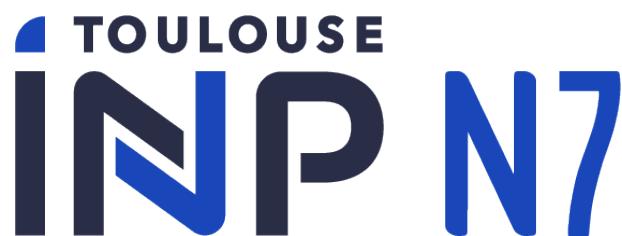

Rapport du Projet A Pizz'app

Application Web

Matthis Bernardini, Enzo Baïta, Julien Gauthier, Sami Mostfa

5 juin 2025



Sommaire

1	Introduction	2
2	Système d'Authentification	2
2.1	Implémentation	2
2.2	Déroulement des opérations	2
3	Page d'Accueil	3
3.1	Implémentation	3
3.2	Déroulement des opérations	3
4	Inscription Utilisateur	3
4.1	Implémentation	3
4.2	Déroulement des opérations	4
5	Menu et Sélection de Pizzas	4
5.1	Implémentation	4
5.2	Déroulement des opérations	5
6	Panier et Récapitulatif de Commande	6
6.1	Implémentation	6
6.2	Déroulement des opérations	6
7	Tableaux de Bord et Gestion des Rôles	7
7.1	Architecture et Implémentation	7
7.2	Fonctionnalités	7
7.3	Sécurité et Contrôle d'Accès	8
7.4	Déroulement des opérations	8
8	Planning	8
8.1	Implémentation	8
8.2	Déroulement des opérations	8
9	Gestion des Ingrédients	9
9.1	Implémentation	9
9.2	Déroulement des opérations	9
10	Comptabilité	10
10.1	Implémentation	10
10.2	Statistiques et Analyses	11
10.3	Déroulement des opérations	11
11	Déploiement Docker	12
11.1	Implémentation	12
11.2	Déroulement des opérations	12
12	Conclusion	12

1 Introduction

A Pizz'app est une application web dédiée à la commande de pizzas, pensée dès sa conception pour être réellement utile dans la pizzeria des parents de Matthias. Elle offre une interface intuitive permettant aux clients de parcourir le menu, personnaliser leurs pizzas avec différents ingrédients, passer commande et planifier leur retrait. L'application permet aussi de pouvoir commander pour un client au téléphone par exemple, de gérer le stock des ingrédients ou encore en tant qu'administrateur de voir les résultats d'une journée de vente.

Sur le plan technique, A Pizz'app repose sur une architecture avec Angular pour le frontend, Java Spring Boot pour le backend et PostgreSQL comme système de gestion de base de données. L'ensemble de l'application est conteneurisé via Docker, facilitant ainsi son déploiement. La structure du projet suit une architecture client-serveur classique avec une API RESTful qui permet la communication entre le frontend et le backend.

L'architecture de l'application tourne autour d'un système multi-utilisateurs avec différents niveaux d'accès (client, opérateur, administrateur) qui définissent les fonctionnalités disponibles. La base de données PostgreSQL stocke les informations des utilisateurs, des pizzas, des ingrédients et des commandes.

2 Système d'Authentification

Le système d'authentification permet aux utilisateurs de se connecter à l'application avec différents rôles (client, opérateur, administrateur) déterminant leurs priviléges et accès aux fonctionnalités.

2.1 Implémentation

Les principaux fichiers impliqués sont :

- `frontend/src/app/features/landing/landing.component.ts` - Gestion du formulaire de connexion
- `frontend/src/app/services/auth.service.ts` - Service d'authentification
- `backend/src/main/java/com/apizzapp/controller/AuthController.java` - Contrôleur d'authentification backend
- `backend/src/main/java/com/apizzapp/model/User.java` - Modèle utilisateur

La méthode `onLogin()` du `LandingComponent` récupère les identifiants du formulaire et les envoie au `AuthService` qui communique avec le backend. Le contrôleur d'authentification vérifie les identifiants et renvoie un objet utilisateur contenant son rôle, permettant une redirection vers le tableau de bord approprié.

