A logo with red and black text

Description automatically generated

# **Finansal Veri Sağlayıcılardan Anlık Veri Toplama ve Hesaplama Projesi**

**Teknik Dokümanı**

**Doküman Tarihçesi**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tarih** | **Versiyon** | **Açıklama** | **Hazırlayan** |
| 26/11/2024 | 0.01 | İlk Versiyon | Elif Demirpolat |
|  |  |  |  |
|  |  |  |  |

# **KAPSAM ve AMAÇ**

# Küresel finansal piyasalarda anlık veri dağıtımı yapan çeşitli uzmanlaşmış kuruluşlar bulunmaktadır. Bu proje, söz konusu çoklu veri sağlayıcıların sistemlerine bağlanarak, finansal verilerin entegre edilmesi, değerlendirilmesi ve hesaplanmasını amaçlayan kapsamlı bir yazılım çözümü geliştirmeyi hedeflemektedir. Geliştirilen uygulama, farklı platformlardan gelen verileri koordine edecek ve bunlar üzerinden türev finansal veriler hesaplayacaktır. Veri hesaplama metodolojileri proje ilerleyen aşamalarında detaylandırılacaktır. Ana uygulama, dinamik sayıda veri sağlayıcıdan bilgi toplayabilen esnek bir Java tabanlı yazılım altyapısı olarak tasarlanmıştır.

# Proje özelinde Forex (döviz piyasası) verileri ele alınacaktır. Bu veri setlerinde zorunlu alan olarak alış fiyatı (bid), satış fiyatı (ask) ve zaman damgası (timestamp) bulunacak olup, her veri setinin benzersiz bir tanımlayıcı kodu da yer alacaktır.

# Proje kapsamında aşağıdaki uygulamalar geliştirilecektir:

# En az iki adet veri platformu simülasyon uygulaması

# Platformlardan veri toplama, hesaplama ve yayınlama işlevine sahip merkezi bir ana uygulama

# Kafka kümesine yazılan verileri veritabanına aktaran bir Kafka Tüketicisi (Consumer)

# Opsiyonel gelişmeler:

# Yerel bellek yerine dağıtık bellek yapıları (Hazelcast veya Redis) kullanımı

# Opensearch üzerinden log verilerine dayalı bir gösterge paneli (dashboard) oluşturulması

# **Genel Beklentiler**

Proje kapsamındaki tüm yazılım bileşenleri için aşağıdaki teknik ve mimari standartlar belirlenmiştir:

1. **Programlama Dili** Projenin tamamı Java programlama dili kullanılarak geliştirilecektir.
2. **Altyapı Gereksinimi** Proje bileşenlerinden en az bir tanesi Spring Boot altyapısı kullanılarak implement edilecektir..
3. **Log Yönetimi** Tüm sınıflarda Log4j2kullanılarak loglama yapılacaktır. Log seviyeleri fatal, error, warn, info, debug, trace olmalıdır.
4. **Yapılandırma Yönetimi** Uygulamalar yüksek derecede parametrik olacak şekilde tasarlanacak, tüm yapılandırma parametreleri konfigürasyon dosyalarından okunacaktır. Bu yaklaşım, uygulamaların esnekliğini ve yönetilebilirliğini artıracaktır.
5. **Konteyner Mimarisi** Yazılım bileşenleri konteyner mimarisine uygun olarak geliştirilecektir. Bununla birlikte, her bileşenin bağımsız çalışabilen (standalone) sürümlerinin de hazırlanması artı puan olacaktır.
6. **Kod ve Dokümantasyon Standardı** Loglama değişkenlerinde İngilizce kullanılacaktır. Değişken isimlendirmeleri için net, açıklayıcı ve tutarlı bir standart izlenecektir. Kod okunabilirliği ve anlaşılabilirliği ön planda tutulacaktır.

