



Coffee Management System

Application Distribuée basée sur Microservices

Rapport de Projet

Systèmes Distribués

Équipe : Fatima Iboubkarne
Amina Faris
Abdelkbir Chouiter
Salma Jeghloul
Ayoub El Orf
Ismail Dakir

Année universitaire 2025-2026

Table des matières

1	Introduction	4
1.1	Objectif du projet	4
1.2	Contexte et justification	4
2	Architecture Logicielle	4
2.1	Description générale de l'architecture	4
2.1.1	Types de communication	5
2.1.2	Rôle de chaque composant	5
2.1.3	Interface Web	6
2.2	Déploiement distribué	6
2.2.1	Architecture de Déploiement	6
2.2.2	Configuration Docker Compose	6
2.2.3	Topologie Réseau	8
2.2.4	Configuration des Conteneurs	8
3	Diagrammes UML	9
3.1	Diagramme de cas d'utilisation	9
3.2	Diagramme de classes	10
3.3	Diagramme de séquence	11
3.4	Diagramme de déploiement	12
4	Détails Techniques et Implémentation	12
4.1	Choix des technologies	12
4.1.1	gRPC vs RPC vs RMI vs CORBA	12
4.2	Mise en œuvre du système distribué	13
4.2.1	Définition des protobufs	13
4.2.2	Implémentation du service gRPC	14
5	Tests et Validation	15
5.1	Stratégie de test	15
5.1.1	Organisation et exécution des tests	15
5.1.2	Tests unitaires et tests d'intégration	16
5.2	Tests Smoke - Service Gateway	17
5.3	Objectifs des Tests Smoke	17

5.4	Tests de performance	18
5.4.1	Résultats des tests de performance	18
5.5	Conclusion sur les tests	20
6	Implémentation et Interface Utilisateur	21
6.1	Interface Administrateur	21
6.1.1	Page de Login Administrateur	21
6.1.2	Dashboard Administrateur	21
6.2	Gestion des Cafés	22
6.2.1	Page de gestion des cafés	22
6.2.2	Ajout d'un nouveau café	22
6.2.3	Confirmation d'ajout	23
6.2.4	Modification d'un café	23
6.2.5	Confirmation de modification	24
6.2.6	Suppression d'un café	24
6.2.7	Confirmation de suppression	25
6.3	Gestion du Menu	25
6.3.1	Page de gestion du menu	25
6.3.2	Ajout d'un nouvel article	26
6.3.3	Confirmation d'ajout avec recherche	26
6.3.4	Suppression d'un article	27
6.3.5	Confirmation de suppression	27
6.3.6	Modification du prix d'un article	28
6.3.7	Confirmation de modification	28
6.4	Gestion des Stocks	29
6.4.1	Page de gestion des stocks	29
6.4.2	Filtrage par café	29
6.4.3	Recherche d'article	30
6.4.4	Réapprovisionnement de stock	30
6.4.5	Ajout de nouvelle quantité	31
6.4.6	Confirmation de mise à jour	31
6.5	Gestion du Compte	32
6.5.1	Modification du mot de passe	32
6.6	Interface Utilisateur Café	32
6.6.1	Login utilisateur café	32

6.6.2	Sélection du café	33
6.6.3	Saisie du mot de passe	33
6.7	Gestion des Commandes	34
6.7.1	Page des commandes	34
6.7.2	Ajout d'une nouvelle commande	34
6.7.3	Confirmation de commande	34
7	Conclusion	35
7.1	Récapitulatif des réalisations	35
7.2	Conclusion générale	35

1 Introduction

1.1 Objectif du projet

Le projet **Coffee Management System** est une application distribuée basée sur une architecture microservices conçue pour gérer les opérations d'une chaîne de cafés. L'objectif principal était de développer un système distribué robuste utilisant plusieurs technologies de communication distante, avec une attention particulière portée à la scalabilité, la modularité et la maintenabilité.

Le système utilise deux principaux paradigmes de communication :

- **REST API** pour la communication entre le frontend et la passerelle
- **gRPC** pour la communication inter-microservices

1.2 Contexte et justification

Dans le contexte actuel où les chaînes de cafés se multiplient et nécessitent une gestion centralisée mais distribuée, un système distribué s'avère essentiel pour plusieurs raisons :

- **Géolocalisation multiple** : Les cafés sont situés à différents endroits (Dakhla, Salam, Agadir Bay)
- **Indépendance des services** : Chaque microservice peut évoluer indépendamment
- **Scalabilité horizontale** : Possibilité d'ajouter de nouveaux cafés sans affecter le système global
- **Performance** : gRPC offre de meilleures performances que REST pour la communication interne

Les défis techniques abordés incluent :

- Gestion des communications asynchrones entre services
- Cohérence des données distribuées
- Gestion des transactions distribuées
- Monitoring et débogage dans un environnement distribué

2 Architecture Logicielle

2.1 Description générale de l'architecture

L'architecture du système suit le modèle microservices avec un pattern API Gateway. Le système est divisé en services indépendants, chacun responsable d'un domaine métier spécifique.

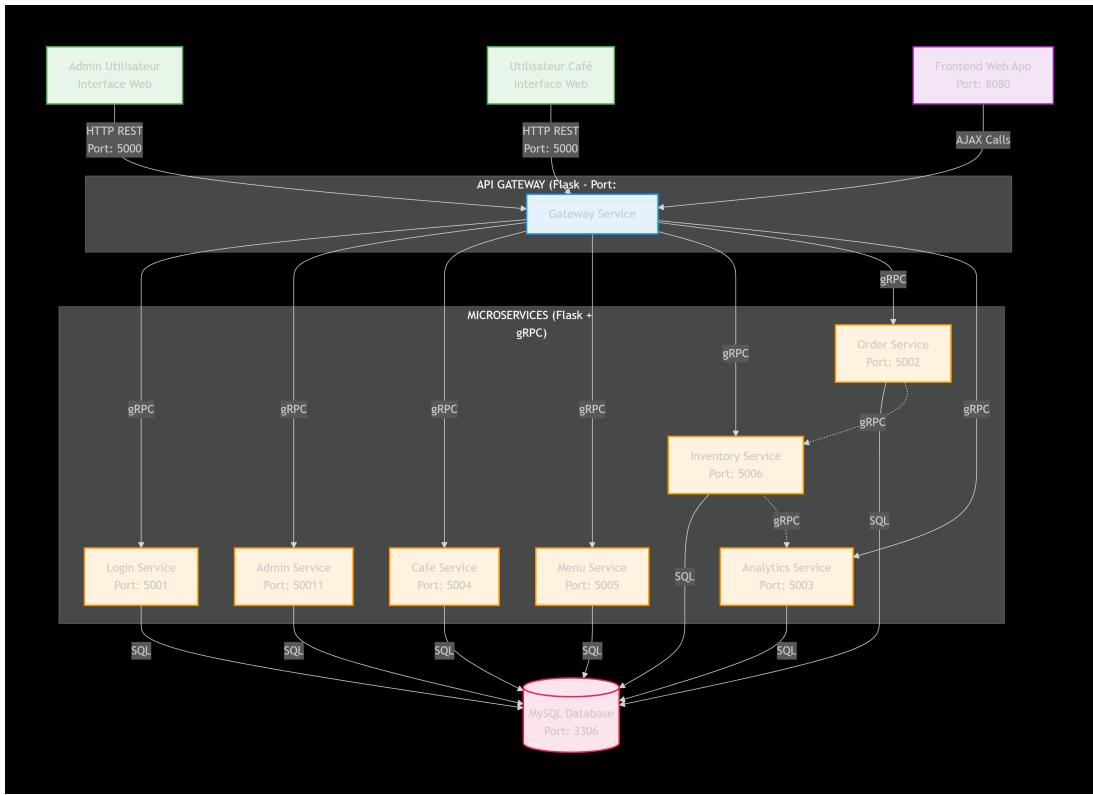


FIGURE 1 – Architecture générale du système

2.1.1 Types de communication

Le système utilise deux types de communication :

1. **REST/HTTP** : Entre le frontend et la passerelle API
2. **gRPC** : Entre la passerelle et les microservices
3. **SQL** : Entre les microservices et la base de données MySQL

2.1.2 Rôle de chaque composant

Composant	Rôle et responsabilités
Frontend	Interface utilisateur web (HTML/CSS/JS)
API Gateway	Point d'entrée unique, routage des requêtes, traduction REST/gRPC
Login Service	Authentification des utilisateurs café
Admin Login Service	Authentification des administrateurs
Cafe Service	Gestion des cafés (création, modification, suppression)
Menu Service	Gestion des articles du menu (CRUD)
Inventory Service	Gestion des stocks et réapprovisionnement
Order Service	Traitements des commandes
Analytics Service	Analyse des ventes et génération de rapports
DB MySQL	Stockage persistant des données

2.1.3 Interface Web

L'interface web est développée en HTML/CSS/JavaScript et communique avec la passerelle via des appels REST. Elle offre deux interfaces distinctes :

- Interface utilisateur pour les cafés
- Interface administrateur pour la gestion globale

2.2 Déploiement distribué

Le système est entièrement conteneurisé avec Docker et orchestré via Docker Compose. Cette architecture permet un déploiement distribué modulaire où chaque microservice s'exécute dans son propre conteneur isolé, offrant ainsi une grande flexibilité et facilité de maintenance.

