Duale Hochschule Baden-Württemberg Mannheim

Studienarbeit

Projektrealisierung - Simulation

Studiengang Wirtschaftsinformatik

Studienrichtung Software Engineering

Kurs: WWI18SEA/C

Aaron Schweig
Troy Kessler
Matthias Vonend
Michael Angermeier
Patrick Mischka
Jan Grübener

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation und Zielsetzung

Im Rahmen der Vorlesung „Projekt“ des WWI18SEAC Kurses ist eine fiktionale Börse exemplarisch als Projekt mit unterschiedlichen Projektteams realisiert worden. Die verschiedenen Funktionen des Wertpapierhandels lagen im Verantwortungsbereich von unterschiedlichen Teams. Ein Vorlesungsfokus stellte die Koordination und Integration der Gruppen untereinander dar, wobei auch auf fachliche Korrektheit der behandelten Sachverhalte geachtet wurde.
Diese Arbeit stellt die Tätigkeiten des „Simulation“-Teams dar. Das Ziel der Simulation war, realistische Marktsituationen für unterschiedliche Wertpapiere darzustellen. Simulierte Kursverläufe sollten realitätsnah verschiedene Ordertypen widerspiegeln und Preisentwicklungen mit verschiedenen Ausmaßen beinhalten.
Andere Projektteams realisierten die Funktionalitäten der zentralen Börse, eines Privatbrokers und eines Geschäftsbrokers. Die simulierten Kurse reagierten auf den Handel von unterschiedlichen Handelsteilnehmern, was für ein realistisches Marktverhalten notwendig war.
Als Grundlage für Kursentwicklungen dienten Szenarien, die dem Kursverlauf der SAP-Aktie aus den letzten zwei Jahren entnommen wurden. Hierbei wurde auf Kursdaten der NASDAQ-Börse zurückgegriffen, da diese umfänglich und kostenfrei zur Verfügung stehen. Bei den deutschen Handelsplätzen war dies im Gegensatz nicht der Fall. Die entnommenen Kursverläufe orientierten sich an unterschiedlichen Ereignissen, womit ein breites Spektrum an Marktverhalten abgedeckt werden konnte.
Das Ziel der Vorlesung war, die Zusammenarbeit von unterschiedlichen Teams innerhalb eines Projekts, die auf ein gemeinsames Projektergebnis hinarbeiten, zu vermitteln und grundlegende Verhaltensweisen aufzuzeigen. Das Ziel des Projektes selbst war, unter Beachtung der fachlichen Korrektheit des Börsenhandels, eine funktionierende Börsenplattform zu realisieren.

1.2 Aufbau der Arbeit

Die Arbeit beginnt mit den theoretischen Ansätzen zur Erstellung der unterschiedlichen Kursverläufe. Erste Überlegungen beschäftigten sich mit dem Fama-French-Dreifaktorenmodell. Bei diesem Modell wird eine Aktienrendite vorhergesagt, welche mithilfe von Marktfaktoren und unterschiedlichen Unternehmens- und Kurseigenschaften berechnet wird. Die Handlungen eines Marktes könnten einer solchen Aktienrendite folgen.
Diese Renditeberechnung war im Rahmen der Simulation schwierig umzusetzen, wodurch die Entscheidung gefallen ist, diesen Ansatz zu verwerfen. Das Fama-French-Dreifaktorenmodell benötigt zur Berechnung den Unternehmensbuchwert, welcher über die Bilanz von Aktiengesellschaften auszuweisen ist. Diesen Wert realistisch zu simulieren stellte die größte Hürde bei einer Verwendung des Fama-French-Modells dar.
Als einfacher umzusetzende Alternative ist nun ein Ansatz verfolgt worden, welcher auf Szenarien beruht und mit vordefinierten Daten arbeitet. Historische Kursdaten sind der NASDAQ-Börse entnommen und die ausgewählten Markttage orientieren sich an unterschiedlichen realen Marktsituationen. Diese variieren von einem verkürzten Handelstag (an Heiligabend wird halbtags gehandelt) bis hin zu einer desaströsen Bilanzveröffentlichung, durch welche ein Massenverkauf und ein damit einhergehender Kurseinbruch ausgelöst wird.

