API Demandes de Transport - Service 3

API REST Spring Boot permettant de :

- Créer des demandes de transport avec calcul automatique du devis
- Valider les demandes par les clients
- Gérer le cycle de vie des demandes (création → validation → affectation)
- Suivre l'historique des demandes par client

Base URL : `/api/v1/demandes`

> **Note importante** : Ce service s'intègre avec les Services 4 (Itinéraires), 5 (Tarification) et 8 (Matching) pour offrir une expérience complète.

Sommaire

Quick Start

1. Lancer la base de données

```
docker-compose up -d postgres-demandes
```

2. Lancer l'application

```
./mvnw spring-boot:run
```

3. Vérifier que ça fonctionne

```
curl http://localhost:8083/actuator/health
```

4. Importer la collection Postman

Importez `Service3_Demandes_Transport.postman_collection.json` dans Postman - le token JWT est déjà configuré !

Endpoints

Demandes de Transport

Méthode	Endpoint	Description	Authentification
POST	`/api/v1/demandes`	Créer une nouvelle demande de transport	JWT requis
GET	`/api/v1/demandes`	Lister toutes mes demandes	JWT requis
GET	`/api/v1/demandes/{id}`	Récupérer une demande par ID	JWT requis
PUT	`/api/v1/demandes/{id}/validation`	Valider une demande (accepter le devis)	JWT requis

Catégories de Marchandise

Méthode	Endpoint	Description	Authentification
GET	`/api/v1/categories`	Lister toutes les catégories	Non (public)
GET	`/api/v1/categories/{id}`	Récupérer une catégorie par ID (UUID)	Non (public)
GET	`/api/v1/categories/nom/{nom}`	Rechercher une catégorie par nom exact	Non (public)
GET	`/api/v1/categories/search?keyword=`	Rechercher par mot-clé	Non (public)
GET	`/api/v1/categories/filter/fragile?=`	Filtrer par fragilité	Non (public)
GET	`/api/v1/categories/filter/dangereux?=`	Filtrer par dangerosité	Non (public)
GET	`/api/v1/categories/filter/temperature?=`	Filtrer par température requise	Non (public)
POST	`/api/v1/categories`	Créer une nouvelle catégorie	JWT requis
PUT	`/api/v1/categories/{id}`	Modifier une catégorie	JWT requis
DELETE	`/api/v1/categories/{id}`	Supprimer une catégorie	JWT requis

Health & Monitoring

Méthode	Endpoint	Description	Authentification
GET	`/actuator/health`	Health check	Non
GET	`/actuator/health/liveness`	Probe Kubernetes liveness	Non
GET	`/actuator/health/readiness`	Probe Kubernetes readiness	Non

Modèles (DTOs)

`DemandeRequestDTO` (création de demande)

```
{
  "volume": 15.5,
  "natureMarchandise": "Meubles de salon",
  "dateDepart": "2025-12-15T10:00:00",
  "adresseDepart": "123 Rue Mohammed V, Casablanca",
  "adresseDestination": "456 Avenue Hassan II, Rabat",
  "categorieId": "cat-001-meubles"
}
```

Champ	Type	Obligatoire	Validation
`volume`	Double	☑	Doit être positif
`natureMarchandise`	String	☑	Non vide
`dateDepart`	LocalDateTime	☑	Doit être dans le futur
`adresseDepart`	String	☑	Non vide
`adresseDestination`	String	☑	Non vide
`categorieId`	String (UUID)	✗	ID d'une catégorie existante

`CategorieRequestDTO` (création de catégorie)

```
{
  "nom": "Produits Alimentaires",
  "description": "Produits alimentaires nécessitant une chaîne de froid",
  "densiteMoyenne": 850.0,
  "fragile": false,
  "dangereux": false,
  "temperatureRequise": "refrigere",
  "restrictions": "Respecter la chaîne du froid"
}
```

Champ	Type	Obligatoire	Validation
`nom`	String	☑	Non vide, unique
`description`	String	✗	Max 500 caractères
`densiteMoyenne`	Double	✗	Doit être positif ou nul (kg/m³)
`fragile`	Boolean	✗	Par défaut: false
`dangereux`	Boolean	✗	Par défaut: false
`temperatureRequise`	String	✗	ambiante, refrigere, congele (défaut: ambiante)
`restrictions`	String	✗	Max 500 caractères

