Einführung und Ziele

Beschreibt die wesentlichen Anforderungen, die bei der Umsetzung der Softwarearchitektur und Entwicklung des Systems berücksichtigt werden müssen.

Dazu gehören:

- Projekt-Rahmenbedingungen
- Projekt-Mindestumfang
- Bewertungskriterien - Auf was legt er Wert?
- wesentliche funktionale Anforderungen,
- wesentliche nicht funktionale Anforderungen
- Domänenmodell
- Komponentenmodell

Rahmenbedingungen

- **Beginn**: Dienstag, 03.12.2024
- **Code Freeze**: Sonntag, 26.01.2024, 23:59
- **Abgabe Gesamtpaket** (Code + Dokumentation + Präsentation): Montag, 26.01.2024, 23:59
- **Präsentation**: Dienstag, 28.01.2024

Projekt-Mindestumfang

- **Reaktive Kotlin Spring Boot Webflux Server Applikation**
- **Anbindung externer API(s)**
- **Anbindung einer Datenbank mit R2DBC**
- **Kleines Frontend**, das eure Applikation ein wenig visualisiert
  - Kann so hacky gebaut sein, wie ihr wollt
  - Fließt, wenn überhaupt, nur geringfügig in die Bewertung ein
- **Test Coverage**: Mindestens für die Kernfunktionen
- **Ausreichende Dokumentation**:
  - **Kommentare**: Falls der Code nicht selbsterklärend ist (*besser: Code so schreiben, dass er selbsterklärend ist*)
  - **README**: Anleitung zur Konfiguration und Nutzung
    - **Optionale Dokumentation**: Erklärungen, warum bestimmte Entscheidungen getroffen oder Dinge so umgesetzt wurden

Bewertungskriterien - Auf was legt er Wert?

- **Idiomatic use of Kotlin features**, i.e.:
  - Scope functions
  - Extension functions where it makes sense
  - Passing functions as arguments
- **Clean Code**
- **Project and package structure / architecture**
- **Naming** of:
  - Classes
  - Functions
  - Variables
- **Separation of concerns**:
  - Who is responsible for what?
- **Grading structure**:
  - **Project grade** (Code + Documentation + Presentation)
  - **Individual grade**:
    - Based on particularly good or less good engagement within the group
  - These combine to form your **personal overall grade**.

Wesentliche funktionale Anforderungen

Minimal Viable Product (MVP)

0. Kontoverwaltung

Zweck: Der Nutzer kann hier Kapitel auf sein Konto legen.

Funktionalität:

- Man kann einen Betrag x auf das Konto legen. Anzugeben ist: der Betrag der auf das Konto soll

1. Depotverwaltung

Zweck: Der Nutzer kann hier mit seinem Geld vom Konto Wertpapiere kaufen, die dann im Depot liegen.

Funktionalität:

- Man kann mit einem Betrag x Wertpapiere für das eigene Depot kaufen.

2. Handelsmöglichkeiten für das Depot

Zweck: Der Nutzer kann hier mit seinem hinterlegtem Kapitel im Konto Wertpapiere kaufen oder gekaufte Wertpapiere verkaufen.

2.1 Buy-Orders

Zweck: Man kann mit einem Betrag x Wertpapiere einer Firma kaufen.

Funktionalität:

- Angeben beim Kauf: Wertpapier-Stückzahl, wann ausführt werden soll (jetzt, Zeitpunkt x), Kosten des Kaufes (auf Basis aktueller Kurswerte)
- nach Kauf verringert sich der Geldwert im Konto (siehe 1.)

2.1 Sell-Orders

Zweck: Man kann seine gekauften Wertpapiere verkaufen und erhällt dafür Geld auf dem Konto.

Funktionalität:

- Angeben beim Verkauf: Wertpapier-Stückzahl, wann ausführt werden soll (jetzt, Zeitpunkt x), Erlös des Kaufes (auf Basis aktueller Kurswerte)
- nach Verkauf erhöht sich der Geldwert im Konto (siehe 1.)

