livreurs ne peuvent pas accéder aux pages du dossier `admin`, conformément

aux restrictions définies.
Par ailleurs, pour accéder à une page dynamique correspondant à un identifiant spécifique, Next.js utilise la syntaxe des dossiers entre crochets. Par exemple, pour afficher une commande en fonction de son identifiant, on crée un fichier `page.tsx` dans le dossier `orderID/[id]/`, ce qui permet d'accéder à la route dynamique `/orderID/{id}`.

```
app/
├── (admin)/                    # Section administration
│   ├── article/                # Gestion des articles
│   │   ├── page.tsx            # Page principale des articles
│   │   └── action.ts           # Actions serveur pour les articles
│   ├── order/                  # Gestion des commandes
│   ├── client/                 # Gestion des clients
│   ├── delivery_man/           # Gestion des livreurs
│   ├── orderID/                # Détails d'une commande
│   │   └── [id]/               # Page spécifique d'une commande
│   │       ├── page.tsx        # Page de détails de la commande
│   │       └── action.ts       # Actions serveur pour la commande
│   ├── clientID/               # Détails d'un client
│   ├── create_delivery_man/    # Création de livreur
│   ├── create_client/          # Création de client
│   ├── create_order/           # Création de commande
│   ├── create_article/         # Création d'article
│   ├── admin/                  # Dashboard admin
│   └── layout.tsx              # Layout admin
├── ready_to_deliever/          # Commandes prêtes à livrer
├── delivery_manID/             # Détails livreur
├── login/                      # Page de connexion
├── private/                    # Section privée
├── insert_data_articles.ts     # Script d'insertion d'articles
├── favicon.ico                 # Icône du site
├── globals.css                 # Styles globaux
├── layout.tsx                  # Layout principal
```

Figure 5.2 architecture du dossier app

Voici les composants utilisés, ainsi que le dossier contenant les appels génériques aux tables pour récupérer leurs données :

```
components/
├── NavigationBar.tsx              # Barre de navigation principale
├── DeliveryManNavigationBar.tsx   # Barre de navigation pour les livreurs
├── Input.tsx                      # Composant de champ de saisie réutilisable
├── Select.tsx                     # Composant de sélection réutilisable
├── SearchBarDeliveryMan.tsx       # Barre de recherche pour les livreurs
├── SearchBarClients.tsx           # Barre de recherche pour les clients
├── SearchBarArticles.tsx          # Barre de recherche pour les articles
└── SearchBarOrders.tsx            # Barre de recherche pour les commandes

datas/
├── data_delivery_men.ts      # Données des livreurs
├── data_clients.ts           # Données des clients
├── data_orders.ts            # Données des commandes
└── data_articles.ts          # Données des articles
```

Figure 5.3 architecture de components et datas

Tests unitaires

Quelques tests unitaires ont été réalisés, et voici l'architecture du dossier __test__

```
__test__/
├── components/          # Tests des composants React
├── pages/               # Tests des pages
└── datas/               # Tests des données
```

FIGURE 5.4 architecture dossier test

5.2.2 Quelques illustrations du rendu

La page admin ressemble à ça :

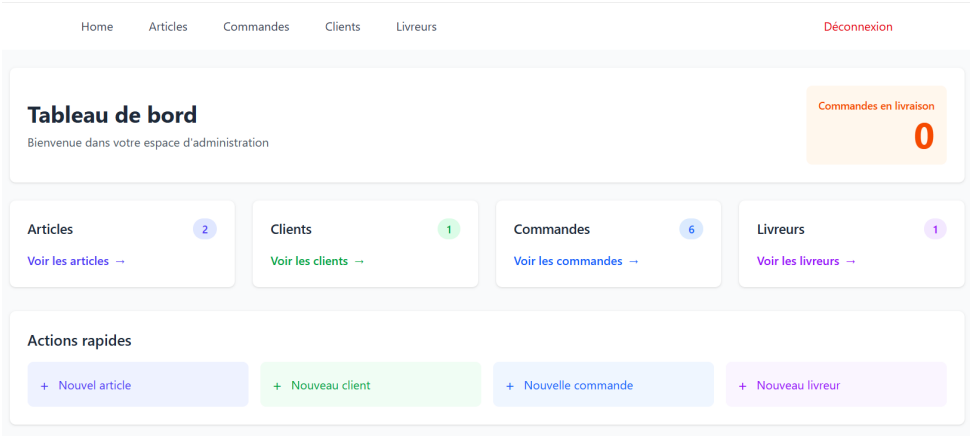

FIGURE 5.5 page admin

La page article ressemble à ça :