Im darauf folgenden Kapitel werden aus den bereits genannten thematischen Rahmenbedingungen Anforderungen an die Simulation abgeleitet. Diese sind zwischen funktionalen und nicht-funktionalen Anforderungen zu unterscheiden. Folglich werden die theoretischen Ansätze in Verbindung mit den herausgearbeiteten Anforderungen kritisch betrachtet und gegebenenfalls eingeschränkt. Als Abschluss dieses Kapitels sind Anforderungen an andere Projektteams dokumentiert.

Im Rahmen der Konzeption wird die technischen Umsetzung näher erläutert und konkrete Vorgehensweisen dargestellt. Als Grundlage für die Konzeption dienten die bereits dokumentierten Anforderungen aus dem vorherigen Kapitel. Die entsprechende Konzeption sollte funktionale und nicht-funktionale Anforderungen realisieren und Schnittstellen beinhalten, um die Inhalte von Anforderungen an andere Projektteams integrieren zu können. Hauptbestandteile der Konzeption sind das Vorgehen bei unterschiedlichen Szenarien und wie aus diesen Szenarien konkrete Orders abgeleitet werden. Des Weiteren wird konkret Bezug darauf genommen, wie der Einfluss von anderen Marktteilnehmern die Orders des simulierten Marktes beeinflussen.

Darauffolgend ist die konkrete Implementierung dokumentiert, wobei die verwendeten Schnittstellen eine besondere Bedeutung haben.
Als inhaltlich letztes Kapitel ist das Nutzerhandbuch ein Anleitung zur Verwendung des entwickelten Codes, um die gezeigten Funktionalitäten nachvollziehen zu können. Hiermit soll die Simulation in Verbindung mit den anderen Projektbestandteilen auch von Dritten rekonstruiert werden können.

Das abschließende Kapitel dieser Dokumentation stellt die Zusammenfassung dar, die einen Überblick über die erbrachten Projektteamleistungen geben soll und die Projektergebnisse kritisch hinterfragt.

2 Theoretische Herleitungen

2.1 Betrachtung von Marktfaktoren

Der erste verfolgte Ansatz, um einen realistischen Aktienmarkt zu simulieren, stellte das Fama-French-Dreifaktorenmodell dar. Bei diesem Modell ist die individuelle Wertpapierrendite, sowohl von individuellen Marktfaktoren, als auch von Marktfaktoren, die auf den gesamten betrachteten Markt wirken, erklärt.
Dieses Modell hat sich im Rahmen der Simulation als ungeeignet erwiesen, jedoch sind anhand der verwendeten Marktfaktoren einige Marktentwicklungen zu erklären, die entsprechend die Wertentwicklung eines Wertpapiers beeinflussen.
Darauffolgend ist die Simulation mithilfe von praxisnahen Szenarien durchgeführt worden, welche im folgenden Kapitel näher dargestellt sind. An dieser Stelle sei darauf hingewiesen, dass bei der Erklärung der unterschiedlichen Szenarien teilweise auf das Fama-French-Dreifaktorenmodells referenziert und die Szenarien damit begründet werden.

Das Fama-French-Dreifaktorenmodell erklärt die Aktienrendite (r_i) anhand von den Aktien-individuellen Marktfaktoren:

- Aktienmarktrisikoprämie : (R_M)
- Größe des Unternehmens : (SMB)
- Buchwert-Kurs-Verhältnis : (HML)
- Risikofreier Zinssatz : (r_F)

Bestimmung der potentiellen Aktienrendite erfolgt dem (vereinfachten) Modell nach mit folgender Formel:

$$r_i = r_F + R_M + \beta_{iSMB} \cdot SMB + \beta_{iHML} \cdot HML$$

(r_F), (R_M), (β_{iSMB}), (β_{iHML}) sind Faktoren, die auf den gesamten Markt wirken. Je nach Marktentwicklung und -situation können die β-Werte positiv oder negativ sein.