3. Depot

Zweck: Das Depot zeigt die aktuellen gekauften Wertpapiere und ihren Kurs an. Dies ist die Übersichtsseite. Klickt man auf ein Wertpapier gelangt man zu 4.

Funktionalität:

- Anzeige aller gekauften Wertpapiere als Liste mit Anzahl, Unternehmens-Namen, Kürzel, Wert pro Wertpapier, %ige-Zunahme/-Fall der letzten x Minuten im Vergleich zu jetzt -> Live-Update dieser Wertpapier-Werte

4. Darstellung ausgewählter Kurse

Zweck: Man kann sich nähere Informationen zu einem Unternehmen und deren Wertpapier anzeigen lassen. Diese Informationen werden live geupdated.

Funktionalität:

- Zeitraumeinschränkung des Wertpapier-Kurses möglich (z.B. von jetzt bis Zeitpunkt X).
- Berechnung des durchschnittlichen Kurswertes der letzten x Minuten
- %ige-Zunahme/-Fall der letzten x Minuten im Vergleich zu jetzt
- Unternehmens-Name + Kürzel

5. Suche von Wertpapieren

Zweck: Man kann hier den Namen des Wertpapiers/Kürzel angeben. Man kann sich die Suchergebnisse anschauen. Bei Klick auf ein Suchergebnis wird einem 4. angezeigt.

Funktionalität:

- Jedes Suchergebnis zeigt an: Unternehmens-Name + Kürzel, Kurswert, Ausgeführte Orders, Handelsvolumen
- Man kann die Suchergebnisse filtern nach: Land, ...

Stretch Goals

1. Abbildung von Kryptowährungen

Zweck:
Erweiterung des Systems zur Unterstützung von Kryptowährungen (z.B. Bitcoin, Ethereum).

Funktionalität:

- Siehe Wertpapier-Spezifikationen

2. Favoritenspeicherung

Zweck:
Der Nutzer kann Wertpapiere als Favoriten speichern und erhält Benachrichtigungen über deren Kursänderungen.

Funktionalität:

- es gibt einen Stern. Klickt man diesen, kann man auswählen, wann man benachrichtigt werden möchte.
- Der Stern wird unter 4. und 5. angezeigt
- Benachrichtigung möglich: bei signifikanten Kursänderungen | "Wunschkurs" ist erreicht -> muss beim favorisieren eingestellt werden, ab wann beides getriggert wird

4. Erweiterte Kauf- und Verkaufsbedingungen

Zweck:
Erweiterte Optionen für den Kauf und Verkauf von Wertpapieren mit preislichen und zeitlichen Bedingungen.

Funktionalität:

- Kauf/Verkauf von Wertpapieren nur zu einem bestimmten Preis (z.B. "nur kaufen, wenn der Kurs unter x liegt").
- Kauf/Verkauf nur zu einem bestimmten Zeitpunkt oder innerhalb eines festgelegten Zeitraums.