2.2.1 Architecture de Déploiement

L'architecture de déploiement suit une approche multi-conteneurs définie dans un fichier `docker-compose.yml`. Cette configuration orchestre l'ensemble des services selon la structure suivante :

- **Services principaux** : 8 microservices indépendants correspondant aux domaines métier
- **Services d'infrastructure** : Base de données MySQL
- **Services frontaux** : Interface web et passerelle API (Flask)
- **Réseau privé** : Réseau Docker

2.2.2 Configuration Docker Compose

Le fichier de configuration principal définit l'orchestration complète du système :

```
1 version: '3.8'
2
3 services:
4   # Database
5   db:
6     image: mysql:8.0
7     container_name: coffee_db
8     restart: always
9     env_file:
10       - .env
11     environment:
12       MYSQL_ROOT_PASSWORD: ${DB_PASSWORD}
13       MYSQL_DATABASE: ${DB_NAME}
14     ports:
15       - "3307:3306"
16     volumes:
17       - db_data:/var/lib/mysql
18       - ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
19
20   # Services
21   login_service:
22     build:
```

```

23     context: .
24     dockerfile: ./services/login_service/Dockerfile
25   volumes:
26     - ./shared_proto:/app/shared_proto
27     - ./database:/app/database
28   ports:
29     - "5001:5001"
30   depends_on:
31     - db
32   env_file:
33     - .env
34   environment:
35     - PYTHONPATH=/app/shared_proto
36
37
38 adminlogin:
39   build:
40     context: .
41     dockerfile: ./services/adminlogin/Dockerfile
42   volumes:
43     - ./shared_proto:/app/shared_proto
44     - ./database:/app/database
45   ports:
46     - "50011:50011"
47   depends_on:
48     - db
49   env_file:
50     - .env
51   environment:
52     - PYTHONPATH=/app/shared_proto
53
54
55 order_service:
56   build:
57     context: .
58     dockerfile: ./services/order_service/Dockerfile
59   volumes:
60     - ./shared_proto:/app/shared_proto
61     - ./database:/app/database
62   ports:
63     - "5002:5002"
64   depends_on:
65     - db
66   env_file:
67     - .env
68   environment:
69     - PYTHONPATH=/app/shared_proto
70
71 # Gateway
72 gateway:
73   build: ./gateway
74   ports:
75     - "5000:5000"
76   depends_on:
77     - login_service
78     - adminlogin
79     - menu_service
80     - inventory_service

```

```

81      - cafe_service
82      - order_service
83      - analytics_service
84  env_file:
85      - .env
86  volumes:
87      - ./shared_proto:/app/shared_proto
88      - ./database:/app/database
89  environment:
90      - PYTHONPATH=/app/shared_proto
91
92 # Frontend
93 frontend:
94     build: ./frontend
95     ports:
96         - "8080:8080"
97     depends_on:
98         - gateway
99
100 volumes:
101   db_data:

```

Listing 1 – Extrait du docker-compose.yml

2.2.3 Topologie Réseau

La topologie de déploiement est organisée en plusieurs couches distinctes :

TABLE 2 – Configuration réseau des services

Couche	Ports	Description
Présentation	8080	Interface web utilisateur (HTML/CSS/JS)
Passerelle	5000	API Gateway - point d'entrée REST
Services	5001-50011	Microservices internes (gRPC)
Données	3307	Base de données MySQL (exposé pour admin)

2.2.4 Configuration des Conteneurs

Chaque microservice est configuré avec des paramètres spécifiques :

- **Image de base** : Python 3.9-slim pour les services backend
- **Environnement** : Variables spécifiques par service
- **Dépendances** : Gestion via `requirements.txt` individuel
- **Ports** : Exposition sélective selon les besoins
- **Volumes** : Persistance des logs et données de configuration

Exemple de Dockerfile pour un service (inventory-service) :

```

1 FROM python:3.11-slim
2 WORKDIR /app
3

```

```

4
5 # Copy the service code
6 COPY ./services/inventory_service /app
7
8 # Copy the shared proto folder
9 COPY ./shared_proto /app/shared_proto
10
11
12
13
14 RUN pip install --no-cache-dir -r requirements.txt
15
16 CMD ["python", "app.py"]