2.2 Déroulement des opérations

1. L'utilisateur saisit son email et mot de passe dans le formulaire de connexion
2. La méthode `onLogin()` du composant `LandingComponent` est déclenchée lors de la soumission
3. Le `AuthService` envoie une requête HTTP au backend avec les identifiants
4. Le backend vérifie les identifiants et renvoie les informations de l'utilisateur avec son rôle
5. En fonction du rôle (ROLE_ADMIN, ROLE_OPERATOR, etc.), l'utilisateur est redirigé vers le tableau de bord correspondant
6. En cas d'échec, un message d'erreur s'affiche

Les mots de passe dans la base de données sont cryptés voici 3 comptes pour se connecter avec les différents rôles :

Rôle	Email	Mot de passe	Accès
Utilisateur	User@email.com	user	Accès aux fonctionnalités client
Opérateur	Operateur@email.com	operateur	Accès au tableau de bord opérateur
Administrateur	Admin@email.com	admin	Accès complet à toutes les fonctionnalités

TABLE 1 – Comptes de démonstration disponibles pour tester l'application

Remarque : Ces comptes sont préchargés dans la base de données à des fins de démonstration. En production, il conviendrait d'utiliser des mots de passe plus sécurisés.

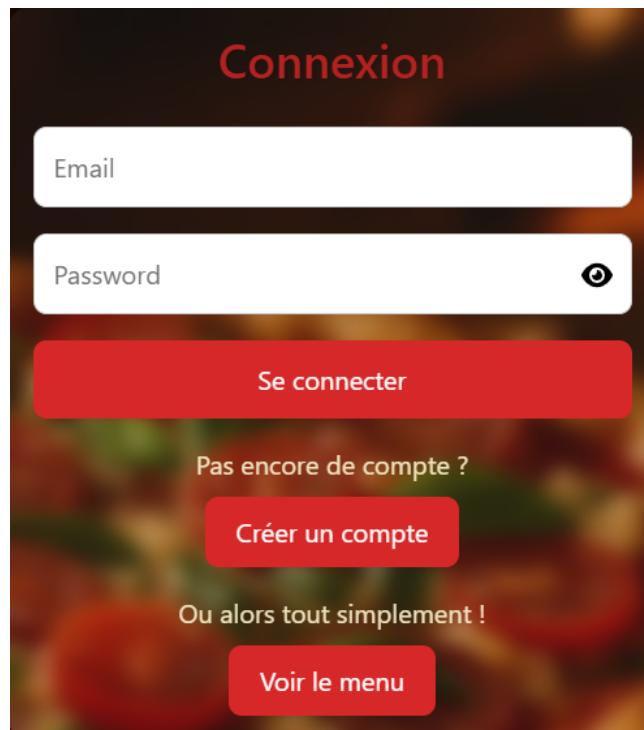


FIGURE 1 – Formulaire de connexion.

3 Page d'Accueil

La page d'accueil est le point d'entrée de l'application, présentant le formulaire de connexion. Elle permet aux utilisateurs de naviguer vers le menu, de s'authentifier ou de créer un compte.

3.1 Implémentation

Les principaux fichiers impliqués sont :

- `frontend/src/app/features/landing/landing.component.ts` - Logique de la page d'accueil
- `frontend/src/app/features/landing/landing.component.html` - Template HTML
- `frontend/src/app/features/landing/landing.component.css` - Styles spécifiques

3.2 Déroulement des opérations

1. L'utilisateur arrive sur la page d'accueil
2. L'utilisateur peut choisir de se connecter via le formulaire d'authentification
3. Ou naviguer directement vers le menu ou la page de création de compte via le bouton dédié (voir figure 1)

4 Inscription Utilisateur

La fonctionnalité d'inscription permet aux nouveaux utilisateurs de créer un compte sur A Pizz'App.