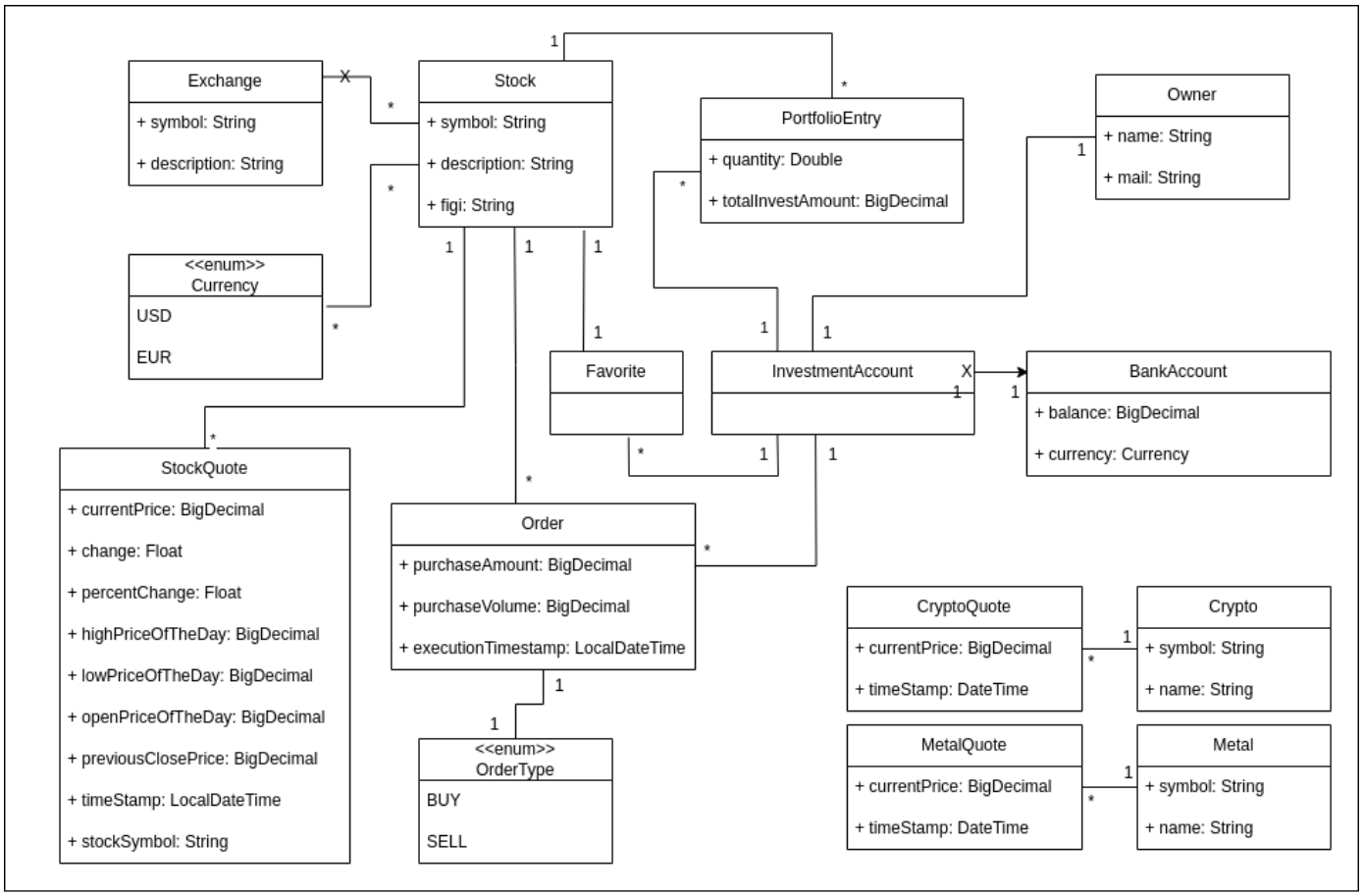
5. Benutzerverwaltung (Mehrere User)

Zweck: Mehrere Nutzer können das System verwenden, mit jeweils eigenen Depots und individuellen Einstellungen.