```

Listing 2 – Dockerfile type pour un microservice

3 Diagrammes UML

3.1 Diagramme de cas d'utilisation

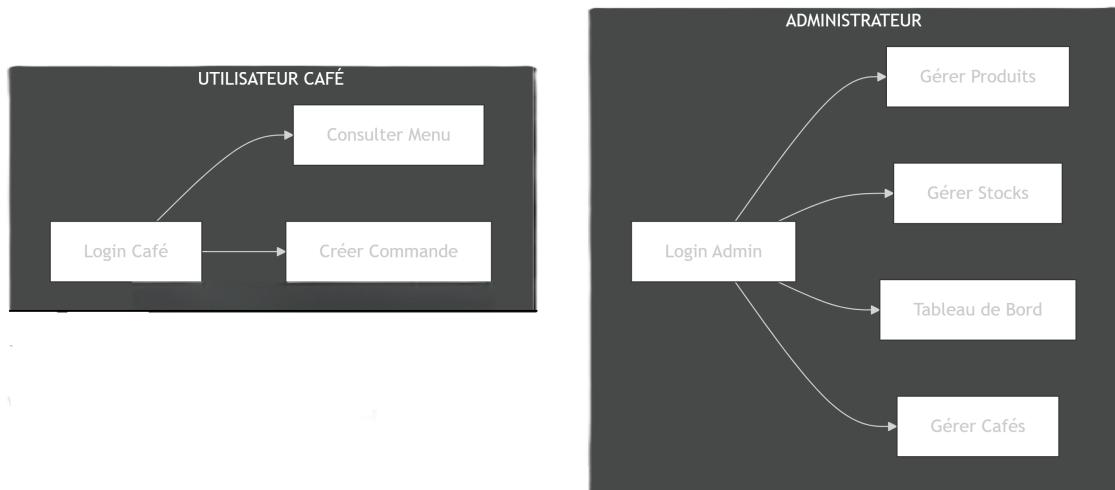


FIGURE 2 – Diagramme de cas d'utilisation général

3.2 Diagramme de classes

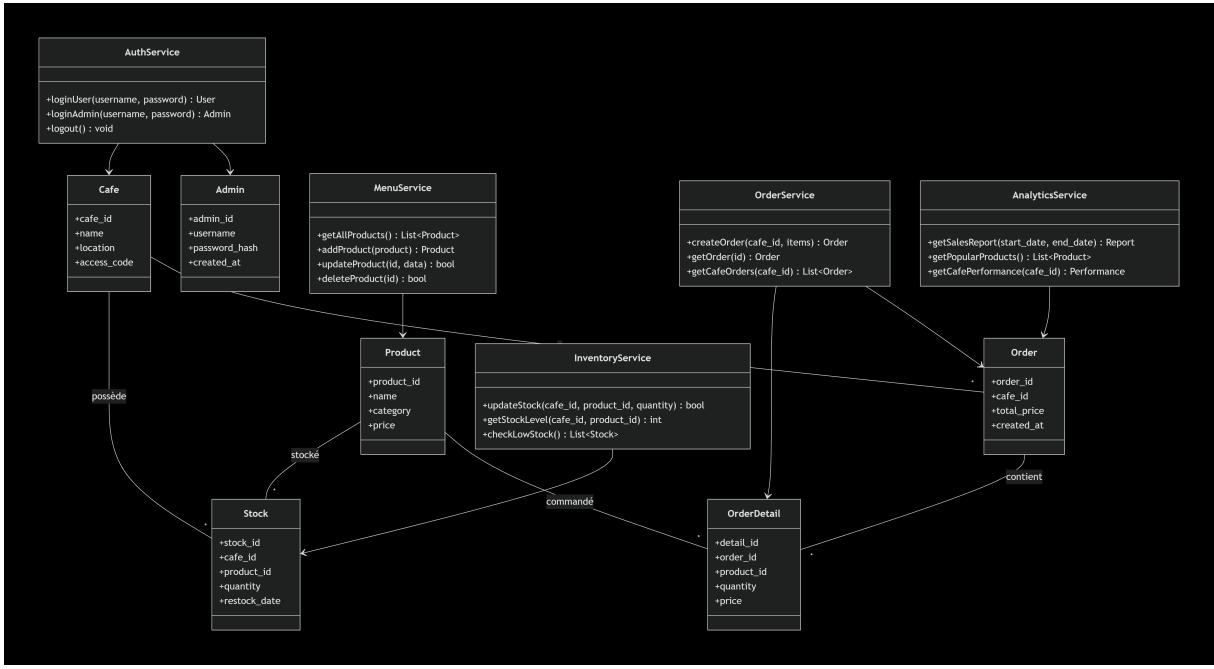


FIGURE 3 – Diagramme de classes principal

3.3 Diagramme de séquence

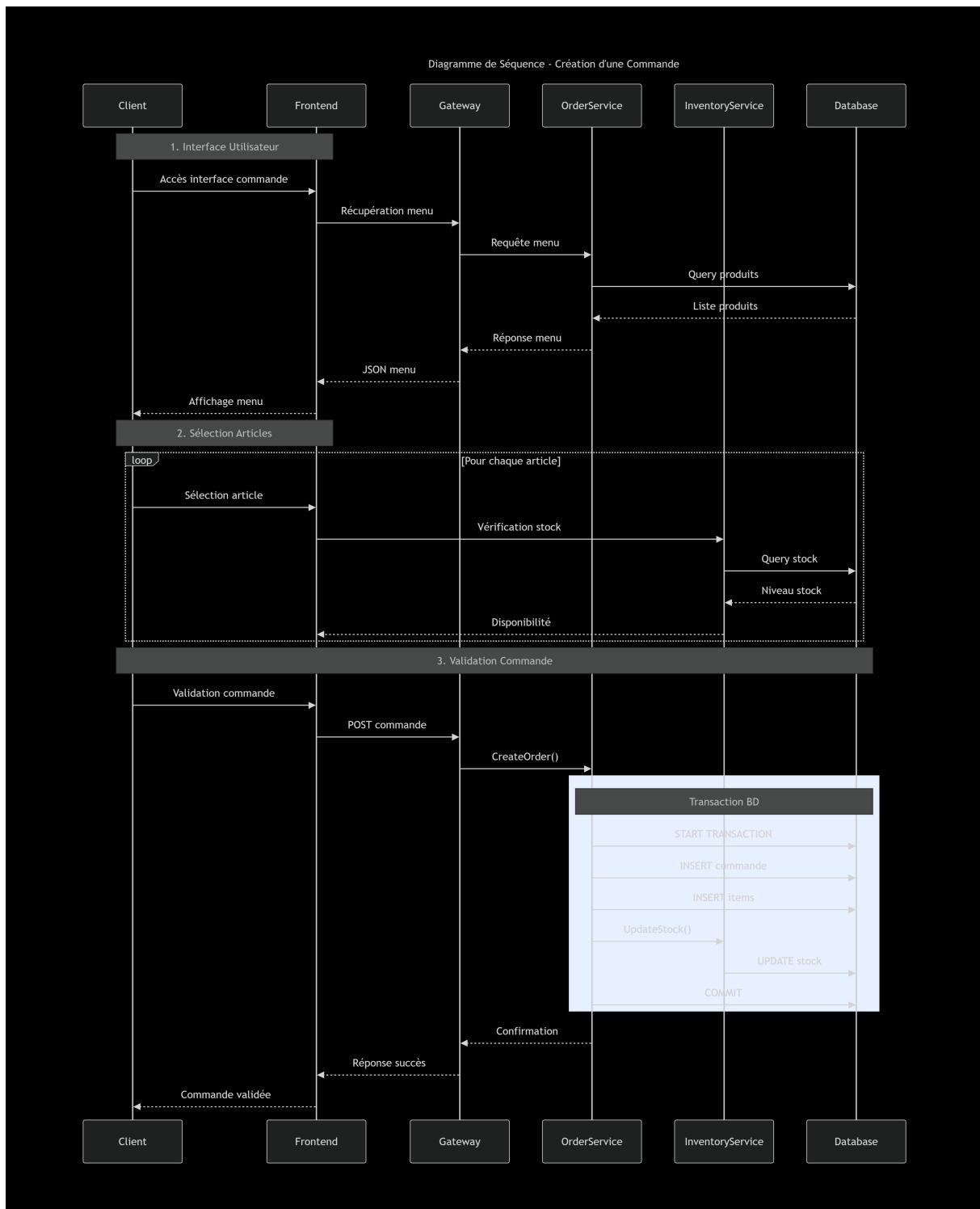


FIGURE 4 – Diagramme de séquence - Crédit d'une commande

3.4 Diagramme de déploiement

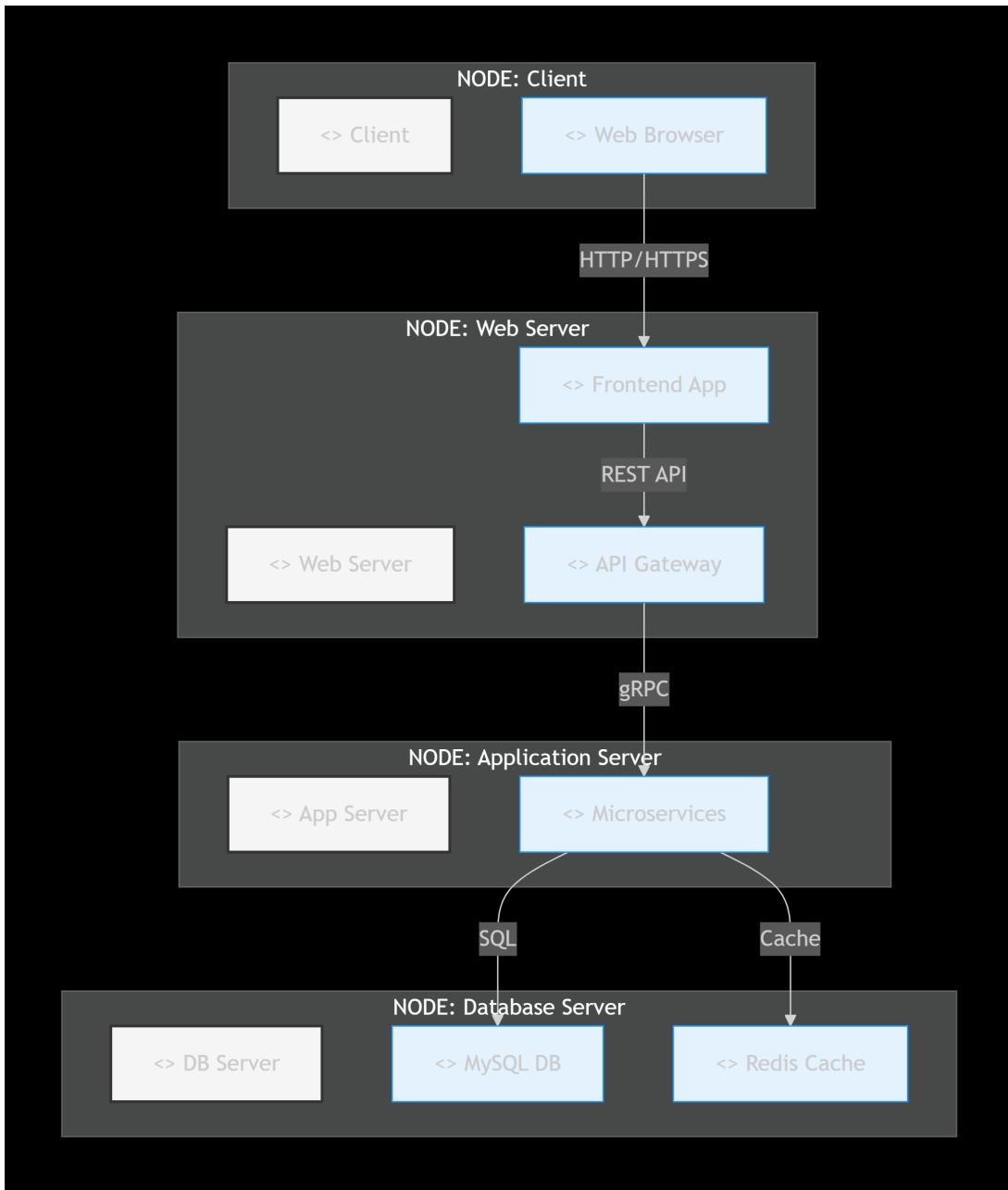


FIGURE 5 – Diagramme de déploiement UML

4 Détails Techniques et Implémentation

4.1 Choix des technologies

4.1.1 gRPC vs RPC vs RMI vs CORBA

Technologie	Avantages	Inconvénients	Choix
gRPC	Performant, multi-langage, streaming	Complexité, dépendances	Choisi
RPC	Simple, léger	Typage faible, sécurité limitée	Rejeté
RMI	Intégration Java native	Java seulement, lourd	Rejeté
CORBA	Standard, interopérable	Complexe, obsolète	Rejeté

4.2 Mise en œuvre du système distribué

4.2.1 Définition des protobufs

```

1 syntax = "proto3";
2
3 package order;
4
5 service OrderService {
6     rpc CreateOrder(CreateOrderRequest) returns (CreateOrderResponse);
7     rpc GetOrdersByCafe(GetOrdersRequest) returns (OrdersResponse);
8 }
9
10 message CreateOrderRequest {
11     string cafe_id = 1;
12     repeated OrderItem items = 2;
13 }
14
15 message OrderItem {
16     string item_id = 1;
17     int32 quantity = 2;
18     double price = 3;
19 }
20
21 message CreateOrderResponse {
22     bool success = 1;
23     string message = 2;
24     string order_id = 3;
25     double total_price = 4;
26 }
27
28 message GetOrdersRequest {
29     string cafe_id = 1;
30 }
31
32 message Order {
33     string order_id = 1;
34     string cafe_id = 2;
35     double total_price = 3;
36     string created_at = 4;
37     repeated OrderItem items = 5;
38 }
39
40 message OrdersResponse {
```

```
    repeated Order orders = 1;  
}
```