Der risikofreie Zinssatz (r_F) ist in Europa zu Zeiten der Nullzinspolitik der EZB faktisch zu vernachlässigen.

Die Aktienmarktrisikoprämie (R_M) ist abhängig von den vorherrschenden Risiken, die auf den Markt wirken.

Ein positives (β_{iSMB}) wirkt sich bei einer kleinen Marktkapitalisierung positiv auf den Aktienkurs aus, eine große Marktkapitalisierung entsprechend negativ. Analog ist dieser Effekt bei einem negativen (β_{iSMB}) gegensätzlich.

Ein positives (β_{iHML}) wirkt sich bei einem steigenden Kurs-Buchwert-Verhältnis positiv aus, bei einem sinkenden entsprechend negativ. Analog ist dieser Effekt bei einem negativen (β_{iHML}) gegensätzlich.

Hierbei ist hervorzuheben, dass sich die Aktienrendite über beide Faktoren (SMB : Marktkapitalisierung = Aktienkurs · Umlaufende Aktien) und
(HML : Kurs-Buchwert-Verhältnis = $\frac{\text{Kurswert der Aktie}}{\text{Buchwert der Aktie}}$) auf den Aktienkurs bezieht. Daher könnte hierbei für jeden individuellen Aktienkurs eine potentielle Aktienrendite errechnet werden, worauf Kauf- oder Verkaufsentscheidungen gegründet sein können. Jedoch können diese Entscheidungen nicht konsistent die Marktentwicklung simulieren, da sich der Markt in vielen Aspekten (besonders in außergewöhnlichen Situationen, z.B. einer weltweiten Pandemie) unberechenbar verhält und eine umfassendere Simulation nicht im Rahmen dieses Projekts umsetzbar gewesen wäre.
Aus diesen Erkenntnissen ist die Entscheidung gefallen, vergangene Kurstage eines DAX-Unternehmens zu betrachten, welche exemplarisch spezifische Situationen darstellen sollen.

2.2 Unterschiedliche Szenarien

Um eine möglichst realistische Simulation des Wertverlaufs einer Aktie darzustellen wurden fünf verschiedene Markttage der SAP-Aktie aus den letzten zwei Jahren betrachtet. Die Entwicklung an diesen Tagen wurde als Vorlage für die Simulation genommen.

Folgend sind die fünf ausgewählten Markttage aufgeführt, wobei jeweils die Begründung für den explizit gewählten Tag gegeben ist.

1. Rapider Kursfall nach Firmenübernahme : 26.10.2020

   Dieses Szenario ist einem Kurstag entnommen, bei welchem der Markt auf die große Übernahme reagierte. Hierbei handelte es sich um einen Montag, nachdem die Information der Übernahme freitags zuvor nach Handelsschluss öffentlich gemacht worden war. Bei einer solchen großen Übernahme wird der Buchwert einer Aktie negativ beeinflusst, was den nachhaltigen Einbruch des Aktienkurses erklären könnte. Dies ist ebenfalls nach dem Fama-French-Dreifaktorenmodell plausibel, wenn das (β_{iHML}) einen negativen Wert annimmt (und die Aktienrendite konstant bleibt).

2. Geringes gehandeltes Volumen : 01.10.2020

   Innerhalb dieses Markttages wurden besonders wenig Aktienanteile gehandelt, wobei auch der Aktienkurs relativ konstant verlief. In diesem Szenario der Simulation hätten einzelne Akteure den Preis der Aktien massiv beeinflussen können.

3. Hohes gehandeltes Volumen : 19.06.2020

   Dieser betrachtete Markttag stellt das Gegenteil zu dem vorherigen Markttag dar. Viele Käufer und Verkäufer handelten, weswegen einzelne Akteure keinen wesentlichen Einfluss auf den Kurswert ausüben konnten.