4.1 Implémentation

Les principaux fichiers impliqués sont :

- `frontend/src/app/features/auth/register/register.component.ts` - Logique d'inscription
- `frontend/src/app/features/auth/register/register.component.html` - Template du formulaire d'inscription
- `frontend/src/app/services/auth.service.ts` - Service d'authentification gérant l'inscription

- backend/src/main/java/com/apizzapp/controller/AuthController.java - Contrôleur backend pour l'inscription

Le composant `RegisterComponent` gère un formulaire avec validation pour la création de compte. Il collecte les informations utilisateur comme le nom, prénom, email, mot de passe et adresse, puis les transmet au backend via le `AuthService`.

La méthode `onRegister()` vérifie la validité du formulaire et envoie les données au service d'authentification qui communique avec le backend pour créer un nouveau compte utilisateur.

4.2 Déroulement des opérations

1. L'utilisateur accède à la page d'inscription via un lien sur la page d'accueil
2. Il remplit le formulaire avec ses informations personnelles
3. Le système valide en temps réel les entrées (format d'email, mot de passe, etc.)
4. À la soumission, la méthode `onRegister()` est déclenchée
5. Le service d'authentification envoie une requête HTTP POST au backend avec les données utilisateur
6. Le backend vérifie que l'email n'est pas déjà utilisé
7. Si les vérifications sont réussies, un nouveau compte est créé avec le rôle USER par défaut
8. L'utilisateur est automatiquement redirigé vers la page d'accueil.
9. En cas d'erreur (email déjà utilisé, problème de serveur), un message approprié s'affiche



FIGURE 2 – Formulaire d'Inscription.

5 Menu et Sélection de Pizzas

Cette fonctionnalité permet aux utilisateurs de consulter le menu des pizzas disponibles, de personnaliser leurs choix en ajoutant ou retirant des ingrédients et d'ajouter des pizzas à leur panier.

5.1 Implémentation

Les principaux fichiers impliqués sont :

- frontend/src/app/features/menu/menu.component.ts - Logique du menu
- frontend/src/app/features/menu/menu.component.html - Template du menu

- backend/src/main/java/com/apizzapp/controller/PizzaController.java - Contrôleur backend pour les pizzas
- backend/src/main/java/com/apizzapp/model/Pizza.java - Modèle de données pour les pizzas

Le contrôleur `PizzaController` expose des endpoints REST qui permettent de récupérer la liste des pizzas et des ingrédients disponibles. Le frontend utilise ces données pour afficher le menu et permettre la personnalisation des pizzas.

5.2 Déroulement des opérations

1. L'utilisateur accède à la page du menu
2. Le frontend fait une requête au backend pour récupérer la liste des pizzas et ingrédients
3. L'utilisateur peut parcourir les pizzas disponibles
4. Pour chaque pizza, l'utilisateur peut ajouter ou retirer des ingrédients
5. L'utilisateur peut ajouter la pizza personnalisée à son panier
6. Le frontend met à jour l'état local du panier



FIGURE 3 – Menu.



FIGURE 4 – Personnalisation.

6 Panier et Récapitulatif de Commande

Cette fonctionnalité permet aux utilisateurs de visualiser leur panier, d'ajuster les quantités, et de finaliser leur commande en saisissant leurs informations personnelles et en choisissant un créneau horaire pour le retrait.

6.1 Implémentation

Les principaux fichiers impliqués sont :

- `frontend/src/app/features/menu/recap-commande/recap-commande.component.html` - Template du récapitulatif
- `frontend/src/app/features/menu/recap-commande/recap-commande.component.ts` - Logique du récapitulatif
- `backend/src/main/java/com/apizzapp/controller/PizzaController.java` - Gestion des commandes côté backend

Le composant `RecapCommandeComponent` affiche les pizzas sélectionnées avec leurs suppléments et permet d'ajuster les quantités via les méthodes `incQuantity()` et `decQuantity()`. Un formulaire permet de saisir les informations de commande (nom, prénom, heure de retrait) qui sont validées avant l'envoi au backend.