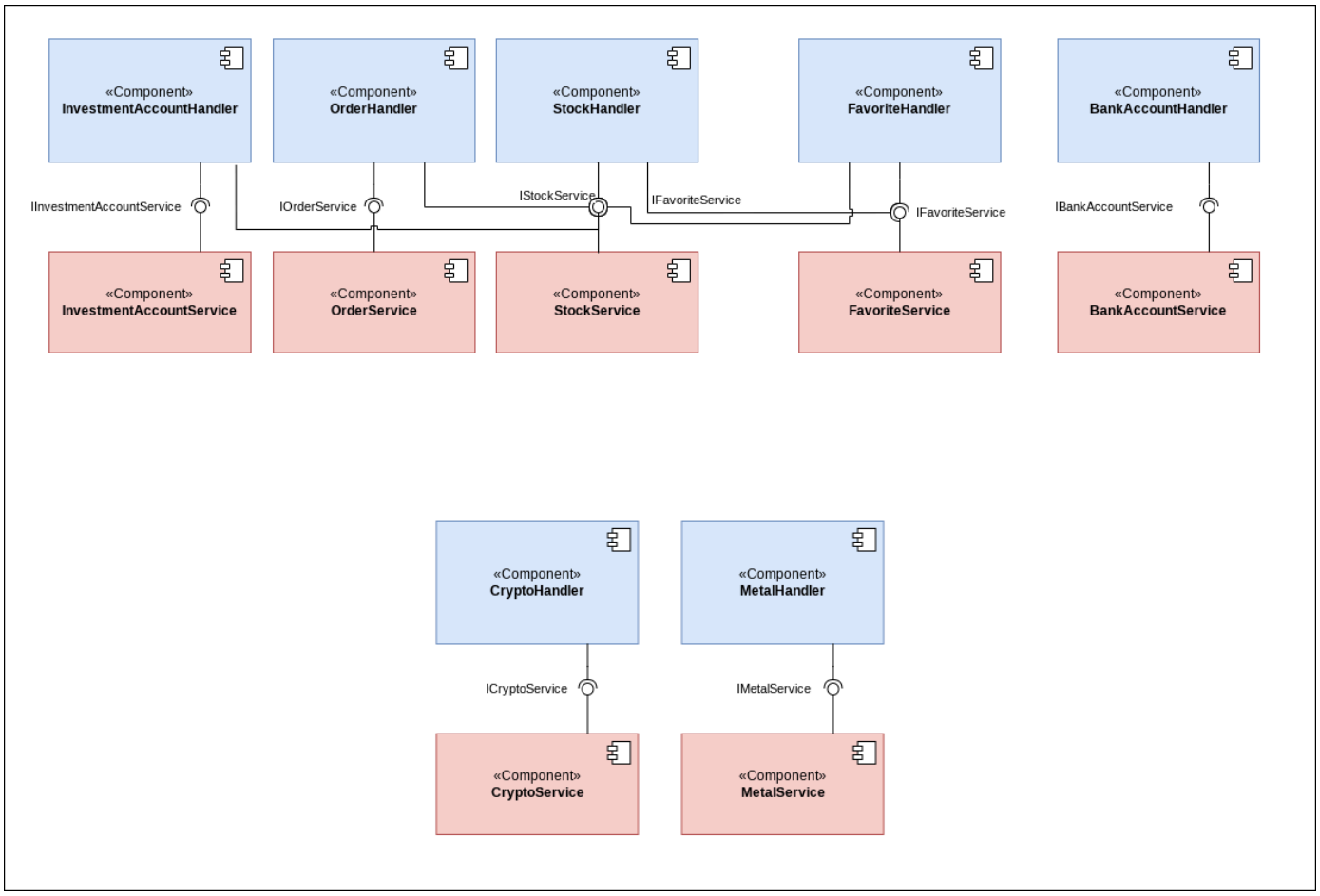
Funktionalität:

- Jeder Nutzer hat ein eigenes Depot und kann eigene Kauf-/Verkaufsaufträge verwalten.
- Möglichkeit, verschiedene Nutzer zu erstellen und zu verwalten.
- Login-Möglichkeit