# **VERİ PLATFORMLARI SİMULASYONU**

Veri toplayıcı sınıfların bağlanacağı platformlar için minimum iki adet platform simülasyonu uygulaması beklenmektedir. Bu simulasyonlardan en az bir tanesi streaming rate sağlamalı yani bir TCP socket üzerinden veri akışı yapabilmelidir, diğeri ise REST API aracılığı ile istek üzerine veri sağlayabilmelidir,

**TCP ile Veri Sağlayan Uygulama için Veri Akışı ve Test Örneği:**

**Test Adımları :**

* Uygulama çalıştırılır,
* Telnet aracılığı ile uygulamaya bağlantı sağlanır.
* Uygulamanın kendi makinanızda 8081 portundan ayağa kalktığı durumda test için aşağıdaki gibi bir senaryo kullanılabilir. Kırmızı renkli kısımlar istek, yeşil kısımlar ise platformdan gelen yanıtı örneklemektedir.

telnet 127.0.0.1 8081

Connected to 127.0.0.1.

Escape character is '^]'.

subscribe|PF1\_USDTRY

Subscribed to PF1\_USDTRY

PF1\_USDTRY|22:number:34.40135543466431|25:number:35.40135543466431|5:timestamp:2024-12-15T11:31:34.509

PF1\_USDTRY|22:number:34.358074486976214|25:number:35.358074486976214|5:timestamp:2024-12-15T11:31:42.490

subscribe|PF1\_EURUSD

Subscribed to PF1\_EURUSD

PF1\_EURUSD|22:number:1.0594055684459591|25:number:1.0804055684459593|5:timestamp:2024-12-15T11:32:13.593

PF1\_USDTRY|22:number:34.40046476019531|25:number:35.40046476019531|5:timestamp:2024-12-15T11:32:14.484

PF1\_EURUSD|22:number:1.050697862658565|25:number:1.0716978626585651|5:timestamp:2024-12-15T11:32:21.593

PF1\_USDTRY|22:number:34.37766805530889|25:number:35.37766805530889|5:timestamp:2024-12-15T11:32:22.485

PF1\_EURUSD|22:number:1.0485447797985348|25:number:1.069544779798535|5:timestamp:2024-12-15T11:32:29.592

PF1\_USDTRY|22:number:34.34733728455729|25:number:35.34733728455729|5:timestamp:2024-12-15T11:32:30.484

PF1\_EURUSD|22:number:1.0538905797830445|25:number:1.0748905797830446|5:timestamp:2024-12-15T11:32:37.593

PF1\_USDTRY|22:number:34.3672463330723|25:number:35.3672463330723|5:timestamp:2024-12-15T11:32:38.485

subscribe|PF1\_TRYUSD

ERROR|Rate data not found for PF1\_TRYUSD

uns

ERROR|Invalid request format

unsubscribe|PF1\_USDTRY

Örnekte;

*telnet 127.0.0.1 8081* komutu ile uygulamaya erişilmiştir.

*subscribe|PF1\_USDTRY*  komutu ile USD.TRY rate ine subscribe olunmuştur.

Kullanılan formatlar örnek amaçlıdır, geliştirici kendi formatlamasını yapmakta serbesttir.

**TCP ile Veri Sağlayan Uygulama için Beklentiler:**

* Uygulama Java dilinde yazılmalı ve Standalone bir uygulama olmalıdır.
* Uygulama başlangıç verilerini bir konfigurasyon dosyasından okuyabilir.
* Başlangıç verileri üzerinde gerçek hayattaki kur dalgalanmasını simule edecek bir algoritma mantığı kurulabilir, bu noktada algoritma mantığı kurularak doğal akışa benzetilen simulasyonlar artı puan olarak değerlendirilir..
* Değerler arasına ne kadar fark eklenip çıkarılacağı belirlenerek yayın devam ettirilebilir.
* Konfigurasyon dosyası içerisinde yayın sıklığı ve kaç adet yayın yapılacağı da belirtilmelidir.