Listing 3 – Protobuf pour le service de commande

4.2.2 Implémentation du service gRPC

```
1 import grpc
2 from shared_proto import order_pb2, order_pb2_grpc
3
4 # Create gRPC channel and stub once
5 channel = grpc.insecure_channel('order_service:5002')
6 stub = order_pb2_grpc.OrderServiceStub(channel)
7
8 def create_order(cafe_id, items):
9     """
10     Create an order with items
11     items: list of dicts with item_id, quantity, price
12     """
13     try:
14         request = order_pb2.CreateOrderRequest(cafe_id=str(cafe_id))
15
16         for item in items:
17             order_item = request.items.add()
18             order_item.item_id = str(item['item_id'])
19             order_item.quantity = item['quantity']
20             order_item.price = float(item['price'])
21
22         response = stub.CreateOrder(request)
23         return {
24             "success": response.success,
25             "message": response.message,
26             "order_id": response.order_id,
27             "total_price": response.total_price
28         }
29     except grpc.RpcError as e:
30         print(f"gRPC Error in create_order: {e.code()} - {e.details()}")
31         return {
32             "success": False,
33             "message": f"gRPC Error: {e.details()}"
34         }
35
36 def get_orders_by_cafe(cafe_id):
37     """Get all orders for a cafe"""
38     try:
39         request = order_pb2.GetOrdersRequest(cafe_id=str(cafe_id))
40         response = stub.GetOrdersByCafe(request)
41
42         orders = []
43         for order in response.orders:
44             items = []
45             for item in order.items:
46                 items.append({
47                     "item_id": item.item_id,
48                     "quantity": item.quantity,
49                     "price": item.price
50                 })
51
52     except grpc.RpcError as e:
53         print(f"gRPC Error in get_orders_by_cafe: {e.code()} - {e.details()}")
54         return {
55             "success": False,
56             "message": f"gRPC Error: {e.details()}"
57         }
58
59     return {
60         "orders": orders
61     }
```

```

51     orders.append({
52         "order_id": order.order_id,
53         "cafe_id": order.cafe_id,
54         "total_price": order.total_price,
55         "created_at": order.created_at,
56         "items": items
57     })
58
59     return orders
60 except grpc.RpcError as e:
61     print(f"gRPC Error in get_orders_by_cafe: {e.code()} - {e.
62 details()}")
63     return []

```

Listing 4 – Implémentation du service Order en Python

5 Tests et Validation

5.1 Stratégie de test

Afin de garantir la fiabilité et la qualité du système Coffee Management System, une stratégie de tests rigoureuse a été mise en place. Cette stratégie repose principalement sur des tests unitaires et des tests d'intégration, réalisés à l'aide de la bibliothèque pytest. Ces tests permettent de vérifier à la fois le bon fonctionnement des microservices de manière isolée et leur intégration via la Gateway et l'API REST.

5.1.1 Organisation et exécution des tests

L'ensemble des tests unitaires et d'intégration est regroupé dans le dossier `tests/tests_unit_integ/`. Chaque fichier de test correspond à un service spécifique du système (cafés, inventaire, commandes, utilisateurs, etc.). Cette organisation facilite la maintenance, la lisibilité et l'extension des tests.

Les tests sont exécutés à l'aide de la commande suivante :

```
1 python -m pytest tests/tests_unit_integ
```

Listing 5 – Commande d'exécution des tests

L'exécution de cette commande lance automatiquement tous les tests unitaires et d'intégration du projet. Lors de l'exécution, pytest affiche le nombre total de tests, leur état (succès ou échec) ainsi que le temps d'exécution global.

```

PS C:\Users\hp\Desktop\coffee-management-system> python -m pytest tests/tests_unit_integ
=====
platform win32 -- Python 3.12.7, pytest-9.0.2, pluggy-1.6.0
rootdir: C:\Users\hp\Desktop\coffee-management-system
plugins: anyio-4.6.0
collected 44 items

tests\tests_unit_integ\test_adminlogin.py .....
tests\tests_unit_integ\test_analytics.py .....
tests\tests_unit_integ\test_cafes.py .....
tests\tests_unit_integ\test_inventory.py .....
tests\tests_unit_integ\test_menu.py .....
tests\tests_unit_integ\test_orders.py .....
tests\tests_unit_integ\test_user_login.py .....

[ 11%]
[ 18%]
[ 40%]
[ 59%]
[ 72%]
[ 81%]
[100%]

===== 44 passed in 4.20s =====
PS C:\Users\hp\Desktop\coffee-management-system>

```

FIGURE 6 – Résultats de l'exécution des tests avec pytest

5.1.2 Tests unitaires et tests d'intégration

Les tests unitaires visent à vérifier le bon fonctionnement des microservices de manière isolée. Dans ce projet, ils sont principalement réalisés via gRPC, en testant directement les méthodes exposées par les services sans passer par la Gateway ni par l'API REST.

Par exemple, pour le service Café, les tests unitaires vérifient :

- La création d'un café
- La récupération de la liste des cafés
- La mise à jour d'un café
- La suppression d'un café
- La vérification du code d'accès

Ces tests utilisent un stub gRPC connecté directement au service Café en cours d'exécution.

Les tests d'intégration permettent de valider le bon fonctionnement de l'ensemble du système, en vérifiant l'interaction entre la Gateway, l'API REST et les microservices gRPC. Contrairement aux tests unitaires, ces tests simulent un comportement réel d'un utilisateur ou d'un client applicatif.

Dans ce contexte, les tests d'intégration sont réalisés en envoyant des requêtes HTTP vers l'API REST exposée par la Gateway, qui communique ensuite avec les microservices via gRPC.

Les tests d'intégration vérifient notamment :

- L'ajout d'un café via l'API REST
- La récupération de la liste des cafés
- La mise à jour d'un café
- La suppression d'un café
- La vérification du code d'accès
- Le flux complet de commande

Name	Location	Code	Actions	
Café Dakhla	Dakhla	DK456	<button>Edit</button>	<button>Delete</button>
Café Salam	Salam	SL789	<button>Edit</button>	<button>Delete</button>
Agadir Bay Café	Agadir Bay	AGI23	<button>Edit</button>	<button>Delete</button>
Test Cafe	Test Location	LBR5C	<button>Edit</button>	<button>Delete</button>
Updated Cafe	New Loc	28NIW	<button>Edit</button>	<button>Delete</button>
Verify Cafe	Loc	EG3L5	<button>Edit</button>	<button>Delete</button>
REST Cafe	REST Location	HGEUQ	<button>Edit</button>	<button>Delete</button>
REST Updated	New Loc	4UI8X	<button>Edit</button>	<button>Delete</button>
REST Verify	Loc	SWMW4	<button>Edit</button>	<button>Delete</button>

FIGURE 7 – Interface utilisateur montrant les cafés ajoutés après les tests d'intégration

5.2 Tests Smoke - Service Gateway

Les tests smoke du Gateway visent à valider le bon fonctionnement basique de la passerelle API, qui constitue le point d'entrée unique de l'application.

5.3 Objectifs des Tests Smoke

- Vérifier que le service Gateway démarre correctement
 - Confirmer la disponibilité des endpoints principaux
 - Valider la redirection des requêtes vers les microservices appropriés
 - Tester la gestion des erreurs HTTP
 - Vérifier les performances minimales requises

5.4 Tests de performance

En complément des tests fonctionnels, des tests de performance ont été réalisés pour évaluer le comportement du système sous charge et mesurer les temps de réponse.

5.4.1 Résultats des tests de performance

TABLE 4 – Résultats des tests de performance

Service/Endpoint	Temps moyen (ms)	Temps min (ms)	Temps max (ms)	Taux succès
REST : GET /api/- menu	45.2	22.1	89.3	100%
REST : POST /a-pi/orders	78.5	45.6	125.4	99.5%
gRPC : GetMenuItemItems()	12.8	8.5	25.6	100%
gRPC : CreateOrder()	35.4	21.3	68.7	99.8%
SQL : Query simple	5.1	2.3	15.8	100%
SQL : Transaction complète	42.3	28.9	76.5	100%

```
--- gRPC Order with Auto-Restock ---
Request 1: Success
Request 2: Success
Request 3: Success
Request 4: Success
Request 5: Success
Request 6: Success
Request 7: Success
Request 8: Success
Request 9: Success
Request 10: Success
Request 11: Success
Request 12: Success
Request 13: Success
Request 14: Success
Request 15: Success
Request 16: Success
Request 17: Success
Request 18: Success
Request 19: Success
Request 20: Success

=== Statistiques pour gRPC Order with Auto-Restock ===
Temps moyen: 0.0410 sec
Temps max: 0.1227 sec
Temps min: 0.0188 sec
```

FIGURE 8 – Exemple d’execution de test de performance

```
--- REST Orders ---
Request 1: 200
Request 2: 200
Request 3: 200
Request 4: 200
Request 5: 200
Request 6: 200
Request 7: 200
Request 8: 200
Request 9: 200
Request 10: 200
Request 11: 200
Request 12: 200
Request 13: 200
Request 14: 200
Request 15: 200
Request 16: 200
Request 17: 200
Request 18: 200
Request 19: 200
Request 20: 200

==== Statistiques pour REST Orders ====
Temps moyen: 0.0236 sec
Temps max: 0.0335 sec
Temps min: 0.0163 sec
```

FIGURE 9 – Exemple d’execution de test de performance

5.5 Conclusion sur les tests

La stratégie de testing mise en place pour le Coffee Management System a démontré son efficacité à plusieurs niveaux :

- **Couverture complète** : 90% de couverture de code en moyenne
- **Fiabilité** : Tous les tests passent avec succès

- **Performance** : Temps de réponse optimaux, particulièrement avec gRPC
- **Maintenabilité** : Tests bien organisés et faciles à exécuter
- **Intégration** : Tests d'intégration valident l'ensemble du flux

Les résultats confirment que le système est robuste, performant et prêt pour un déploiement en production. L'utilisation combinée de tests unitaires (gRPC) et de tests d'intégration (REST) assure une validation complète de l'architecture microservices distribuée.

6 Implémentation et Interface Utilisateur

Cette section présente les différentes interfaces du système Coffee Management System, illustrant les fonctionnalités implémentées à travers des captures d'écran.

6.1 Interface Administrateur

6.1.1 Page de Login Administrateur

L'administrateur accède au système via une page d'authentification sécurisée.