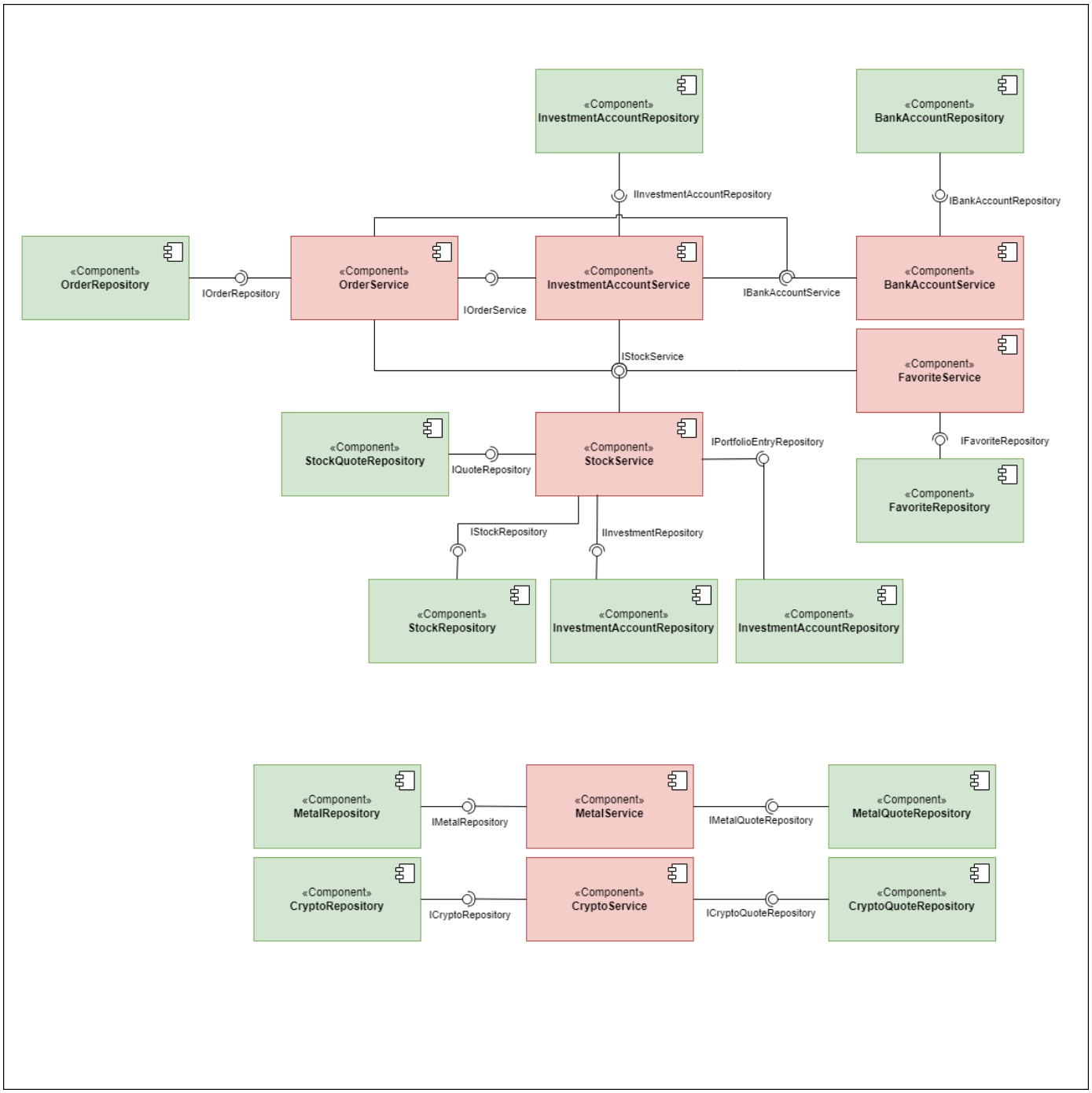
Domänenmodell