`CategorieResponseDTO` (réponse)

```
{
  "idCategorie": "550e8400-e29b-41d4-a716-446655440000",
  "nom": "Produits Alimentaires",
  "description": "Produits alimentaires nécessitant une chaîne de froid",
  "densiteMoyenne": 850.0,
  "fragile": false,
  "dangereux": false,
  "temperatureRequise": "refrigere",
  "restrictions": "Respecter la chaîne du froid",
  "dateCreation": "2025-11-26T10:30:00",
  "dateModification": "2025-11-26T14:45:00"
}
```

`DemandeResponseDTO` (réponse)

```
{
  "id": 1,
  "clientId": 1,
  "volume": 15.5,
  "natureMarchandise": "Meubles de salon",
  "dateDepart": "2025-12-15T10:00:00",
  "adresseDepart": "123 Rue Mohammed V, Casablanca",
  "adresseDestination": "456 Avenue Hassan II, Rabat",
  "statutValidation": "EN_ATTENTE_CLIENT",
  "devisEstime": 1500.0,
  "itineraireAssocieId": 42,
  "groupeId": null,
  "categorie": {
    "idCategorie": "cat-001-meubles",
    "nom": "Meubles",
    "fragile": true,
    "temperatureRequise": "ambiante"
  },
```

```
  "dateCreation": "2025-11-25T22:30:00",
  "dateModification": "2025-11-25T22:30:00"
}
```

Catégories Prédéfinies

Le système inclut les catégories suivantes par défaut :

ID	Nom	Fragile	Dangereux	Température
cat-001-meubles	Meubles	☑	✗	ambiante
cat-002-electro	Électroménager	☑	✗	ambiante
cat-003-aliment	Produits Alimentaires	✗	✗	refrigere
cat-004-surgele	Produits Surgelés	✗	✗	congele
cat-005-constr	Matériaux de Construction	✗	✗	ambiante
cat-006-chimiq	Produits Chimiques	✗	☑	ambiante
cat-007-pharma	Produits Pharmaceutiques	☑	✗	refrigere
cat-008-texti	Textiles	✗	✗	ambiante

Énumérations

StatutValidation

Valeur	Description
EN_ATTENTE_CLIENT	Demande créée, en attente de validation
VALIDEE_CLIENT	Demande validée par le client
VALIDEE_PRESTATAIRE	Validée par le prestataire
TERMINEE	Demande terminée
ANNULEE	Demande annulée

Exemples de requêtes

1. Créer une demande de transport

```
POST /api/v1/demandes
Authorization: Bearer <jwt_token>
Content-Type: application/json

{
  "volume": 15.5,
  "natureMarchandise": "Meubles de salon",
```

```
  "dateDepart": "2025-12-15T10:00:00",
  "adresseDepart": "123 Rue Mohammed V, Casablanca",
  "adresseDestination": "456 Avenue Hassan II, Rabat"
}
```

Réponse (201 Created):

```
{
  "id": 1,
  "clientId": 1,
  "volume": 15.5,
  "natureMarchandise": "Meubles de salon",
  "statutValidation": "EN_ATTENTE_CLIENT",
  "devisEstime": 1500.00,
  ...
}
```

2. Lister mes demandes

```
GET /api/v1/demandes
Authorization: Bearer <jwt_token>
```

3. Voir une demande spécifique

```
GET /api/v1/demandes/1
Authorization: Bearer <jwt_token>
```

4. Valider une demande (accepter le devis)

```
PUT /api/v1/demandes/1/validation
Authorization: Bearer <jwt_token>
```

Réponse (200 OK):

```
{
  "id": 1,
  "statutValidation": "VALIDEE_CLIENT",
  ...
}
```

Authentification JWT

Tokens de test (valides 1 an)