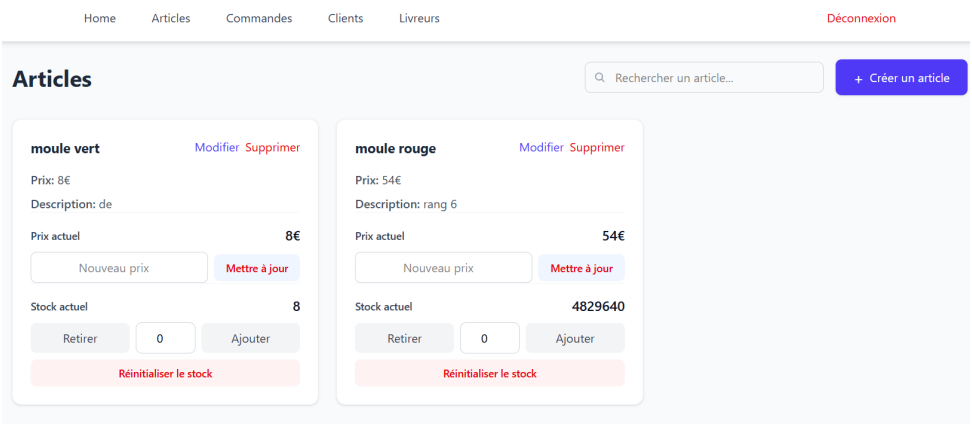

FIGURE 5.6 page article

La page commande ressemble à ça :

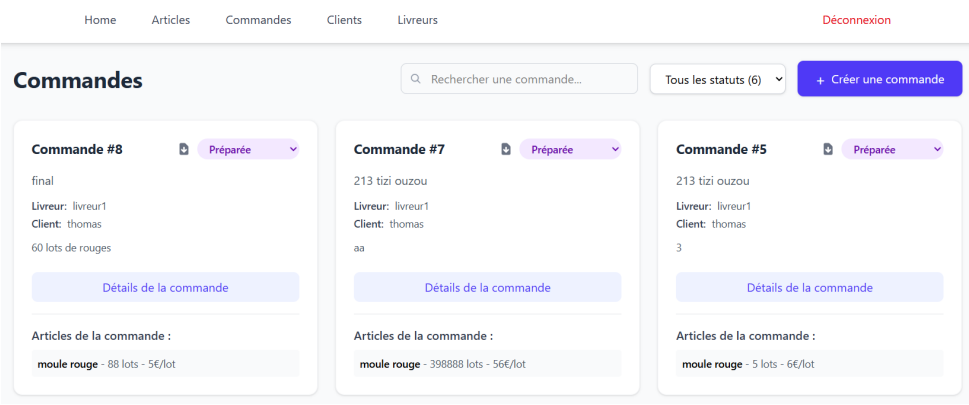

Figure 5.7 page commande

La page pour créer les articles ressemble à ça :

Figure 5.8 page créer article

Bilan des objectifs

6.1 Objectifs atteints

L'un des principaux objectifs de ce site web est la gestion du stock, des livraisons et de la facturation client. Voici les fonctionnalités actuellement prises en charge par le site :

Deux interfaces :
Admin qui a un accès complet à toutes les fonctionnalités. La création des livreurs doit obligatoirement être effectuée par l'administrateur, via la page dédiée, afin d'enregistrer leur adresse email et de leur attribuer un compte.

Livreur : aussi est faite, dedans ils retrouveront chaque commande prete à être livrée. Grace à un bouton, ils changeeront le statut de la commande à livraison et à la fin de livriason.

Gestion des Clients
Création et modification de profils clients - génération de factures par client - Recherche et filtrage des clients - Gestion des adresses de livraison.

Gestion des Commandes
Création de nouvelles commandes - Suppression, modification- Génération de factures PDF personnalisées - Suivi de l'état des commandes - Toutes les commandes par date - Filtrage des commandes par statut.

Gestion des Articles
Création, suppression, modification - Gestion des stocks - prix et descriptions - historique des prix.

Gestion des Livreurs
Création, suppression, modification - Attribution des commandes - suivi en temps réel grace au statut - historique des livreurs - interface dédiée à chaque livreur.

Interface d'Administration
Tableau de bord - gestion des utilisateurs - paramètres de l'application - rapports et analyses.

Système d'Authentification
Connexion sécurisée - gestion des sessions - niveaux d'accès (admin, livreur).

6.2 Objectifs non atteints

Certaines fonctionnalités prévues ne sont pas encore implémentées à ce jour :

- **Vérification des paiements** : le site ne permet pas encore de vérifier automatiquement si les paiements ont bien été effectués. Cela nécessiterait une connexion à l'API de paiement, que le client doit encore fournir.