**REST API ile Veri Sağlayan Uygulama için Veri Akışı ve Test Örneği:**

**Test Adımları :**

* Postman üzerinden istek örneği

http://localhost:8080/api/rates/PF2\_USDTRY

* Yanıt Örneği (Hatalı istek atılması durumu için uygun yanıtlar oluşturulmalıdır.)

{

"rateName": "PF2\_USDTRY",

"bid": 34.44561303491713,

"ask": 35.43720648609209,

"timestamp": "2024-12-14T21:18:21.178245087"

}

Kullanılan formatlar örnek amaçlıdır, geliştirici kendi formatlamasını yapmakta serbesttir.

**REST API ile Veri Sağlayan Uygulama için Beklentiler:**

* Uygulama java dilinde spring boot altyapısı kullanılarak hazırlanmalıdır,
* Uygulama başlangıç verilerini bir konfigurasyon dosyasından okuyabilir.
* Başlangıç verileri üzerinde gerçek hayattaki kur dalgalanmasını simule edecek bir algoritma mantığı kurulabilir, bu noktada algoritma mantığı kurularak doğal akışa benzetilen simulasyonlar artı puan olarak değerlendirilir..
* Değerler arasına ne kadar fark eklenip çıkarılacağı belirlenerek yayın devam ettirilebilir.
* Konfigurasyon dosyası içerisinde yayın sıklığı ve kaç adet yayın yapılacağı da belirtilmelidir.

# **ANA UYGULAMA**

Uygulama farklı platformlardan çekilecek verileri ayıklayıp, hesaplayacak hesaplanmış verileri ve hesap sırasında kullanılan verileri DB ye aktaracak aynı zamanda da loglama yapacaktır.

**Ana Uygulama için Beklentiler:**

* Platformlardan verileri çekmek için kullanılacak sınıflar bağımsız threadler içinde çalışmalıdır. Bu sınıflar bir interface i implement etmeli ya da bir abstract sınıftan türemelidir. Olması beklenen minimum methodlar aşağıdaki gibidir.

// Bağlantıyı gerçekleştirmek için olan metod

connect(String platformName, String userid, String password)

// Bağlantıyı kesmek için olan metod

disConnect(String platformName, String userid, String password)

// Bir rate e subscribe olmak için çağrılacak metod

subscribe(String platformName, String rateName)

// Bir rate e subscription ı bitirmek için çağrılacak metod

unSubscribe(String platformName, String rateName)

* Kontrol edebilmek için koordinatör sınıfı oluşturulmalıdır. Koordinatör sınıf aşağıda örnek olarak paylaşılan callback metodlarını içeren bir interface’i implement etmeli ya da abstract sınıftan türetilmelidir.

// Bağlantı gerçekleştiğinde çalışacak callback

onConnect(String platformName, Boolean status)

// Bağlantı koptuğunda çalışacak callback

onDisConnect(String platformName, Boolean status)

// istenen veri ilk defa geldiğinde

onRateAvailable(String platformName, String rateName, Rate rate)

// istenen verinin sonraki güncellemeleri

onRateUpdate(String platformName, String rateName, RateFields rateFields)

// istenen verinin durumu ile ilgili bilgilendime

onRateStatus(String platformName, String rateName, RateStatus rateStatus)

* Veri toplama uygulamasının ana sınıfı Koordinatör sınıf olmalıdır. Bu sınıf içerisinde veri toplama sınıflarını ayağa kaldırma, kapatma, bu sınıflardan gelen verileri işleme, hesaplama, loglama işlemleri yapılabilmelidir.
* Veri toplama sınıfları Koordinator sınıfında implement edilmiş callback metodlarını çağırarak koordinator sınıfı ile haberleşmelidir, bu amaçla koordinator sınıfı veri toplayan sınıfları ayağa kaldırırken onlara bir isim vermeli ve callback’ler için kendi referansını geçirmelidir.
* Veri toplama sınıflarından gelen veriler ortak bir formata çevrilmelidir.
* Formatlanan veriler Koordinatör Uygulaması tarafından cache de saklanmalıdır. Bu noktada Cache basit bir map uyarlaması olarak geliştirilebilir. Ancak harici olarak Hazelcast ya da Redis kullanımı artı puan olarak değerlendirilecektir.
* Koordinator sınıfı veri toplayan sınıfları bir tanım dosyası yardımı ile okumalı, bu sınıfların neler olduğunu kodlama sırasında bilmemeli sadece interface veya abstract class tan haberdar olmalıdır.