Client (userId=1):

```
eyJhbGciOiJIUzM4NCJ9.eyJyb2xlIjoiQ0xJRU5UIiwidXNlcklkIjoxLCJzdWIiOiIxIiwiaWF0IjoxNzY0MTA4NjQ4LCJleHAiOjE3OTU2NDQ2NDh9.MsAIo8mq0sGFYTZ5XNK8oHU-fcQhZNCRWIJ_CxTtB2sau88MBHz4JiO6-DhhqHnl
```

Admin (userId=2):

```
eyJhbGciOiJIUzM4NCJ9.eyJyb2xlIjoiQURNSU4iLCJ1c2VySWQiOjIsInN1YiI6IjIiLCJpYXQiOjE3NjQxMDg2NDksImV4cCI6MTc5NTY0NDY0OX0.8PeDf_FMo6_e7sIWNVV-UeNXzrM4qobpQikw0jBBMFsQMomi2e-bN8PX-QFlggu9
```

Transporteur (userId=3):

```
eyJhbGciOiJIUzM4NCJ9.eyJyb2xlIjoiVFJBTlNQT1JURVVSIiwidXNlcklkIjozLCJzdWIiOiIzIiwiaWF0IjoxNzY0MTA4NjQ5LCJleHAiOjE3OTU2NDQ2NDl9.KKNKKVgXw0eHXsS3rErM1eDyyLN-2SZjUW6tJrJ0gA_rD7Uuus3Zk3Xjt1MDX48d
```

Générer de nouveaux tokens

```
./mvnw compile exec:java "-Dexec.mainClass=ma.tna.microservice3.util.JwtTokenGenerator"
```

Format du header

```
Authorization: Bearer <token>
```

Payload JWT attendu

```
{
  "sub": "1",
  "userId": 1,
  "role": "CLIENT",
  "iat": 1764108648,
  "exp": 1795644648
}
```

Health Check

Vérifier la santé de l'application

```
GET /actuator/health
```

Réponse:

```
{
  "status": "UP",
  "groups": ["liveness", "readiness"]
}
```

Probes Kubernetes

```
GET /actuator/health/liveness   → Application vivante ?
GET /actuator/health/readiness  → Application prête ?
```

Configuration

Variables d'environnement

Variable	Description	Valeur par défaut
`SERVER_PORT`	Port de l'application	`8083`
`SPRING_DATASOURCE_URL`	URL de connexion PostgreSQL	`jdbc:postgresql://localhost:5433/demandes_db`
`SPRING_DATASOURCE_USERNAME`	Utilisateur PostgreSQL	`demandes_user`
`SPRING_DATASOURCE_PASSWORD`	Mot de passe PostgreSQL	`demandes_password`
`JWT_SECRET`	Clé secrète JWT (Base64)	Voir `application.properties`
`SERVICE_URL_ITINERAIRES`	URL Service Itinéraires	`http://localhost:8084/api/v1/itineraires`

Variable	Description	Valeur par défaut
`SERVICE_URL_TARIFICATION`	URL Service Tarification	`http://localhost:8085/api/v1/tarifs`
`SERVICE_URL_MATCHING`	URL Service Matching	`http://localhost:8088/api/v1/matching`

Docker

Lancer uniquement la base de données

```
docker-compose up -d postgres-demandes
```

Lancer tout (application + base de données)

```
docker-compose up -d
```

Build et lancer (première fois ou après modifications)

```
docker-compose up --build -d
```

Rebuild sans cache (si problèmes de cache)

```
docker-compose build --no-cache service-demandes
docker-compose up -d
```

Redémarrer les services

```
# Redémarrer tous les services
docker-compose restart

# Redémarrer uniquement l'application
docker-compose restart service-demandes
```

Arrêter les services

```
# Arrêter (garde les conteneurs)
docker-compose stop
```

```
# Arrêter et supprimer les conteneurs
docker-compose down

# Arrêter et supprimer tout (volumes inclus)
docker-compose down -v
```

Voir les logs

```
# Tous les logs
docker-compose logs -f

# Logs de l'application uniquement
docker-compose logs -f service-demandes

# Dernières 100 lignes
docker-compose logs --tail=100 service-demandes
```