4. Positive Nachricht : 24.04.2019

   Innerhalb dieses Markttages sind besser ausgefallene Gewinne aus dem vorherigen Quartal veröffentlicht worden, was zu einem rasanten Anstieg des Kurswerts an diesem Tag führte. Damit verringerte sich das Kurs-Buchwert-Verhältnis, wobei - gleichbedeutend mit einem negativen (β_{iHML}) - der Aktienkurs (bei gleicher Aktienrendite) stieg.

5. Normaler Handelstag mit durchschnittlichem Volumen : 22.10.2019

   Dieser Tag wurde als letztes Szenario betrachtet, um einen möglichst durchschnittlich verlaufenden Markttag darstellen zu können.

3 Anforderungsanalyse

In diesem Kapitel werden zuerst die funktionalen und nicht-funktionalen Anforderungen der Simulation erläutert. Anschließend werden diese Anforderungen übersichtlich zusammengefasst. Im Anschluss werden kurz die Anforderungen aus unserer Sicht zu den anderen Teilprojekten erläutert.

3.1 Funktionale Anforderungen

- Anmeldung:
  Damit das Marktgeschehen nicht durch jede beliebige Person beeinflusst werden kann, soll die Anwendung durch ein Login geschützt werden. So können nur dedizierte Personen Szenarien auswählen und wichtige Marktfaktoren beeinflussen. Denkbar wäre hierdurch auch das Verteilen von Berechtigungen. Beispielsweise wäre es möglich, lesende Operationen wie das gehandelte Volumen für alle einsehbar zu machen, während schreibende Operationen, die das Marktgeschehen beeinflussen könnten, nur ausgewählte Personen tätigen können.
- Auswahl vordefinierter Szenarien:
  Es soll eine Liste an vordefinierten Szenarien geben. Diese Szenarien können bestimmte Ereignisse sein, wie der Rücktritt einzelner Vorstandsmitglieder, auf die der Markt in einem vorher definierten Rahmen reagiert. Diese Reaktion soll automatisch mit dem Beginn des nächsten Szenarios durch die Simulation auf den Markt übertragen werden.
- Übersicht über die gerade gehandelten Trades:
  Es sollte eine grafische Übersicht geben, die einen Überblick über das durch die Simulation gehandelten Volumina und Preise verschafft. So soll ein besserer Einblick über die aktuelle Simulationsaktivität gegeben werden.
- Kommunikation von Events:
  Bestimmte Ereignisse beeinflussen das Marktgeschehen. Damit Privatkunden und Unternehmen, die an der Börse handeln, entsprechend reagieren können, müssen sie

über solche Ereignisse aktiv informiert werden. Ob und wie sie auf diese Nachrichten reagieren bleibt ihnen selbst überlassen.

- Reaktion auf Marktgeschehen durch Privat- und Unternehmensbroker:
  Um die Simulation möglichst realistisch zu gestalten, muss auch das Verhalten und die Trades der anderen Marktteilnehmer berücksichtigt werden. Wichtig ist hier jedoch, dass nicht eine einzelne Transaktion eines Privatkunden schon die Marktpreise zum Schwanken bringt. Stattdessen sollen Transaktionen erst eine kritische Masse erreichen müssen, um wirklich auf dem Markt ins Gewicht zu fallen.
- Anpassung der Dauer eines Szenarien:
  Um das schnelle Testen von Anlagestrategien zu ermöglichen, soll die Länge der Szenarien skalierbar sein.

3.2 Nicht-Funktionale Anforderungen

- Geschwindigkeit:
  Für Nutzer ist die Geschwindigkeit besonders wichtig. Lang ladende Anwendungen sind nervig in der Benutzung und demotivieren die Nutzung der Anwendung. Über lange Ladezeiten beschweren sich in einer Umfrage mehr als 70%, wobei 25% aller Nutzer bereits bei 4 Sekunden die Seite verlassen. Aus diesem Grund sollte die Ladezeit gering gehalten werden.
- Skalierbarkeit:
  Um das bereits angesprochene hohe Ordervolumen zu realisieren, muss die Anwendung gut skalieren können. Bei zunehmender Zahl der Transaktionen oder Wertpapieren darf die Reaktion und die Performance der Anwendung nicht beeinträchtigt werden.