Bu sayede ana yapı bozulmadan ileride farklı implementasyonlar eklemek mümkün olacaktır. Konfigürasyon dosyasında bu sınıfların implementation classlarının ismi yer almalı ve runtime da dinamik yüklenmelidirler.

* Runtime da dinamik yüklenen bu sınıflar kendi konfigürasyon dosyalarından yapacağı bağlantı ile ilgili bilgileri almalıdır, hangi rate lere subscription yapılacağı ana koordinator uygulaması tarafından yönetilecektir.
* Koordinasyon uygulamasındaki onRateAvailable ve onRateUpdate callback metodları tetiklendiğinde ortak bir formata gelmiş veriler Kafka cluster a gönderilmelidir.
* Ham veriler örneklerle açıklanan hesaplama metodları ile hesaplanmalı ve calculated data ya ulaşılmalıdır.
* Hesaplanan kurlar ile ilgili hesaplama yöntemi dinamik olmalıdır hesaplama anında nasıl bir formül kullanılacağı ve bu formülün hangi yöntem ile derleneceği konfigürasyonda belirtilmelidir.
* Hesaplama yöntemi için harici bir java sınıf kullanılabileceği gibi, JVM scripting dillerinden Javascript veya Groovy kullanılabilir. Her birinin ayrı kullanımı artı puan olacaktır.

**Örnek Senaryo :**

Platform 1 Verileri:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Örnek Veri** | | |
| **Sembol** | **Açıklama** | **Alış (bid)** | **Satış(ask)** | **Zaman (timestamp)** |
| PF1\_USDTRY | ABD Doları /Türk lirası verisi | 33.60 | 35.90 | 2024-12-16T16:07:15.504 |
| PF1\_EURUSD | Euro / ABD Doları verisi | 1.0220 | 1.0450 | 2024-12-16T16:07:25.504 |
| PF1\_GBPUSD | Sterlin / ABD Doları verisi | 1.2510 | 1.2645 | 2024-12-16T16:07:27.504 |

Platform 2 Verileri:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Örnek Veri** | | |
| **Sembol** | **Açıklama** | **Alış (bid)** | **Satış(ask)** | **Zaman (timestamp)** |
| PF2\_USDTRY | ABD Doları /Türk lirası verisi | 34.80 | 35.10 | 2024-12-16T16:07:16.504 |
| PF2\_EURUSD | Euro / ABD Doları verisi | 1.0370 | 1.0410 | 2024-12-16T16:07:26.504 |
| PF2\_GBPUSD | Sterlin / ABD Doları verisi | 1.2590 | 1.2615 | 2024-12-16T16:07:28.504 |

Tablolarda bulunan USD/TRY, EUR/USD, GBP/USD verileri ham data kabul edilerek, USD/TRY, EUR/TRY, GBP/TRY oranları hesaplanacaktır.

USD.TRY platformlardan direkt erişilebilir durumda iken EUR.TRY ya da GBP.TRY oranlarına platformlardan direkt erişilemez, belli hesaplar ile türetilmeleri gerekir.

Tablolarda Sembol kolonu altında isimler örnek olarak standardize edilmiştir.

**USD/TRY hesabı ,**

USD.TRY oranı BID ve ASK değerleri için iki platformun ortalaması alınmaktadır.