Le formulaire s'adapte en fonction du statut d'authentification de l'utilisateur. Les champs nom et prénom ne sont affichés que si l'utilisateur n'est pas connecté ou s'il est connecté en tant qu'administrateur ou opérateur :

6.2 Déroulement des opérations

1. L'utilisateur accède à la page de récapitulatif de commande
2. Le panier s'affiche avec les pizzas sélectionnées et leurs suppléments
3. L'utilisateur peut ajuster les quantités ou retourner au menu pour ajouter d'autres pizzas
4. L'utilisateur remplit le formulaire avec ses informations personnelles et choisit un créneau horaire
5. La méthode `validateOrder()` vérifie la validité du formulaire et envoie la commande au backend
6. En cas de succès, l'utilisateur est redirigé vers une page de confirmation

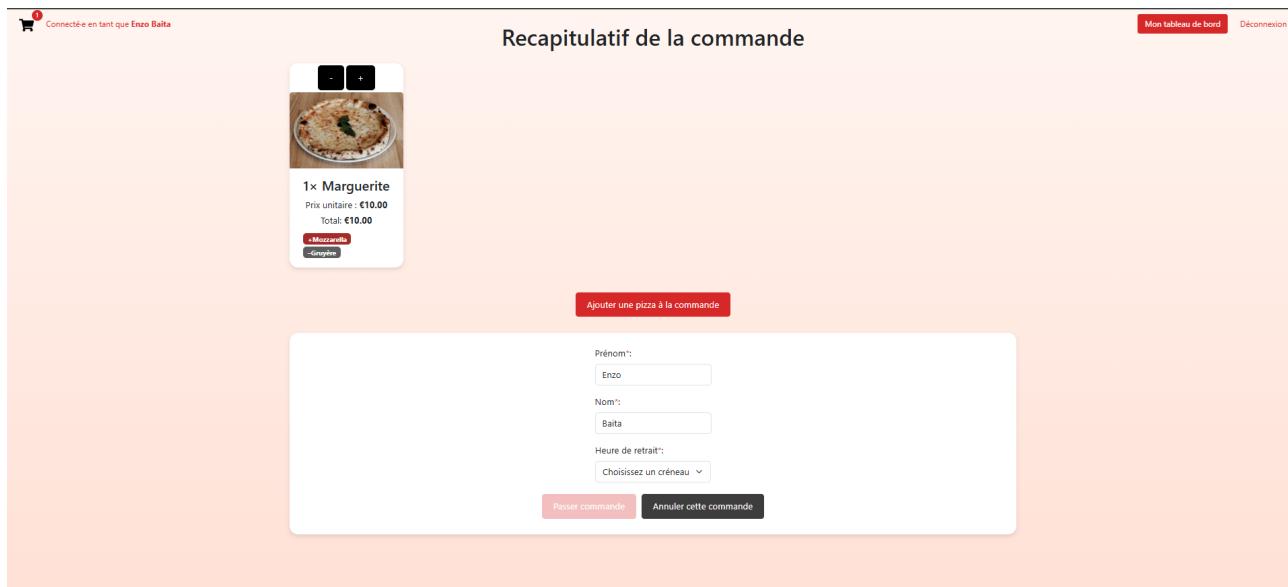


FIGURE 5 – Récapitulatif de Commande.

7 Tableaux de Bord et Gestion des Rôles

L’application A Pizz’app implémente un système de tableaux de bord différenciés selon le rôle de l’utilisateur, offrant ainsi une interface adaptée aux besoins spécifiques de chaque type d’utilisateur (client, opérateur, administrateur).

7.1 Architecture et Implémentation

Les tableaux de bord sont organisés selon une structure hiérarchique basée sur les rôles :

- `frontend/src/app/features/dashboards/` - Dossier principal contenant tous les tableaux de bord
- `frontend/src/app/features/dashboards/user-dashboard/` - Tableau de bord client
- `frontend/src/app/features/dashboards/operator-dashboard/` - Tableau de bord opérateur
- `frontend/src/app/features/dashboards/admin-dashboard/` - Tableau de bord administrateur
- `frontend/src/app/services/auth.service.ts` - Service gérant l’authentification et les rôles
- `backend/src/main/java/com/apizzapp/model/ERole.java` - Enumération des rôles côté backend

Lors de la connexion, le système identifie le rôle de l’utilisateur et le redirige vers le tableau de bord approprié.

