Spécification Fonctionnelle Détaillée (US01)

User Story 01 : Initialisation du service de fond

En tant que système, je veux démarrer un service de fond qui initialise les timeframes et les symboles.

1. Objectif de la fonctionnalité

Cette fonctionnalité constitue l'étape initiale indispensable au fonctionnement du module `MarketDataCollector`. Elle permet la mise en place des éléments nécessaires à la collecte temps réel des données de marché : configuration des timeframes, accès à la liste des symboles actifs, construction dynamique de l'URL WebSocket Binance, initialisation des buffers internes et lancement de la session d'écoute WebSocket.

Le bon démarrage de cette chaîne garantit la disponibilité du flux de données, prérequis à toute détection ou analyse ultérieure.

2. Acteurs

- **Système** : le service backend démarrant le process de collecte (BackgroundService .NET).
- **SymbolStore** : module ou composant responsable de fournir la liste actualisée des symboles de trading actifs.
- **Binance WebSocket Endpoint** : point d'entrée externe pour la récupération des klines (bougies).
- **Configuration applicative** : fichier `appsettings.json` ou provider d'injection des paramètres runtime.

3. Hypothèses

- La configuration contient une clé `Timeframes` (liste de strings comme "M1", "M5", "M15").
- Le composant SymbolStore expose une méthode synchrone ou asynchrone permettant d'obtenir les symboles actifs sous forme de liste.
- Les symboles sont valides pour Binance (ex : BTCUSDT, ETHUSDT, etc.).
- Le démarrage du service est systématique au lancement de l'application (hosté dans une API ou un worker).

4. Entrées

- Paramètres de configuration :

- `Timeframes` : ["M1", "M5", "M15"]

- `MaxSymbols` : nombre maximal de symboles à surveiller (par défaut : 200)

- Données runtime :

- Liste des symboles actifs retournée par le SymbolStore

5. Résultat attendu (sorties)

- Construction d'une URL WebSocket de type :

```
wss://stream.binance.com:9443/stream?streams=btcusdt@kline_1m/
ethusdt@kline_1m/btcusdt@kline_5m/...
```

- Initialisation des buffers internes pour chaque (symbol, timeframe).
- Connexion active au WebSocket Binance, prête à recevoir les messages kline.

6. Déroulé fonctionnel

6.1 Lecture de la configuration

- Le service lit les timeframes depuis la section de configuration dédiée.
- En cas d'absence ou d'erreur de parsing, le service s'arrête avec un log d'erreur explicite.

6.2 Récupération des symboles

- Appel à `SymbolStore.GetActiveSymbols()`.
- Le résultat est filtré pour ne pas dépasser le seuil `MaxSymbols`.
- Si la liste retournée est vide ou nulle, le service loggue et ne tente pas d'écoute WebSocket.

6.3 Construction dynamique de l'URL WS Binance

- Pour chaque symbole, le service crée une chaîne de souscription de type :
- `btcusdt@kline_1m`, `btcusdt@kline_5m`, etc.
- Ces chaînes sont concaténées dans l'URL WebSocket Binance (max 1024 streams).

6.4 Initialisation des buffers FIFO

- Pour chaque (symbol, timeframe), le service crée une structure mémoire circulaire de taille X (configurable).
- Les buffers gèrent : volume, close, high-low (volatilité).

6.5 Connexion WebSocket

- Le service ouvre la connexion WebSocket vers Binance
- Il est en mode passif : il attend les messages avant de passer à la phase de traitement.

7. Règles de gestion

- Si le nombre de symboles > 1024 * nbTimeframes, une alerte critique est logguée.
- Si le SymbolStore renvoie une erreur, l'initialisation est stoppée.
- Si Binance refuse la connexion, une politique de retry (configurable) est activée.

8. Scénarios d’erreurs

- SymbolStore inaccessible : échec de démarrage.
- WebSocket Binance inaccessible : retry 3 fois, puis log et déconnexion.
- Timeframes invalides : parsing KO = log erreur critique.

9. Réversibilité

- Aucun état persistant n'étant modifié, le redémarrage du service permet une réinitialisation complète.

10. KPI et observabilité

- Dernière URL WebSocket construite
- Nombre de symboles chargés
- Nombre total de souscriptions calculées
- Timestamp de la dernière initialisation

11. Limites fonctionnelles

- Le service ne vérifie pas la validité des symboles auprès de Binance avant la souscription.
- Aucune persistance ou cache n'est prévu pour les buffers initiaux.
- Le SymbolStore doit être disponible localement dans le même process (pas d'appel HTTP).

12. Points ouverts

- Doit-on notifier via SignalR la liste des symboles souscrits après initialisation ?
- Une mécanique de test local (dry-run) est-elle nécessaire pour valider le setup sans connexion WS ?

Souhaites-tu que je passe à la version technique maintenant ?