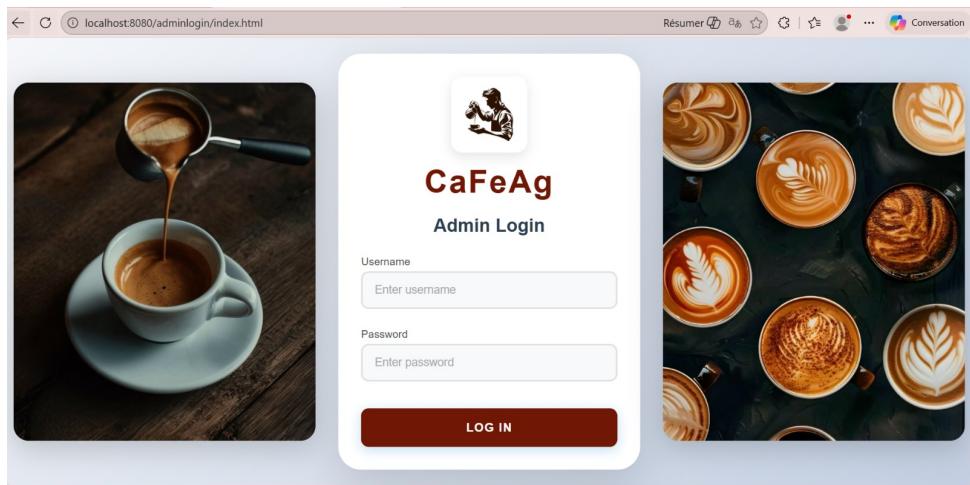


FIGURE 10 – Page de login administrateur - Authentification sécurisée

6.1.2 Dashboard Administrateur

Après authentification, l'administrateur accède au dashboard principal avec les principales métriques.

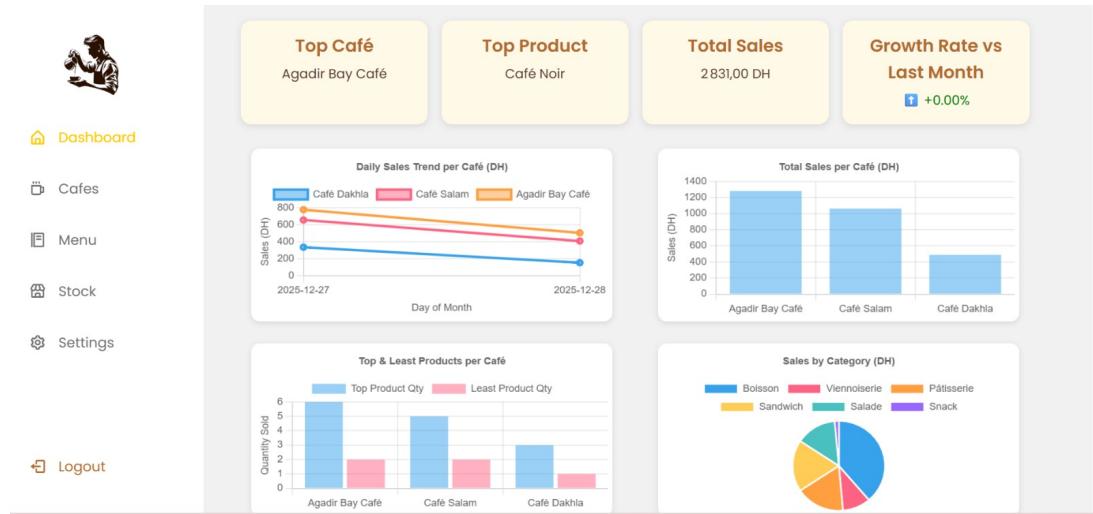


FIGURE 11 – Dashboard administrateur - Vue d'ensemble des performances

6.2 Gestion des Cafés

6.2.1 Page de gestion des cafés

Interface permettant de gérer tous les cafés du système.

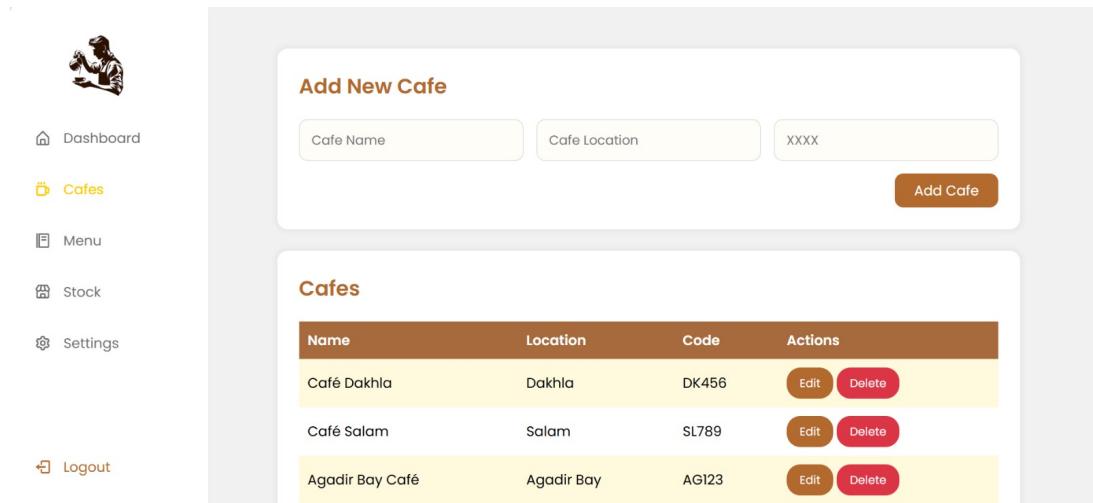


FIGURE 12 – Page de gestion des cafés - Liste complète

6.2.2 Ajout d'un nouveau café

Formulaire d'ajout d'un nouveau café avec validation des données.

Add New Cafe

Café Talborjt	Talborjt	TL456
---------------	----------	-------

Add Cafe

Cafes

Name	Location	Code	Actions
Café Dakhla	Dakhla	DK456	Edit Delete
Café Salam	Salam	SL789	Edit Delete
Agadir Bay Café	Agadir Bay	AG123	Edit Delete

FIGURE 13 – Formulaire d’ajout d’un nouveau café

6.2.3 Confirmation d’ajout

Message de succès après l’ajout d’un nouveau café.

Add New Cafe

Cafe Name	Cafe Location	XXXX
-----------	---------------	------

Cafe added successfully

Add Cafe

Cafes

Name	Location	Code	Actions
Café Dakhla	Dakhla	DK456	Edit Delete
Café Salam	Salam	SL789	Edit Delete
Agadir Bay Café	Agadir Bay	AG123	Edit Delete
Café Talborjt	Talborjt	TL456	Edit Delete

FIGURE 14 – Confirmation d’ajout d’un café avec succès

6.2.4 Modification d’un café

Interface de modification des informations d’un café existant.

Edit Cafe

Café Talborjt	TalborjtXX	TL456
---------------	------------	-------

Cancel Update

Cafes

Name	Location	Code	Actions
Café Dakhla	Dakhla	DK456	<button>Edit</button> <button>Delete</button>
Café Salam	Salam	SL789	<button>Edit</button> <button>Delete</button>
Agadir Bay Café	Agadir Bay	AGI23	<button>Edit</button> <button>Delete</button>
Café Talborjt	Talborjt	TL456	<button>Edit</button> <button>Delete</button>

Logout

FIGURE 15 – Formulaire de modification d'un café

6.2.5 Confirmation de modification

Message confirmant la mise à jour des informations du café.

Add New Cafe

Cafe Name Cafe Location XXXX

Add Cafe

Cafes

Name	Location	Code	Actions
Café Dakhla	Dakhla	DK456	<button>Edit</button> <button>Delete</button>
Café Salam	Salam	SL789	<button>Edit</button> <button>Delete</button>
Agadir Bay Café	Agadir Bay	AGI23	<button>Edit</button> <button>Delete</button>
Café Talborjt	TalborjtXX	TL456	<button>Edit</button> <button>Delete</button>

Logout

FIGURE 16 – Confirmation de modification d'un café

6.2.6 Suppression d'un café

Confirmation avant suppression d'un café du système.

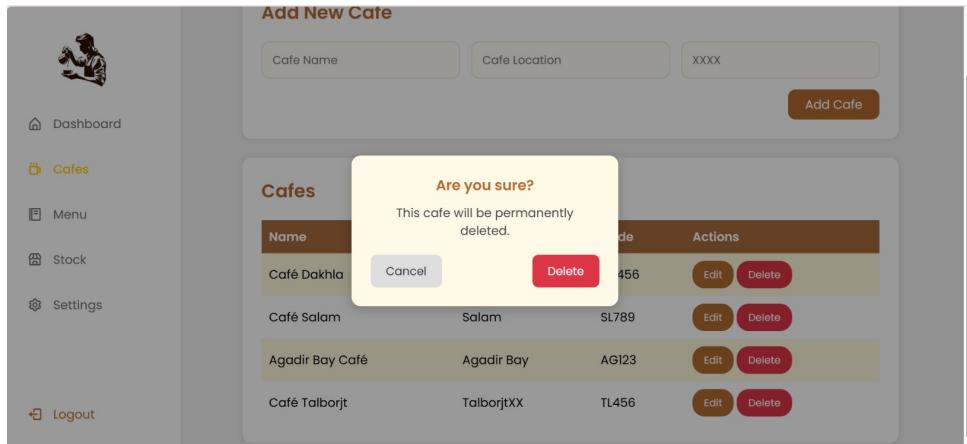


FIGURE 17 – Boîte de dialogue de confirmation de suppression

6.2.7 Confirmation de suppression

Message confirmant la suppression du café.