Cette architecture permet une séparation claire et chaque utilisateur accède uniquement aux fonctionnalités pertinentes pour son rôle.

7.2 Fonctionnalités

Tableau de Bord Client Offre des fonctionnalités orientées consommateur :

- Accès au menu pour passer une nouvelle commande

Tableau de Bord Opérateur Destiné au personnel de la pizzeria :

- Gestion du planning des commandes
- Gestion des ingrédients disponibles
- Possibilité de commander pour un client

Tableau de Bord Administrateur Pour les gérants de l’établissement :

- Toutes les fonctionnalités du tableau de bord opérateur
- Gestion de la comptabilité

7.3 Sécurité et Contrôle d'Accès

Le système implémente des gardes de route Angular pour empêcher l'accès non autorisé aux différents tableaux de bord

7.4 Déroulement des opérations

1. L'utilisateur se connecte via le formulaire de connexion
2. Le backend authentifie l'utilisateur et renvoie ses informations, incluant son rôle
3. Le frontend redirige automatiquement vers le tableau de bord correspondant au rôle
4. L'interface s'adapte pour présenter uniquement les fonctionnalités autorisées
5. Les gardes de route empêchent l'accès direct aux URL des tableaux de bord non autorisés

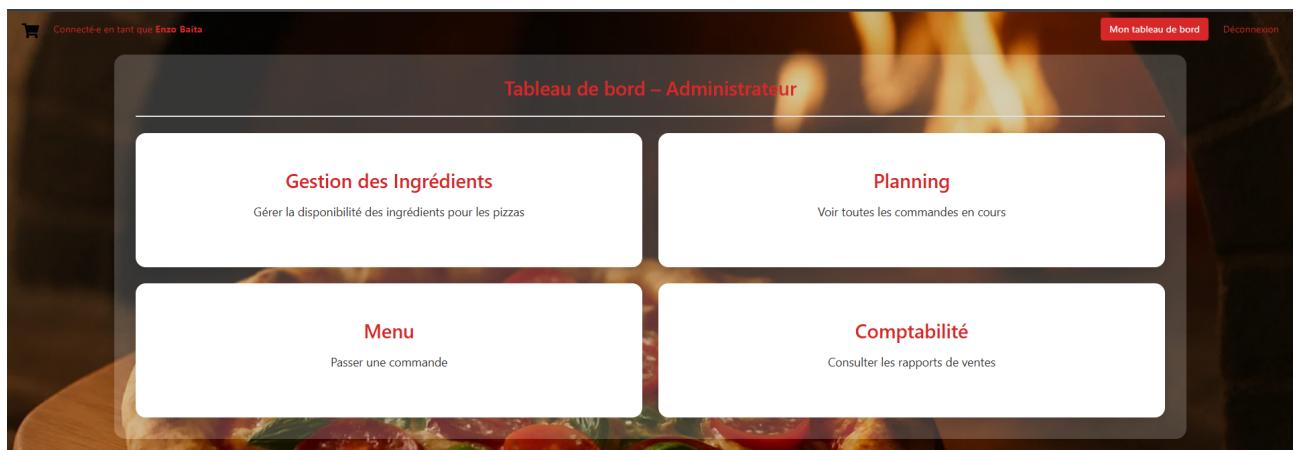


FIGURE 6 – Tableau de bord d'un administrateur.

8 Planning

Le planning est un élément de l'application A Pizz'app qui permet aux opérateurs et administrateurs de visualiser et gérer les commandes planifiées par créneau horaire, ce planning est surtout utile au pizzaïolo qui prépare les pizzas.