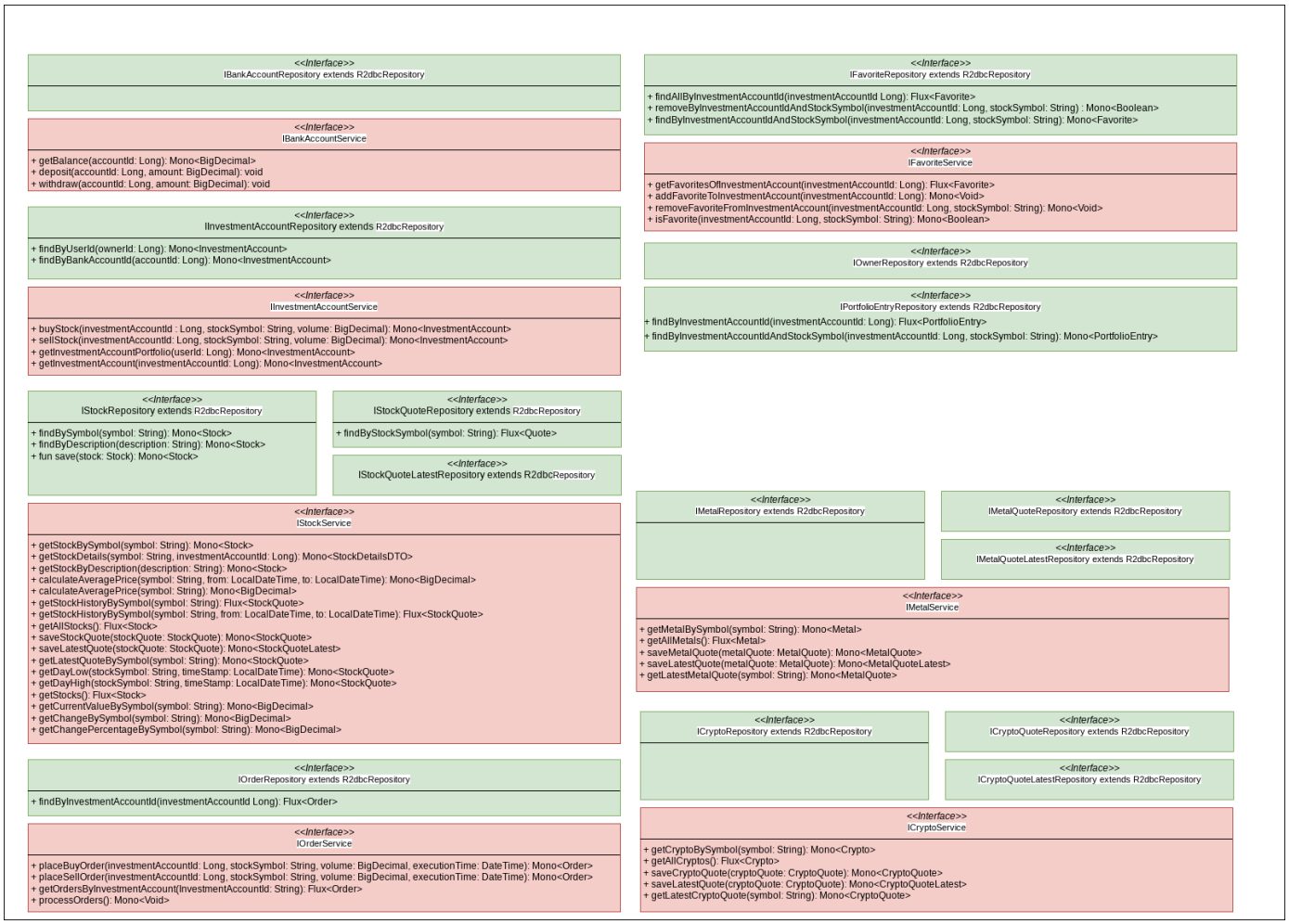
Komponentenmodell (Handler-Services)