USDTRY.bid = (PF1\_USDTRY.bid+PF2\_USDTRY.bid)/2

USDTRY.ask = (PF1\_USDTRY.ask+PF2\_USDTRY.ask)/2

(33.6+34.8)/2= 34.2

(35.9+35.1)/2= 35.5

**EUR/TRY hesabı ,**

EUR.TRY oranı BID ve ASK değerleri için EUR.USD (iki platform ortalaması) ve USDmid oranı çarpılmalıdır.

usdmid = ((PF1\_USDTRY.bid+PF2\_USDTRY.bid)/2 +(PF1\_USDTRY.ask+PF2\_USDTRY.ask)/2 )/2

EURTRY.bid =usdmid x ((PF1\_EURUSD.bid+PF2\_EURUSD.bid)/2)

EURTRY.ask =usdmid x ((PF1\_EURUSD.ask+PF2\_EURUSD.ask)/2)

((33.6+34.8)/2+(35.9+35.1)/2)/2=34.85

34.58 x (1.022+1.037)/2 =35.878075

34.58 x (1.045+1.041)/2 =36.34855

**GBP/TRY hesabı ,**

GBP.TRY oranı BID ve ASK değerleri için GBP.USD (iki platform ortalaması) ve USDmid oranı çarpılmalıdır.

usdmid = ((PF1\_USDTRY.bid+PF2\_USDTRY.bid)/2 +(PF1\_USDTRY.ask+PF2\_USDTRY.ask)/2 )/2

GBPTRY.bid =usdmid x ((PF1\_GBPUSD.bid+PF2\_GBPUSD.bid)/2)

GBPTRY.ask =usdmid x ((PF1\_GBPUSD.ask+PF2\_GBPUSD.ask)/2)

((33.6+34.8)/2+(35.9+35.1)/2)/2=34.85

34.58 x (1.251+1.259)/2 =43.73675

34.58 x (1.2645+1.2615)/2 =44.01555

* Hesaplanan bu veriler için hesaplama formülleri uygulama dışından dinamik olarak verilecektir, bu noktada java sınıfı, javscript veya groovy tercih edilebilir.
* Platformlardan sağlanan veriler güncellendiğinde kendisinin hangi hesaplamalarda kullanıldığını bilmeli ve ona göre ilgili verinin hesaplama metodunu çağırabilmelidir.

# **KAFKA CONSUMER**

* Kafka consumer uygulamaları kafka clusterlarına gelen verileri dinleyecektir .
* Consumerlardan biri gelen veriyi bir veritabanına yazacak , bir diğeri ise opensearch / elasticsearch gibi bir platformu besleyerek loglama yapacaktır.
* Veritabanını besleyen uygulamada JPA API kullanması , veritabanı olarak ise PostgreSQL tercih edilmesi beklenmektedir.

**Veri Tabanı Tablo Deseni:**

TblRates

id : integer (autoincrement)

rateName : varchar(10;) (veriden gelecek)

bid : decimal (veriden gelecek)

ask : decimal (veriden gelecek)

rateUpdatetime : Timestamp (veriden gelecek)

dbUpdatetime : Timestamp

**Kafka ya Yazılacak Veri Formatı Örneği:**

PF1\_USDTRY|33.60|35.90|2024-12-16T16:07:15.504

PF2\_USDTRY|34.80|35.10|2024-12-16T16:07:16.504

USDTRY|34.20|35.50|2024-12-16T16:07:16.504

PF1\_EURUSD|1.022|1.045|2024-12-16T16:07:25.504

PF2\_EURUSD|1.037|1.041|2024-12-16T16:07:26.504

EURUSD|35.878075|36.3485|2024-12-16T16:07:26.504

PF1\_GBPUSD|1.251|1.2645|2024-12-16T16:07:27.504

PF2\_GBPUSD|1.259|1.2615|2024-12-16T16:07:28.504

GBPUSD|43.73675|44.01555|2024-12-16T16:07:28.504

# **MİMARİ**