8.1 Implémentation

Les principaux fichiers impliqués sont probablement :

- `frontend/src/app/features/planning/planning.component.ts` - Logique du planning
- `frontend/src/app/features/planning/planning.component.html` - Interface utilisateur du planning
- `frontend/src/app/services/planning.service.ts` - Service de gestion des données du planning
- `backend/src/main/java/com/apizzapp/controller/PizzaController.java` - Contrôleur backend

Seuls les utilisateurs ayant les rôles `ROLE_OPERATOR` et `ROLE_ADMIN` peuvent accéder et gérer le planning, tandis que les clients réguliers (`ROLE_USER`) peuvent uniquement sélectionner un créneau horaire lors de leur commande.

8.2 Déroulement des opérations

1. L'opérateur ou l'administrateur accède au planning depuis son tableau de bord
2. Le système affiche la vue du planning avec les commandes regroupées par créneau horaire
3. Modification de l'état de la commande en cliquant sur le bouton d'état
4. Possibilité de supprimer la commande en base de données
5. Possibilité de modifier la commande (Pas implémenté)

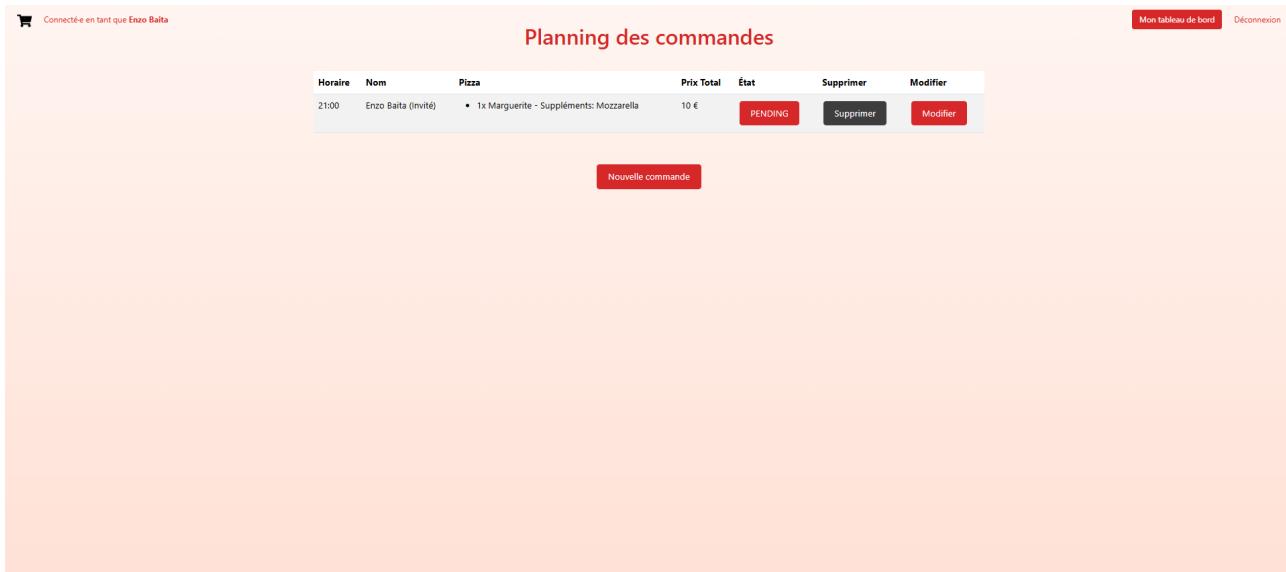


FIGURE 7 – Planning.

9 Gestión des Ingrédients

Cette fonctionnalité permet aux opérateurs et administrateurs de gérer les ingrédients disponibles pour la préparation des pizzas, en ajustant leur disponibilité (Disponible ou non). On pourrait penser que à chaque fois qu'une pizza est commandé on diminue le stock d'ingrédient dans la base de donnée pour calculer s'il reste ou non des ingrédients mais en réalité il est trop difficile de savoir quel quantité d'ingrédient on met sur une pizza donc dans une vraie pizzeria comme celle des parents de Matthias on ne signifie que les ingrédients qui manque pour la recette et il faudra alors en racheter.