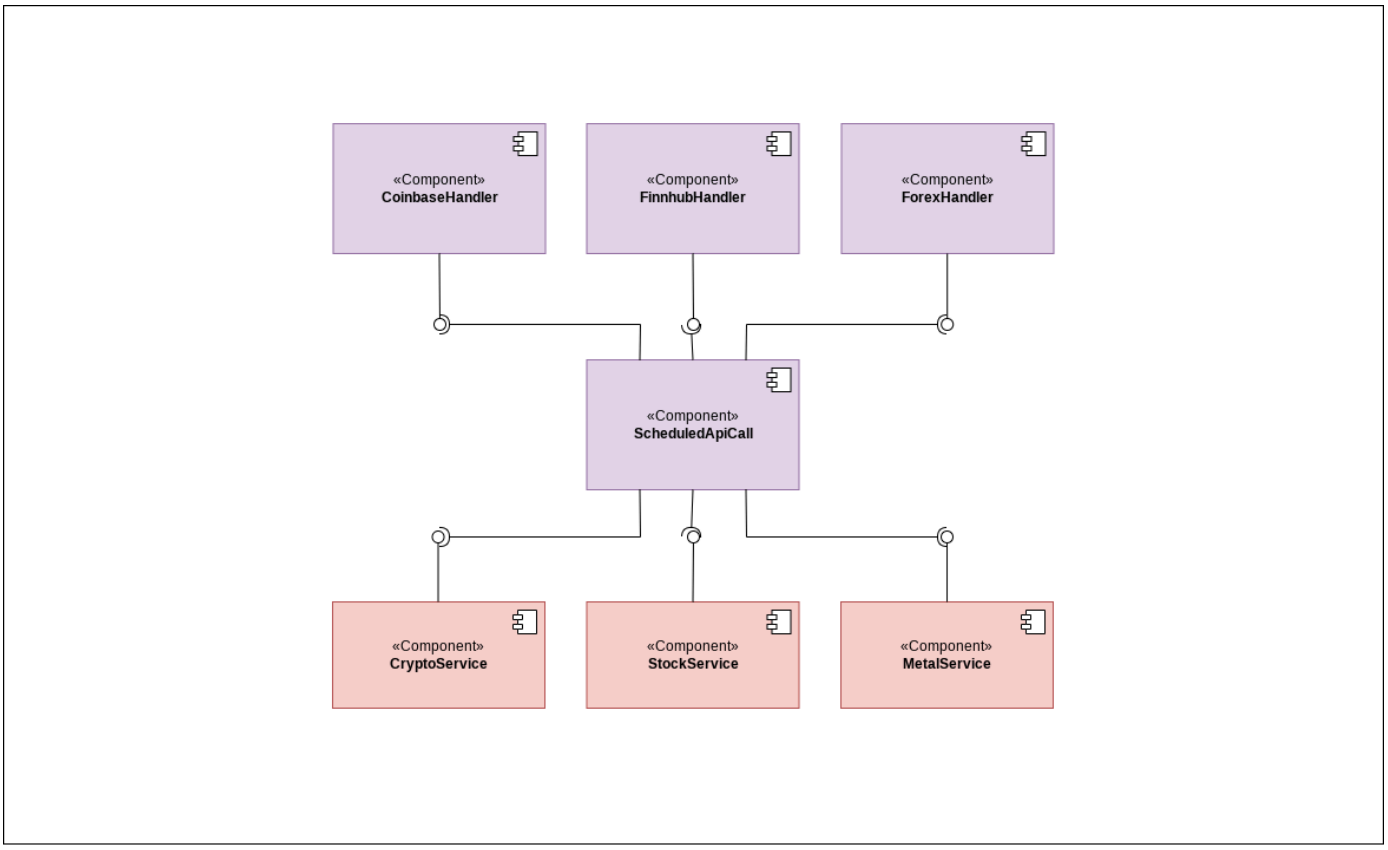
Komponentenmodell (Service-Repositories)



Interfacedefinitionen



Komponentenmodell (Scheduled API Call)



Als CRUD Repository verwenden wir R2DBC.

Tech-Stack

Frontend:

- Vue.js
- TypeScript

Backend:

- Kotlin
- SpringWebFlux

Datenhaltung:

- R2DBC
- Postgres
- Docker