3.3 Zusammenfassung

Nr.	Beschreibung
F1	Am System soll eine Anmeldung möglich sein
F2	Es soll eine Szenarienauswahl geben
F3	Nutzer sollen eine Übersicht über die aktuelle Simulationsaktivität bekommen
F4	Events, die das Marktgeschehen beeinflussen sollen aktiv kommuniziert werden
F5	Es soll auf das Marktgeschehen durch andere Privat- und Unternehmensbroker reagiert werden
F6	Die Zeiteinheiten und somit die Dauer eines Szenarien sollte angepasst werden können
NF1	Anwendung lädt schnell
NF2	Die Anwendung ist skalierbar

Tabelle 3.1: Zusammengefasste Anforderungen

3.4 Anforderungen an andere Teams

Im Falle der Simulation bestehen keine Abhängigkeiten zu anderen Brokern. Bei den Privatkunden und Unternehmenskunden handelt es sich auch um Broker. Diese kaufen und verkaufen an der Börse Wertpapiere. Genauso wie bei der Simulation handelt es sich damit um Broker bzw. Clients der Börse, die sich lediglich in ihrer Zielgruppe und dem Zweck unterscheiden. Somit bleibt als einzige Schnittelle die Börse. Dabei wurden folgende Anforderungen aufgestellt:

GET-Requests

- aktueller Preis eines Wertpapiers
- im Handel befindliche Wertpapiere

POST-Requests

- Order einstellen

Aktive Benachrichtigungen

- Buchung der Order
- Handel unterbrochen

4 Konzeption

4.1 Herleitung und Betrachtung verschiedener Szenarien

Damit mit der Simulation der Markt möglich genau widergespiegelt werden kann, steht zunächst die Frage offen, wie ein typischer Handelstag an der Börse aussieht. Um dies analysieren zu können, wird ein Aktienkurs betrachetet. Ausgewählt wurde die SAP Aktie, da diese in den letzten Jahren meistens sehr stabil war. Nachdem es bei einer Aktie unterschiedliche Szenarien geben kann, wurden die fünf häufigsten Szenarien definiert:

- Normaler Handelstags mit einem durchschnittlichen Handelsvolumen
- Positive Nachricht an einem Handelstag
- Negative Nachricht an einem Handelstag
- Niedriges Handelsvomlumen
- Hohes Handelsvolumen

Für das erste Szenario wurde der 22.10.2019 ausgewählt. An diesem Tag gab es zwischen dem Open-Preis und dem Close-Preis eine Preisdifferenz von knappen 3$. Außerdem wurden etwa 1.000.000 Aktien an der NASDAQ-Börse gehandelt.
Am 24.04.2019 wurde die Bilanz des ersten Quartals veröffentlicht. Diese sind deutlich besser ausgefallen als damals erwartet, wodurch es eine deutliche Auswirkung auf den Aktienmarkt gab. Die Aktie hatte einen Unterschied von 8$ zwischen Open und Close. Gleichzeitig wurden knapp 4.700.000 Aktien gehandelt.
Das dritte Szenario ist der erste Handelstag nach der Veröffentlichung der Quartalszahlen (26.10.2020). Aus diesen Zahlen war abzulesen, dass es einen deutlichen Rückgang des Gewinnes gab. Aus diesem Grund ist der Aktienkurs von 149$ (Freitag Close) auf 115$ (Montag Close) gefallen. Zusätzlich gab es ein sehr hohes Handelsvolumen von 11.000.000 Aktien.
Als Beispieltag für ein gering gehandeltes Volumen wurde Weihnachten 2019 ausgewählt. An diesem Tag wurden nur 117.000 Aktien gehandelt. Die Preisdifferenz von Open und

Close betrug lediglich 1$.
In den letzten Jahren war der Handelstag mit dem größten Handelsvolumen der 26.10.2020. Um für zwei Szenarien nicht den gleichen Tag zu modellieren, wurde der 28.10.2020 ausgewählt. An diesem Tag lag das Handelsvolumen bei 5.500.000 Aktien.
Die Daten für alle Szenarien sind von der NASDAQ-Börse aus New York und wurden auf Minutenbasis gespeichert.