9.1 Implémentation

Les principaux fichiers impliqués sont :

- frontend/src/app/features//ingredients/ingredients.component.ts
- frontend/src/app/features/ingredients/ingredients.component.html
- backend/src/main/java/com/apizzapp/controller/PizzaController.java
- backend/src/main/java/com/apizzapp/model/Ingredient.java

Le contrôleur backend permet de récupérer, ajouter, modifier et supprimer des ingrédients. Le frontend permet de visualiser et gérer ces ingrédients.

9.2 Déroulement des opérations

1. L'opérateur/administrateur accède à la page de gestion des ingrédients
2. Le système affiche la liste des ingrédients disponibles avec leur statut
3. L'utilisateur peut modifier la disponibilité des ingrédients
4. Les modifications sont envoyées au backend qui met à jour la base de données
5. Le frontend reflète immédiatement les changements, dans le menu les pizzas avec ces ingrédients ne sont plus disponibles et de même pour les suppléments.

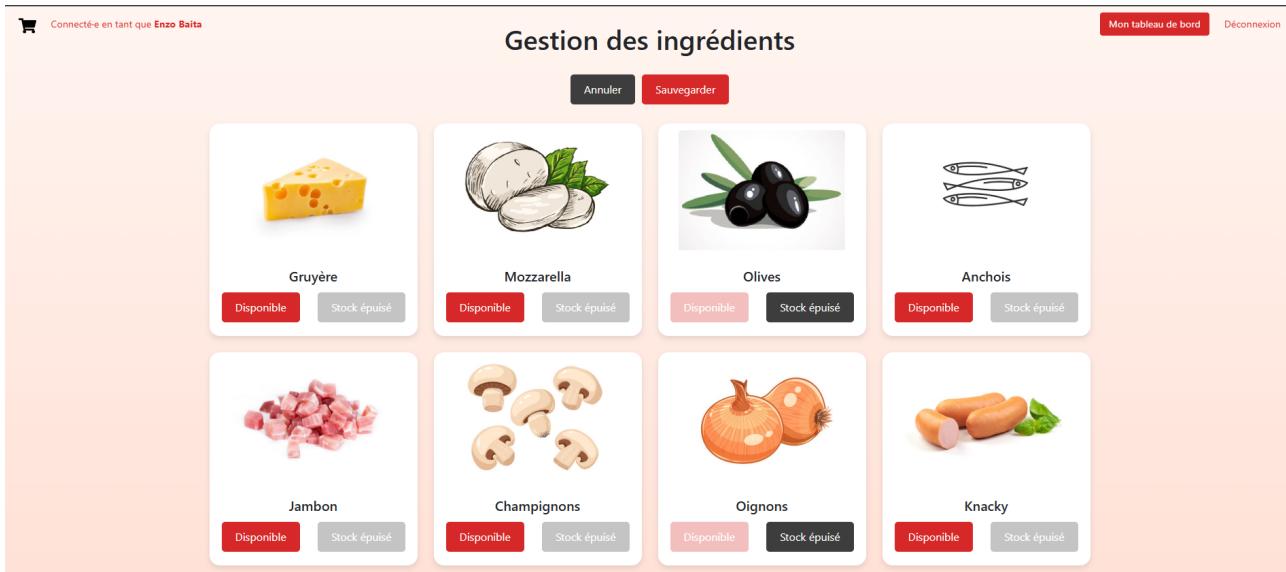


FIGURE 8 – Gestion des ingrédients.



FIGURE 9 – Menu2.

10 Comptabilité

La page de comptabilité d'A Pizz'app offre aux administrateurs une vue d'ensemble des performances financières et opérationnelles de la pizzeria. Cette fonctionnalité permet de suivre les revenus, d'analyser les tendances de vente et d'identifier les produits les plus populaires.

10.1 Implémentation

Les principaux fichiers impliqués dans cette fonctionnalité sont :

- `frontend/src/app/features/comptabilite/comptabilite.component.ts` - Logique de la page comptabilité
- `frontend/src/app/features/comptabilite/comptabilite.component.html` - Template de l'interface utilisateur
- `frontend/src/app/services/planning.service.ts` - Service récupérant les données des commandes
- `backend/src/main/java/com/apizzapp/controller/PizzaController.java` - côtés backend.