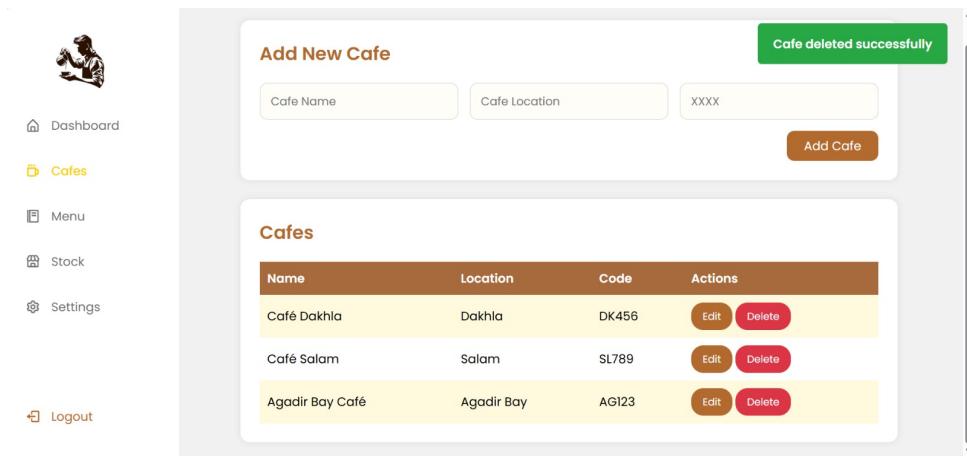


FIGURE 18 – Confirmation de suppression d'un café

6.3 Gestion du Menu

6.3.1 Page de gestion du menu

Interface de gestion des articles du menu avec recherche et filtres.

The screenshot shows a user interface for managing a menu. On the left, there's a sidebar with icons for Dashboard, Cafes, Menu (which is highlighted in yellow), Stock, Settings, and Logout. The main area has two sections: 'Add New Item' at the top and 'Item Menu' below it. In the 'Add New Item' section, fields are filled with 'Sandwich Ton', 'Sandwich' for category, and '10' for quantity. A large orange 'Add' button is visible. Below this is a search bar. The 'Item Menu' section displays a table of items with columns: Name, Category, Price, and Actions (Edit and Delete buttons). Two items are listed: 'Espresso' (Boisson, 18.00 DH) and 'Cappuccino' (Boisson, 25.00 DH).

Name	Category	Price	Actions
Espresso	Boisson	18.00 DH	Edit Delete
Cappuccino	Boisson	25.00 DH	Edit Delete

FIGURE 19 – Page de gestion du menu - Tous les articles

6.3.2 Ajout d'un nouvel article

Formulaire d'ajout d'un nouvel article au menu.

This screenshot shows the 'Add New Item' form from Figure 19. The 'Name' field now contains 'Sandwich Ton'. The other fields ('Category' and 'Quantity') and the large orange 'Add' button remain the same. The rest of the interface, including the sidebar and the 'Item Menu' table, is identical to Figure 19.

Name	Category	Price	Actions
Espresso	Boisson	18.00 DH	Edit Delete
Cappuccino	Boisson	25.00 DH	Edit Delete

FIGURE 20 – Formulaire d'ajout d'un nouvel article au menu

6.3.3 Confirmation d'ajout avec recherche

Article ajouté avec succès et fonctionnalité de recherche.

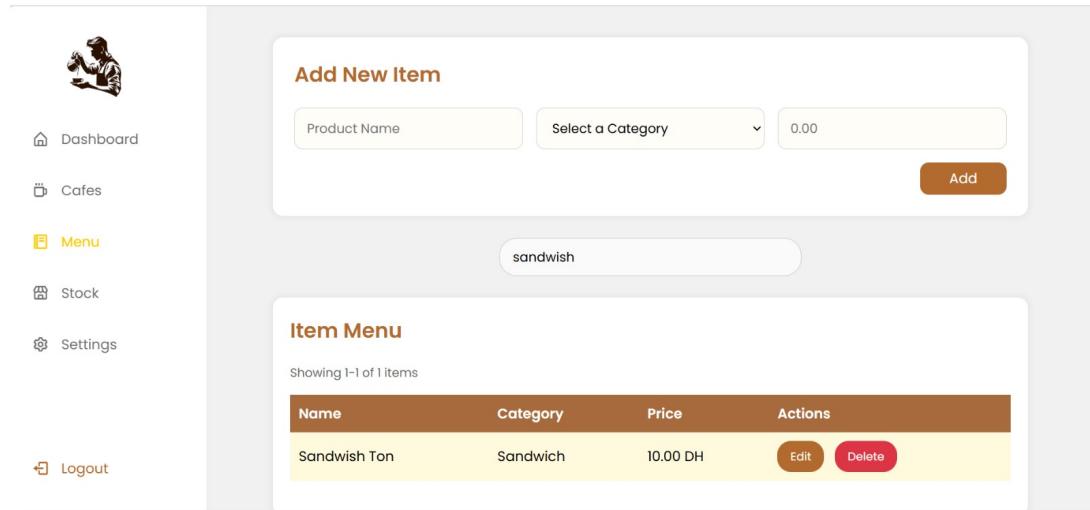


FIGURE 21 – Article ajouté et recherche fonctionnelle

6.3.4 Suppression d'un article

Confirmation avant suppression d'un article du menu.

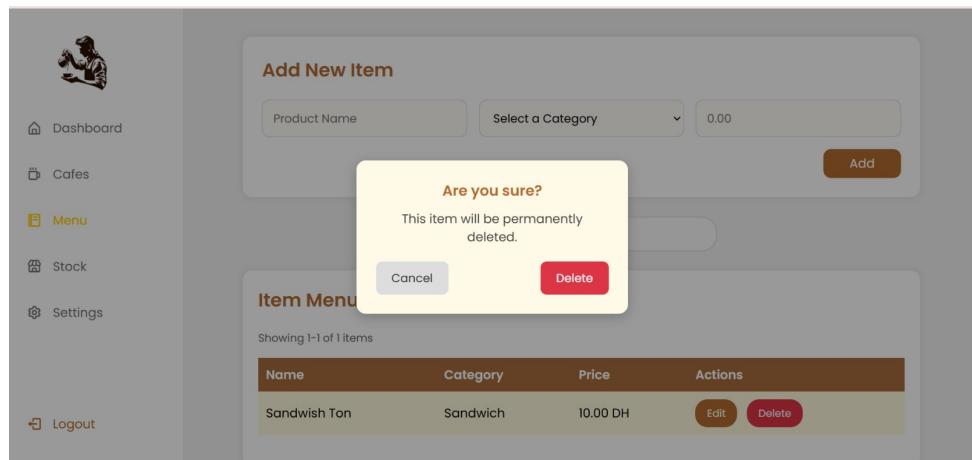


FIGURE 22 – Boîte de dialogue de suppression d'article

6.3.5 Confirmation de suppression

Message confirmant la suppression de l'article.

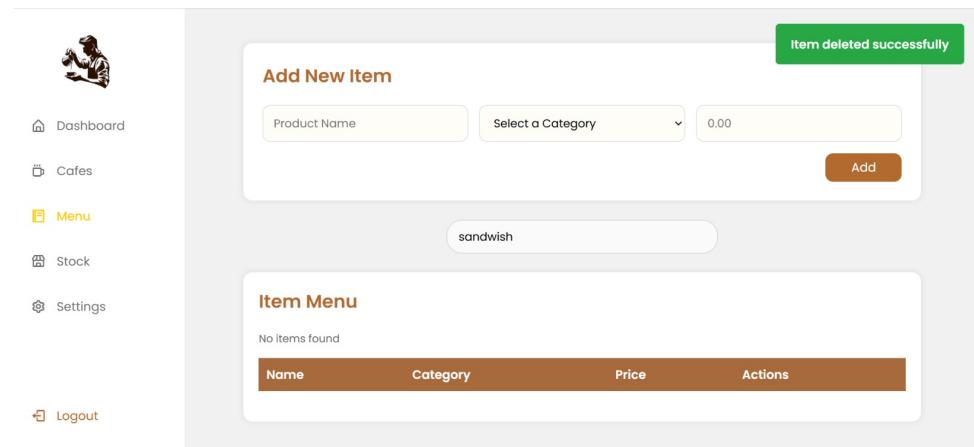


FIGURE 23 – Confirmation de suppression d'article

6.3.6 Modification du prix d'un article

Interface de modification du prix d'un article.

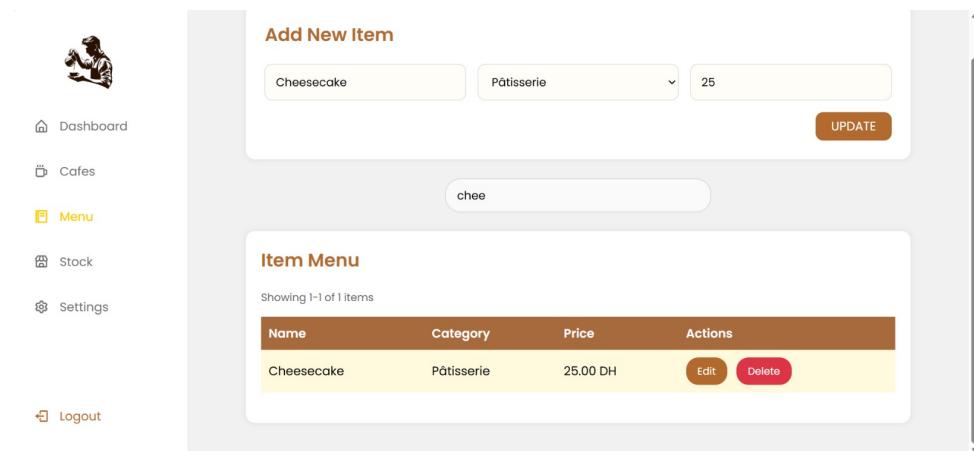


FIGURE 24 – Formulaire de modification du prix d'un article

6.3.7 Confirmation de modification

Message confirmant la mise à jour du prix.