Vérifier l'état des conteneurs

```
docker-compose ps
```

Accéder à PostgreSQL

```
docker exec -it demandes_db_ms3 psql -U demandes_user -d demandes_db
```

Commandes utiles

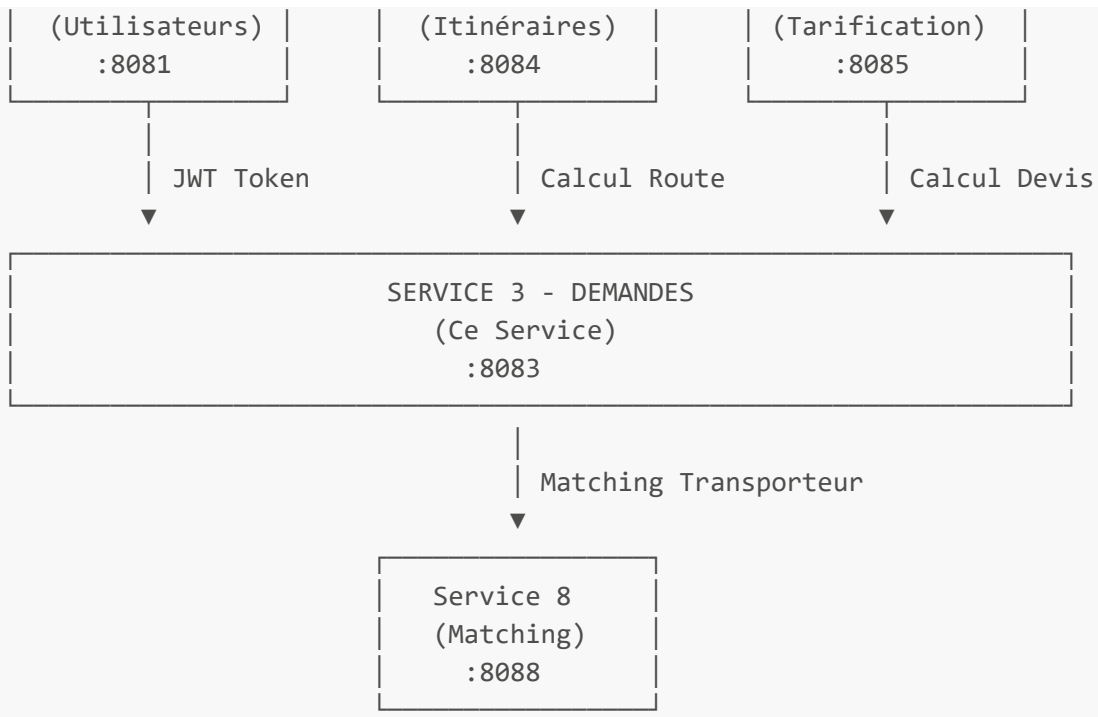
```
# Voir les images construites
docker images | grep microservice3

# Supprimer l'image pour forcer un rebuild complet
docker rmi microservice3-service-demandes

# Nettoyer les ressources Docker non utilisées
docker system prune -f
```

Architecture

```
┌──────────────────┐     ┌──────────────────┐     ┌──────────────────┐
│   Service 1      │     │   Service 4      │     │   Service 5      │
```



Collection Postman

Le fichier `Service3_Demandes_Transport.postman_collection.json` contient :

- 🏥 **Health & Status** - Endpoints de monitoring
- 📦 **Demandes CRUD** - Créer, lister, voir, valider
- 📝 **Exemples** - Différents types de demandes
- 🔒 **Tests Sécurité** - Vérification authentification

Le token JWT est pré-configuré - importez et testez directement !

Technologies

Technologie	Version	Usage
Java	21	Langage
Spring Boot	3.5.8	Framework
Spring Security	6.x	Authentification JWT
Spring Data JPA	3.x	Accès base de données
PostgreSQL	16	Base de données
Docker	-	Containerisation
Maven	3.9+	Build
JJWT	0.12.3	Gestion tokens JWT
Lombok	-	Réduction boilerplate

```
## Auteur

**MicroService3** - Service Demandes de Transport
Fait partie de l'architecture microservices de Transport Maroc.
```