Le composant ComptabiliteComponent utilise le PlanningService pour récupérer toutes les commandes et calcule diverses statistiques à partir de ces données.

10.2 Statistiques et Analyses

- **Chiffre d'affaires total** - Somme des montants de toutes les commandes
- **Nombre total de commandes** - Décompte des commandes passées
- **Panier moyen** - Valeur moyenne des commandes
- **Nombre total de pizzas vendues** - Quantité totale de pizzas commandées
- **Top 5 des pizzas** - Les cinq pizzas les plus vendues avec leur quantité et le revenu généré
- **Distribution par créneau horaire** - Répartition des commandes selon l'heure de retrait

Ces statistiques sont précieuses pour l'optimisation des opérations, comme la gestion des stocks d'ingrédients et la planification du personnel. Par exemple, en identifiant les créneaux horaires les plus chargés, le gérant peut ajuster les effectifs en conséquence.

10.3 Déroulement des opérations

1. L'administrateur se connecte au système avec ses identifiants
2. Il accède au tableau de bord administrateur
3. En cliquant sur la carte "Comptabilité", il est redirigé vers la page de comptabilité
4. Le système charge les données des commandes depuis le backend
5. Les statistiques sont calculées automatiquement et affichées
6. L'administrateur peut actualiser les données à tout moment via un bouton dédié

The screenshot shows a dashboard titled 'Comptabilité' with a subtitle 'Statistiques des commandes'. It displays four key metrics: 'Commandes' (1), 'Chiffre d'affaires' (€10.00), 'Panier moyen' (€10.00), and 'Pizzas vendues' (1). Below these are two tables: 'Top 5 des pizzas' (showing Marguerite sold 1 unit for €8.50) and 'Répartition par créneau horaire' (showing 21:00 with 1 command). A red button at the bottom right says 'Actualiser les données'.

Pizza	Quantité vendue	Revenu généré
Marguerite	1	€8.50

Créneau horaire	Nombre de commandes
21:00	1

FIGURE 10 – Comptabilité.

11 Déploiement Docker

L'application est entièrement conteneurisée avec Docker, facilitant son déploiement et sa mise à l'échelle dans différents environnements.

11.1 Implémentation

Les principaux fichiers impliqués sont :

- `docker-compose.yml` - Configuration des services
- `backend/Dockerfile` - Instructions de build pour le backend
- `frontend/Dockerfile` - Instructions de build pour le frontend

11.2 Déroulement des opérations

1. Le développeur exécute `docker compose up -build` pour construire et démarrer tous les services
2. Docker crée d'abord le conteneur de base de données PostgreSQL
3. Ensuite, il construit et démarre le backend Spring Boot qui se connecte à la base de données
4. Enfin, il construit et démarre le frontend Angular accessible sur le port 4200
5. Le développeur peut voir les changements du code frontend en temps réel
6. Pour les modifications du backend ou de la base de données, une reconstruction est nécessaire

Voir le README.md pour plus d'information sur comment déployer l'application

12 Conclusion

A Pizz'app représente une solution pour la gestion des commandes de pizzas. L'application a réussi à implémenter toutes les fonctionnalités essentielles : affichage du menu, personnalisation des pizzas, gestion du panier, authentification des utilisateurs avec différents rôles, et tableaux de bord spécifiques pour chaque type d'utilisateur. La conteneurisation avec Docker facilite le déploiement et le développement.

Pour les améliorations futures, plusieurs pistes peuvent être envisagées :

- Implémentation d'un système de paiement en ligne intégré
- Ajout d'une fonctionnalité de suivi en temps réel de la préparation des commandes
- Développement d'un système de fidélité avec points et récompenses
- Amélioration de l'interface mobile pour une expérience optimisée sur smartphone

Le développement d'A Pizz'app a permis de mettre en pratique de nombreuses technologies modernes du développement web, de l'architecture frontend-backend à la conteneurisation. L'expérience acquise est précieuse pour de futurs projets. Ce projet permettra d'améliorer l'expérience consommateur et producteur dans un cadre réel de pizzeria.