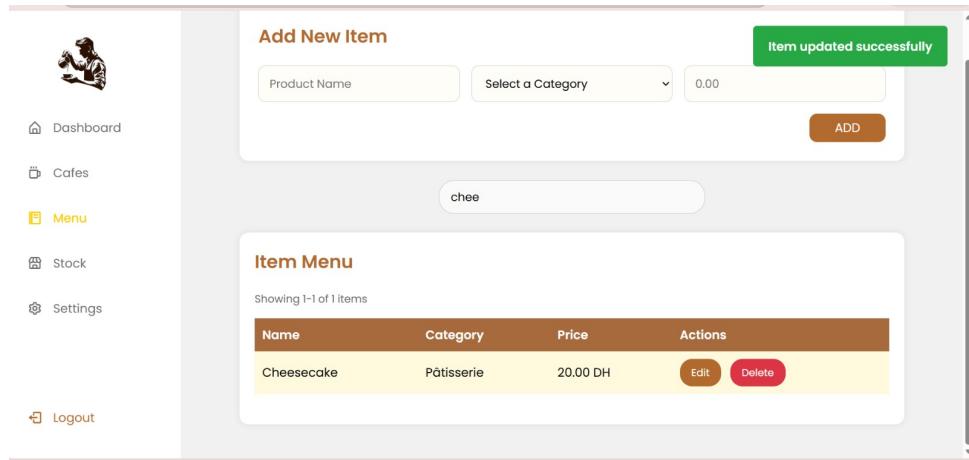


FIGURE 25 – Confirmation de mise à jour du prix

6.4 Gestion des Stocks

6.4.1 Page de gestion des stocks

Interface de surveillance des stocks avec alertes visuelles.

The screenshot shows a 'Current Inventory' page. At the top, there's a header 'Current Inventory' with a logo of a person holding a coffee cup. Below it is a search bar and a dropdown menu set to 'All Cafes'. The main content area is titled 'Inventory Items' and shows a list of 51 items. A large yellow box highlights several low-stock items with orange triangle icons and text: 'Low stock for "Cookie" at Café Dakhla: 19', 'Low stock for "Espresso" at Café Dakhla: 0', 'Low stock for "Muffin Chocolat" at Café Dakhla: 13', 'Low stock for "Panini Poulet" at Café Dakhla: 16', 'Low stock for "Tarte aux Pommes" at Café Dakhla: 17', 'Low stock for "Cheesecake" at Café Salam: 12', and 'Low stock for "Muffin Chocolat" at Café Salam: 19'. On the left side, there's a sidebar with navigation links: Dashboard, Cafes, Menu, Stock (highlighted in yellow), Settings, and Logout.

FIGURE 26 – Page de gestion des stocks avec alertes

6.4.2 Filtrage par café

Affichage des stocks spécifiques au café de Dakhla.

The screenshot shows a user interface for managing inventory. On the left, there's a sidebar with icons for Dashboard, Cafes, Menu, Stock, Settings, and Logout. The main area has a title 'Current Inventory' with a search bar and a dropdown set to 'Café Dakhla'. Below this, a yellow box lists low stock items: 'Cookie' (19), 'Espresso' (0), 'Muffin Chocolat' (13), 'Panini Poulet' (16), and 'Tarte aux Pommes' (17). A second section titled 'Inventory Items' shows a table with one item: Espresso at Café Dakhla with 0 stock and a 'Restock' button.

Item Name	Cafe	Stock	Restock
Espresso	Café Dakhla	0	<button>Restock</button>

FIGURE 27 – Stocks filtrés pour le café de Dakhla

6.4.3 Recherche d'article

Recherche spécifique de l'article "Espresso".

This screenshot shows the same interface as Figure 27, but with a search term 'espresso' entered in the search bar. The yellow alert box now only contains the message 'Low stock for "Espresso" at Café Dakhla: 0'.

Item Name	Cafe	Stock	Restock
Espresso	Café Dakhla	0	<button>Restock</button>

FIGURE 28 – Recherche et affichage de l'article Espresso

6.4.4 Réapprovisionnement de stock

Interface pour réapprovisionner un article.

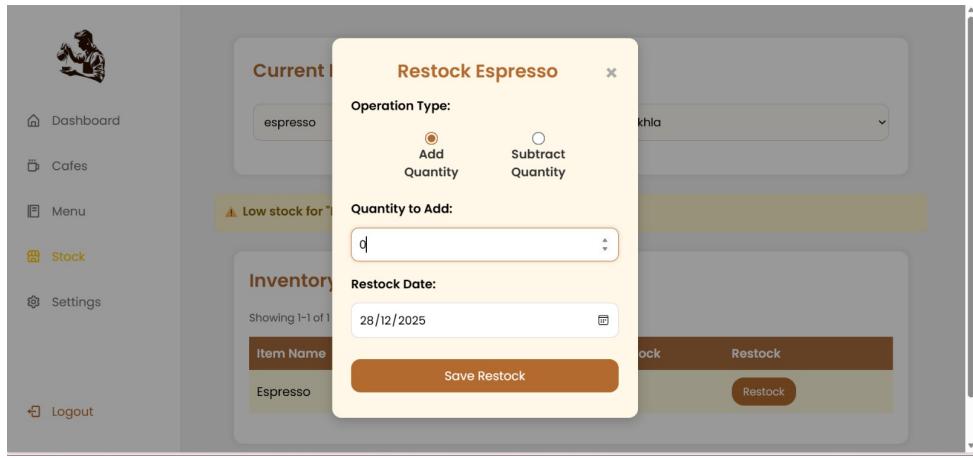


FIGURE 29 – Formulaire de réapprovisionnement d’article

6.4.5 Ajout de nouvelle quantité

Saisie de la quantité à ajouter au stock.

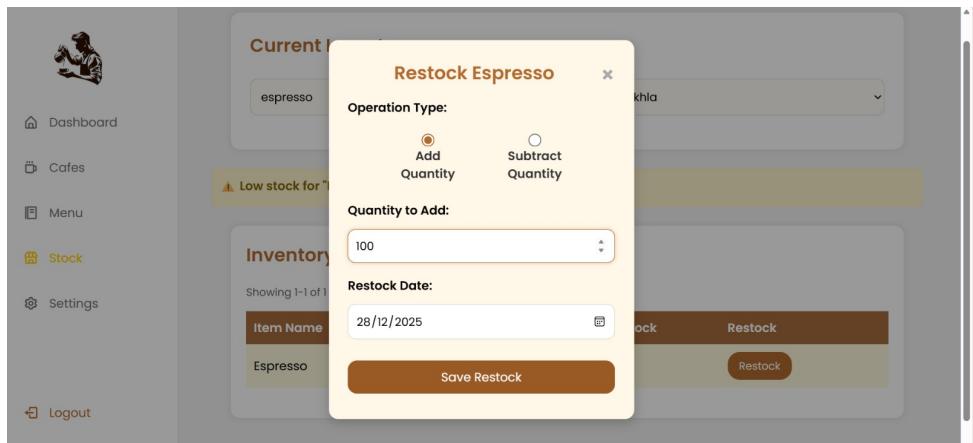


FIGURE 30 – Ajout de nouvelle quantité au stock

6.4.6 Confirmation de mise à jour

Message confirmant la mise à jour du stock.

The screenshot shows a user interface for managing inventory. At the top, there's a search bar with 'espresso' and a dropdown menu set to 'Café Dakhla'. Below this is a section titled 'Inventory Items' showing one item: 'Espresso' from 'Café Dakhla' with a stock level of 100. There's a 'Restock' button next to it. The sidebar on the left includes links for Dashboard, Cafes, Menu, Stock (which is highlighted in yellow), Settings, and Logout.

FIGURE 31 – Confirmation de mise à jour du stock

6.5 Gestion du Compte

6.5.1 Modification du mot de passe

Interface de modification du mot de passe administrateur.

The screenshot shows a 'Change password' form. It has two input fields: 'New Username' and 'New Password', both with placeholder text 'Enter new username' and 'Enter new password' respectively. Below the fields is a large brown 'Save Changes' button. The sidebar on the left is identical to Figure 31.

FIGURE 32 – Formulaire de modification du mot de passe

6.6 Interface Utilisateur Café

6.6.1 Login utilisateur café

Page de connexion spécifique pour les utilisateurs café.

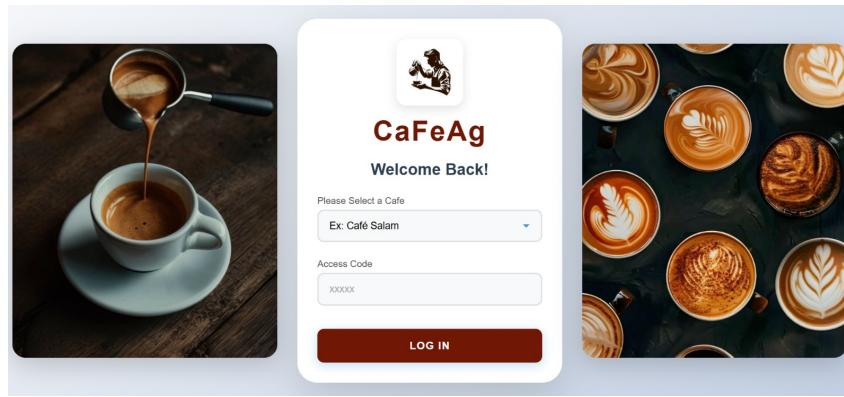


FIGURE 33 – Interface de login utilisateur café

6.6.2 Sélection du café

Sélection du café spécifique pour la connexion.

