**Anwendungsanleitung - Testdaten:**

Damit das System getestet werden kann ist wichtig zu wissen welche Daten sich bereits im System befinden. Dabei wird nicht auf die Daten aus der Influx-DB eingegangen, da diese wie in Abschnitt X beschrieben ist, dynamisch von einer externen Quelle abgefragt werden. Des Weiteren sind diese Testdaten an sich nicht für das testen relevant.

Die für das testen relevantesten Daten die die verfügbaren User bzw. deren Login Daten. Es sind dazu zwei User verfügbar. Als diese kann man sich mit den Benutzernamen „testuser1“ und „testuser2“, sowie bei beiden dem Passwort „securepassword“ einloggen. testuser1 besitzt dabei initial 10 und testuser2 12 IBM Aktien. Auf dem bankkonto des testuser1 befinden sich anfangs 2000,99$, auf dem des testuser2 5000$.

In Abschnitt X wurde zuvor einige weitere Attribute aufgeführt, die zum Betreiben des Systems von Nöten sind. Wie diese Testdaten aussehen ist für das Testen an sich nicht für Bedeutung und kann, wenn benötigt mittels der phpmyadmin-Suite nachgeschaut werden. Die Login Daten dafür sind: Server: „mariadb“, Username: „secureuser“, Password: „securepassword“.

Weitergehend ist beim Testen nichts Besonderes zu beachten. Es wurde versucht das System so einzugrenzen, dass der User verständliche Meldungen bekommt, sollte er „Fehler“ machen, so dass durch die direkte Nutzung des Systems an sich keine Fehler im System entstehen können. Auch wurde darauf geachtet, dass die Oberfläche einfach gehalten und damit selbsterklärend ist und so eine hohe Usability aufweist.

**Einführung in die Komponenten - Datenbankenstruktur:**

Um die vom System benötigten Daten zu speichern haben wir uns dazu entschieden dies in Datenbanken zu tun, da dies unter dem Gesichtspunkt eines Verteilten Systems am meisten hergibt. Außerdem könnte so eine zentrale Speicherung möglich gemacht werden so dass es wäre möglich sich unabhängig von dem Endgerät in das System einzuloggen und mit dem gleichen Datenstand weiterarbeitet wie man vorher hatte. Auf die verschiedenen Datenbanken und deren Aufbau und warum wir genau diese verwendet haben, wird in diesem Abschnitt genauer eingegangen.

Es wurden zwei Arten an Datenbaken verwendet. Zum einen altbewährte relationale Datenbanken und zum anderen eine Zeitreihendatenbank (Time Series Database). Für die ‚normalen‘ Daten, also mit gewöhnlichen Daten ohne komplexen Zusammenhang, oder zeitlichen Reihenfolge werden die relationalen Datenbanken benutzt. Von diesen wurde zwei aufgesetzt. Eine ist für den Großteil der Daten die direkt zum System gehören, die zweite für die simulierte Anbindung an die Bank. Diese wurden mit MariaDB umgesetzt. Direkt für das System wurden folgende Daten benötigt. Zum einen die User mit Username, Passwort und Kontonummer der Bank. Außerdem hat jeder User ein Depot in den die Aktien als Symbol und dessen Anzahl gespeichert werden müssen. Da in unserem System nur bestimmte Aktien gehandelt werden können, müssen diese mit dessen Name und dem Symbol gespeichert werden. Außerdem soll jeder kauf und jeder Verkauf festgehalten werden. Dies wurde in den Tabellen aus Abbildung 1 umgesetzt.

Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung

Abbildung : DB-Tabellenstruktur IMS

Die Daten der simulierten Bank umfassen die Kontonummer, den Namen, das Passwort und den Kontostand des Besitzers. Dabei ist ein Passwort und ein Name in diesem System nicht von Nöten, es wurde aber der Vollständigkeitshalber mit aufgenommen. Wie dies umgesetzt wurde ist in Abbildung 2 zu sehen.

![Ein Bild, das Screenshot enthält.