- **Historique détaillé des prix** : bien qu'il soit possible de retrouver le prix d'une commande à partir de la facture téléchargée (puisqu'il est pris en compte à chaque génération), le système ne conserve pas un historique structuré de tous les prix définis pour

chaque article au fil du temps.

- **Fonction "Mot de passe oublié"** : le bouton est présent dans l'interface, mais ne fonctionne pas encore. Il devra être relié à un système d'envoi de lien sécurisé par email pour permettre la réinitialisation du mot de passe.

6.3 Difficultés

Au cours du développement, plusieurs difficultés techniques et organisationnelles se sont présentées.

L'une des premières a été de comprendre le fonctionnement du **rendu côté serveur avec Next.js**. N'ayant jamais travaillé avec ce framework auparavant, j'ai dû m'adapter à une nouvelle logique de développement, notamment la gestion du Server-Side Rendering (SSR) et des routes dynamiques.

L'utilisation de **Supabase** a également représenté un défi. J'ai été amené à modifier à plusieurs reprises la structure de la base de données pour répondre aux besoins du projet. Cela impliquait de bien comprendre les relations entre les tables, la gestion des rôles et des permissions, ainsi que les contraintes imposées par le service.

La collaboration avec le client a parfois nécessité des **modifications fréquentes**, voire des refontes partielles de certaines fonctionnalités. Ces ajustements imprévus demandaient de la flexibilité et rallongeaient souvent les délais prévus initialement.

La **gestion du temps** a constitué un autre défi. Il fallait organiser les tâches de manière efficace tout en faisant face à des **bugs parfois complexes à identifier**, souvent causés par de petits détails difficiles à repérer sans une analyse approfondie.

Enfin, certaines fonctionnalités, comme le bouton **"Mot de passe oublié"**, n'ont pas pu être finalisées à temps, ce qui souligne l'importance d'une meilleure planification et d'une marge pour les imprévus.

Amélioration

Plusieurs évolutions sont prévues pour améliorer l'expérience utilisateur et automatiser davantage certaines fonctionnalités :

- **Création d'une interface dédiée au client** : actuellement absente, cette interface permettrait aux clients de consulter leurs commandes, leurs factures, et de suivre l'état de leur livraison en temps réel.

- **Autonomie des livreurs** : l'objectif est de permettre aux livreurs de gérer eux-mêmes leurs livraisons via leur interface, sans dépendre de l'attribution manuelle par l'administrateur.

- **Activation de la fonctionnalité "Mot de passe oublié"** : bien que présente dans l'interface, cette fonctionnalité n'est pas encore opérationnelle. Il s'agira de la connecter à un système d'envoi de mail sécurisé pour la réinitialisation des mots de passe.

- **Ajout de photos pour les articles** : intégrer des images aux produits améliorerait la présentation du catalogue et offrirait une meilleure expérience utilisateur, notamment pour les clients.

- **Développement d'une application mobile** : pour rendre le service plus accessible, notamment pour les livreurs en déplacement, une version mobile de l'application est envisagée, avec un accès simplifié aux fonctionnalités essentielles.

- **Vérification du paiement** : mettre en place un système permettant de savoir si un paiement a bien été effectué, via une connexion à l'API de paiement fournie par le client, afin de garantir la fiabilité du suivi des commandes.

Conclusion

Ce projet m'a permis de développer une application web complète pour la gestion du stock, des livraisons et de la facturation, en utilisant des technologies modernes telles que Next.js et Supabase. J'ai pu approfondir mes connaissances techniques, notamment en ce qui concerne le rendu côté serveur, la gestion des bases de données et la mise en place d'interfaces utilisateur adaptées aux besoins du client.

Toutefois, plusieurs défis ont été rencontrés au cours du développement. L'adaptation à de nouvelles technologies, les nombreuses modifications demandées par le client, ainsi que la gestion de bugs parfois difficiles à identifier ont représenté des obstacles importants. De plus, certaines fonctionnalités essentielles, comme la vérification automatique des paiements ou la réinitialisation sécurisée des mots de passe, restent à implémenter.

Pour l'avenir, des améliorations importantes sont envisagées, telles que la création d'une interface client dédiée, l'autonomie des livreurs dans la gestion des livraisons, l'ajout de photos aux articles, et le développement d'une application mobile. Ces évolutions permettront d'optimiser l'expérience utilisateur, de rendre le système plus flexible et de renforcer la fiabilité du suivi des commandes.

En somme, ce livrable constitue une base solide sur laquelle construire un outil performant et évolutif, répondant aux besoins actuels et futurs des utilisateurs. Le travail accompli offre une expérience précieuse et pose les fondations d'un développement continu vers une solution toujours plus complète et intuitive.