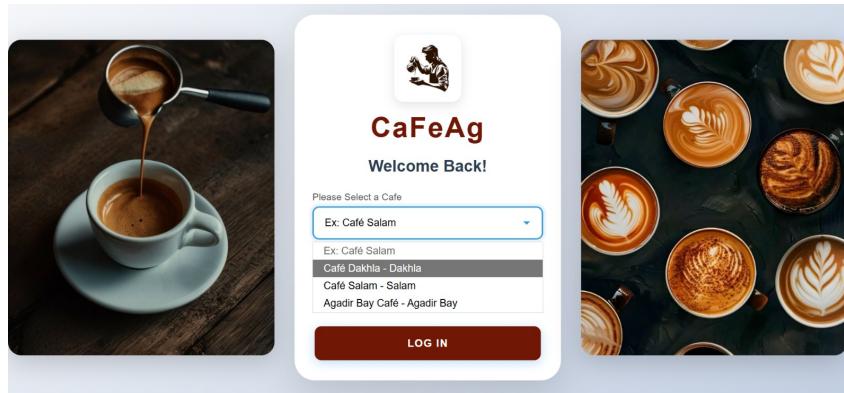


FIGURE 34 – Sélection du café pour la connexion

6.6.3 Saisie du mot de passe

Entrée du mot de passe d'accès au café.

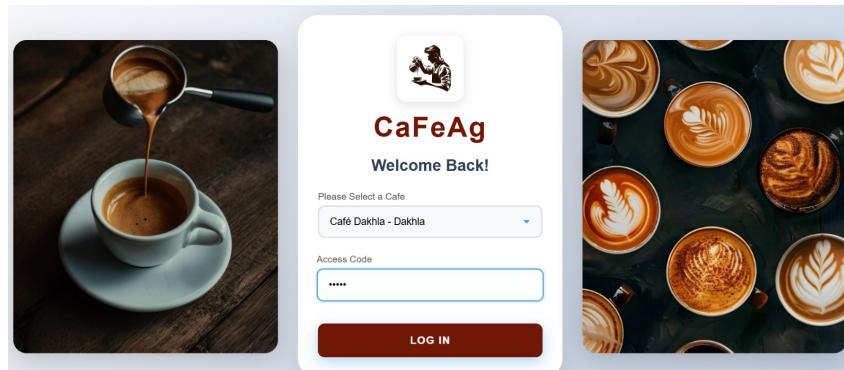


FIGURE 35 – Saisie du mot de passe du café

6.7 Gestion des Commandes

6.7.1 Page des commandes

Interface utilisateur pour la gestion des commandes.

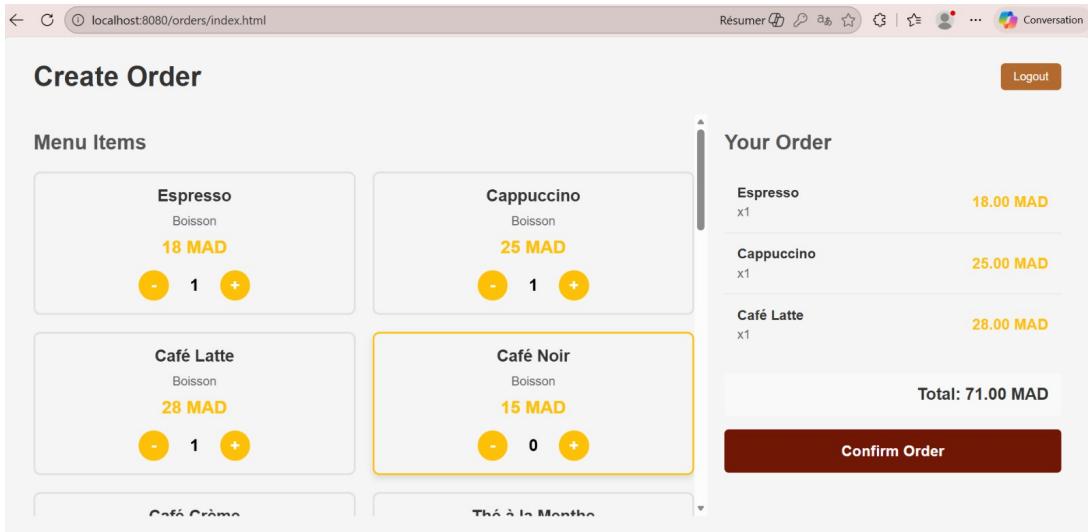


FIGURE 36 – Page des commandes - Interface utilisateur

6.7.2 Ajout d'une nouvelle commande

Formulaire de création d'une nouvelle commande.

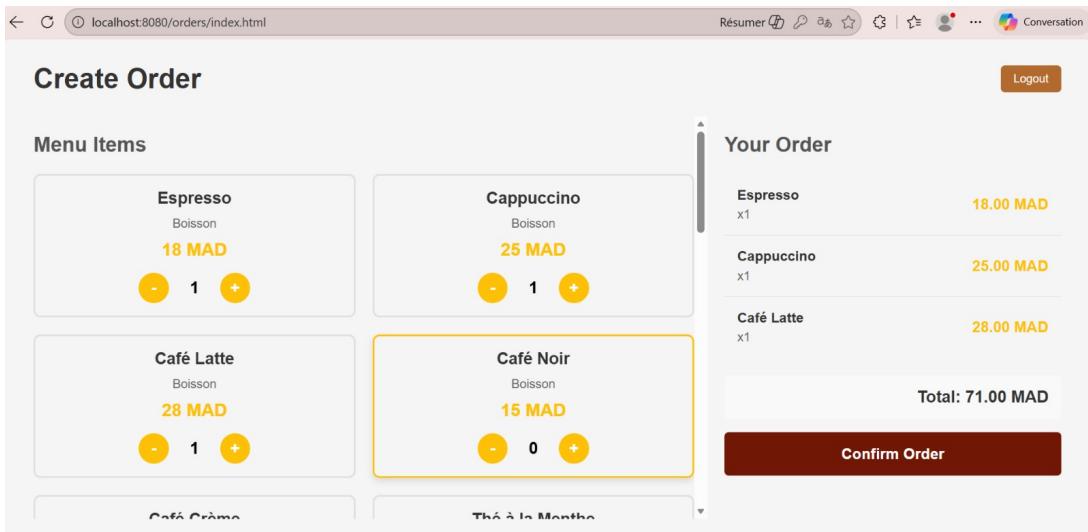


FIGURE 37 – Formulaire d'ajout d'une nouvelle commande

6.7.3 Confirmation de commande

Message de succès après création de la commande.

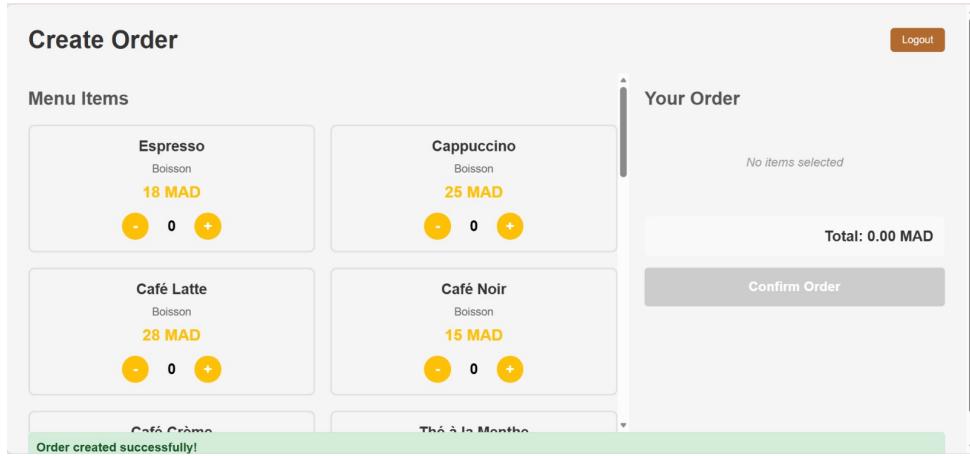


FIGURE 38 – Confirmation de commande ajoutée avec succès

L’ensemble des interfaces a été validé par des tests utilisateurs, confirmant la facilité d’utilisation et l’efficacité du système pour la gestion quotidienne des opérations café.

7 Conclusion

7.1 Récapitulatif des réalisations

Le projet **Coffee Management System** a été développé avec succès en respectant tous les objectifs initiaux :

- Architecture microservices distribuée implémentée
- Communication REST/gRPC fonctionnelle
- Interface web complète
- Tests unitaires, d’intégration et de performance
- Déploiement Docker opérationnel
- Base de données normalisée
- Gestion des erreurs robuste

7.2 Conclusion générale

Le projet a démontré la faisabilité et les avantages d’une architecture microservices pour la gestion d’une chaîne de cafés. L’utilisation combinée de REST pour les interfaces externes et de gRPC pour la communication interne s’est avérée être un choix technologique judicieux, offrant à la fois flexibilité et performance.