Automatisch generierte Beschreibung]()

Abbildung : DB-Tabellenstruktur Bank

Der kleinere Teil der Daten die direkt von unserem System benötigt werden, wurden in der benannten Zeitreihendatenbank gespeichert, da dieser Teil der Daten von der zeitabhängig ist. Es handelt sich also um Datensätze, die immer wieder die gleichen Attribute haben aber zudem immer von der Zeit abhängig sind. Da diese Eigenschaft eine Zeitreihendatenbank perfekt unterstützt und weil wir dieses System noch nicht kannten und dadurch viel zu lernen erwarteten wurde sich dafür entschieden. Außerdem hat es dazu beigetragen die Aufgabenstellung, ein verteiltes System aufzubauen zusätzlich beigetragen, da damit ein weiteres anders System hinzukam. Umgesetzt wurde diese Datenbank mit InfluxDB. In der InfluxDb wurden zwei Arten von Datensätzen gespeichert, also zwei Tabellen angelegt. Beide dienen dazu die Stände der Aktien zu einem bestimmten Zeitpunkt zu speichern um daraus den aktuellen Wert, die Veränderung und einen Grafen lesen zu können. In der ersten Tabelle werden die aktuellen untertägigen Daten, also die Daten von diesem Tag gespeichert. In der zweiten werden die Daten des jeweiligen Tages gespeichert. Dabei haben beiden den gleichen Aufbau. Jeder Datensatz hat folgende Daten unter diesen Attributen gespeichert: Close – mit welchem Aktienwert wurde der Datensatz versendet; Open – mit welchem Aktienwert wurde an diesem Tag eingestiegen; High – Höchster Wert in diesem Zeitraum; Low – Niedrigster Wert in diesem Zeitraum; Symbol – Kurzzeichen der Aktie; Timezone – In welcher Zeitzone wird diese Aktien gehandelt.

**Aufbau der Komponenten – IMS-Server:**

Der IMS-Server ist die zentrale Komponente unseres Systems in der alle Komponenten Anschluss finden und die Inforationen zusammengeführt und weitergegeben werden.

Dabei ist immer eine Anfrage vom Client der Auslöser für Datenaustausch und Logikabläufe. Die Anfragen werden mittels eines Posts an verschiedene Pfade der Adresse gestellt und lösen den gewünschten Prozess aus woraus Daten am Ende zurückgegeben werden. Die Daten der InfluxDB werden mithilfe der Datadelivery Komponente (datadelivery.js) unabhängig von dem User regelmäßig aktualisiert. Darauf wird jedoch genauer bei der Beschreibung der Komponente in Abschnitt X eingegangen.

Der Logik des IMS-Servers befindet sich vollständig in der „imsserver.js“. Diese besteht aus zwei wesentlichen Bestandteilen. Zum einen der Logik zum Starten und aufbauen des lauffähigen Servers und die Logik für die oben benannten Anfragen. Wird das System gestartet und der Server hochgefahren wird ein Listener auf die angegebene Adresse (0.0.0.0) und Port (8080) gesetzt, wodurch es möglich wird Anfragen gegen den Server zu senden. Anschließend wird eine Verbindung zu den verwendeten Datenbanken aufgebaut und ggf. die in Abschnitt X beschriebenen Testdaten angelegt.

Es ist möglich vier verschiedene Anfragen gegen den Server zu senden, um Daten anzufragen oder in das System zu gegeben. Die „/transaction“ Anfrage beinhaltet die Logik für einen Kauf oder einen Verkauf einer Aktie. Dabei werden die nötigen Prüfungen, z.B. eine gegen den Bank-Server über die Höhe des Kontostandes getätigt und sichergestellt das das System konsistente Informationen hat. Sollte es z.B. nicht möglich sein die Aktienanzahl bei einem Verkauf zu aktualisieren wird auch der Auftrag einer Gutschrift auf das Konto nicht abgeschickt. Aufträge gegen den Bank-Server werden dabei mit Hilfe von Axios genau wie bei einer Anfrage gegen den IMS-Server gestellt. Mit „/login“ wird bei einem Login verschiedene Daten aus der Datenbank bezüglich des Users der sich anzumelden versucht abgefragt. Die Logik ob der Login erfolgreich ist wird beim Client mit den erfragten Daten durchgeführt. Diese Vorgehensweise ist nicht sicher und vollkommen realitätsfremd. Wir haben uns jedoch einfachheitshalber für diese Umsetzung entschieden, da diese Logik nicht essentiell für die Funktionalität des Systems ist, nicht direkt zum System gehört und sonst andere essentielle Funktionen des Systems zu kurz gekommen wären. Die „/fetch\_...“ Anfragen bedienen Abfrage für Daten bezüglich der Aktien. „/fetch\_depoinhalt“ gibt für den angefragten Nutzer alle Aktien mit Anzahl zurück, die sich im Depot dessen befinden. Die „/fetch\_data“ Anfrage liefert alle benötigten Daten, die für die Erstellung der Grafen für die einzelnen Aktien und die allgemeine Auflistung auf der Übersichtsseite benötigt werden.

Die Übergabe basiert bei allen Anfragen auf JSON Datenstrukturen.