🚀 Serveur de Logs Centralisé

📋 Description

Serveur de logs centralisé haute performance développé en Java pour collecter, traiter et stocker des logs depuis plusieurs applications clientes. Implémente un buffer circulaire avec mécanisme de back-pressure, traitement multi-threadé, et stockage avec rotation automatique.

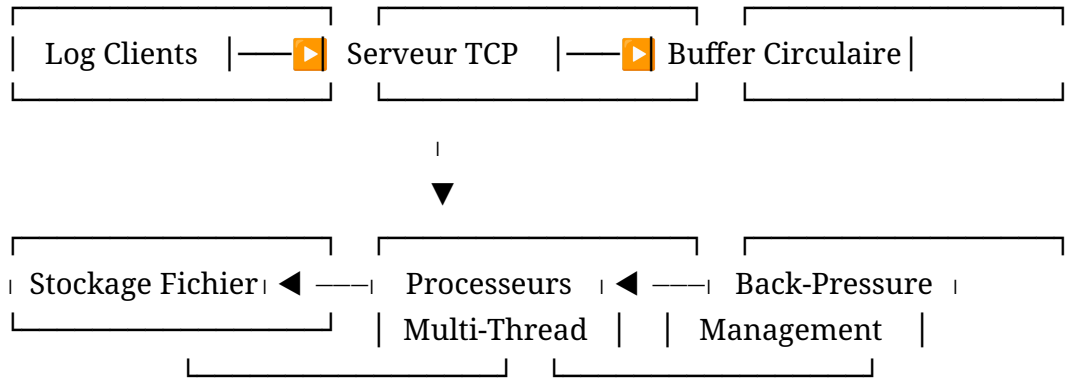
✨ Fonctionnalités

Architecture:
- Buffer circulaire thread-safe avec gestion de la surcharge
- Mécanisme de back-pressure intelligent
- Traitement multi-threadé par batch
- Stockage avec rotation quotidienne par application (FileLogStorage)
- Interface console interactive pour monitoring

Monitoring:
- Statistiques temps réel du serveur
- Suivi du buffer et du stockage
- Suivi des clients connectés
- Métriques de performance

Connectivité:
- Support multi-clients simultanés
- Commandes de contrôle intégrées
- Formats de logs flexibles
- Métadonnées enrichies automatiquement

🏗️ Architecture Technique

```


```

⚙️ Configuration

application.properties:

```
spring.application.name=server_centralise
server.port=8080            # Port d'écoute du serveur
server.maxClients=50        # Nombre maximum de clients
buffer.size=1000            # Taille du buffer circulaire
storage.directory=./logs    # Répertoire de stockage (reste en file)
threads.processor=4         # Nombre de threads processeurs
```

🚀 Installation et Démarrage

Prérequis:

- Java 11 ou supérieur
- Maven 3+
- 512 MB RAM minimum

Compilation & Package avec Maven:

```
mvn clean compile
mvn package
```

```

Le JAR généré: target/server_centralise-1.0-SNAPSHOT.jar

Démarrage:
```
java -jar target/server_centralise-1.0-SNAPSHOT.jar
java -Xmx1g -Xms512m -jar target/server_centralise-1.0-SNAPSHOT.jar
```

## 💻 Utilisation

Interface console:
```
LogServer> help
LogServer> status
LogServer> stats
LogServer> buffer
LogServer> clients
LogServer> memory
LogServer> stop
```

Client de Test:
```
java -cp target/server_centralise-1.0-SNAPSHOT.jar
com.univ.logserver.client.LogClient localhost 8080 MonApp
java -cp target/server_centralise-1.0-SNAPSHOT.jar
com.univ.logserver.client.LogClient localhost 8080 TestApp 1000 10
```

## 📝 Format des Messages

Format Complet:
LEVEL|APPLICATION|HOSTNAME|MESSAGE|metadata_key=value,key2=value2

Format Simple:
LEVEL MESSAGE
```

Exemples:
INFO|WebApp|server01|User login successful|user_id=123,session=abc
ERROR|DatabaseApp|db-server|Connection timeout|
retry_count=3,duration=5000ms
WARN Memory usage high

🔍 Surveillance et Monitoring

Métriques Clés:
- Buffer: utilisation, back-pressure, messages supprimés
- Stockage: fichiers créés, logs stockés, taille totale
- Clients: connexions actives, messages reçus/rejetés
- Performance: throughput, latence, utilisation mémoire

Logs de Rotation:
logs/
├── WebApp_2025-08-15.log
├── DatabaseApp_2025-08-15.log
└── ApiGateway_2025-08-15.log

🧪 Tests

Exécution des Tests:
```
mvn test
java -cp target/server_centralise-1.0-SNAPSHOT.jar
com.univ.logserver.LogServerTest testServerClientIntegration
java -cp target/server_centralise-1.0-SNAPSHOT.jar
com.univ.logserver.LogServerTest testMultipleClientsLoad
```
Couverture des Tests:
- Modèles de données (LogEntry, LogLevel)
- Parser de messages avec validation
- Buffer circulaire et back-pressure
- Stockage fichier avec rotation
- Intégration serveur-client
- Tests de charge multi-clients

- Thread-safety et accès concurrent
- Gestion d’erreurs et cas limites

🔧 Maintenance

Gestion des Logs:

```
find ./logs -name "*.log" -mtime +30 -delete
gzip logs/*.log
```

Optimisation Performance:
- Ajuster buffer.size selon le volume de logs
- Augmenter threads.processor sur serveurs multi-cœurs
- Configurer la JVM avec -Xmx approprié
- Monitoring régulier du back-pressure

🐛 Dépannage

Problèmes Courants:
Buffer plein / Back-pressure actif:
Solution: Augmenter buffer.size ou threads.processor
Connexions refusées:
Solution: Vérifier server.maxClients et port
Fichiers de logs volumineux:
Solution: Rotation + compression

👥 Architecture Détaillée

Composants Principaux:
1. ServerConfig
2. LogEntry / LogLevel
3. CircularBuffer
4. FileLogStorage
5. LogParser / LogProcessor
6. LogServer / ClientHandler
7. LogClient
8. LogServerApplication

Flux de Traitement:
Client → TCP → Parser → Buffer → Processor → Storage → Fichiers