4.2 Umwandlung von Szenarien in Orders

Nachdem die Daten exportiert wurden und in JSON-Format abgespeichert wurden, müssen diese noch angepasst werden. Die Szenarien sollen später für verschiedene Aktien verwendbar sein. Um dies zu ermöglichen, werden die Aktienkurse nicht in absoluten Zahlen gespeichert, sondern als Änderungsrate. Die Änderungsrate ist die Änderung des Aktienkurses im Vergleich zum Aktienkurs einer Minute davor. Dadurch ist es möglich, dass das Szenario mit beliebigen Aktien durchgespielt werden kann.

Folgender Abschnitt zeigt die Struktur der von der NASDAQ-Börse erhaltenen Daten:

```
{
  "time": "2020-10-28 19:58:00",
  "open": 108.5,
  "high": 108.5,
  "low": 108.5,
  "close": 108.5,
  "volume": 391
},
{
  "time": "2020-10-28 19:54:00",
  "open": 108.5,
  "high": 108.5,
  "low": 108.5,
  "close": 108.5,
  "volume": 205
},
```

../backend/src/assets/szenarios/3.Fall_SAP_28.10.20_Hohes_gehandeltes_Volumen.json

Nachfolgender Ausschnitt zeigt die von uns transformierte JSON Datei:

```
{
  "time": "2020-10-28 19:54:00",
  "volume": 205,
  "delta": 0.0
},
{
  "time": "2020-10-28 19:58:00",
  "volume": 391,
  "delta": 0
}
```

../backend/src/assets/szenarios/28.10.20 Hohes gehandeltes Volumen.json

Im nächsten Schritt müssen anhand der Daten Orders erstellt werden, um den Aktienkurs zu simulieren. Dafür muss es jeweils eine Limit-Buy-Order und eine Limit-Sell-Order zum gleichen Preis geben, damit die Orders matchen und es einen neuen Aktienpreis gibt. Der Preis für die Limit-Orders wird aus dem aktuellen Aktienpreis und der Änderungsrate berechnet. Entscheidend ist dabei vor allem das Volumen der durch die Simulation eingestellten Transaktionen. Um die im Orderbuch befindlichen Orders zu bedienen und darüber hinaus mit „eigenen Orders“ den Referenzpreis zu beeinflussen müssen alle im Orderbuch befindlichen Sell und Buy Orders zuerst bedient werden. Aus diesem Grund muss die Simulation genügend Wertpapiere kaufen und verkaufen, damit diese Orders bedient werden können und anschließend den neuen Preis zu bilden. Da durch die anderen Broker sowohl Sell als auch Buy Orders eingestellt werden und nicht abzusehen ist, dass einer der beiden Kaufvorgänge überproportional häufig vertreten ist, werden Sell und Buy Orders von der Simulation in gleichem Verhältnis eingestellt.

Mit diesem Vorgang würde es schon reichen den Aktienkurs eines Szenarios durchzuspielen, denn die beiden Orders würden direkt matchen und den neuen Aktienpreis bilden. Nachdem aber durch die Simulation der gesamte Markt widergespiegelt werden soll, müssen mehr Orders erstellt werden. Da auch andere Marktteilnehmer Aktie handeln, darf es nicht passieren, dass andere Marktteilnehmer den Aktienkurs mit unerwarteten Orders beeinflussen. Damit diese Probleme gelöst werden, müssen immer eine gewisse Anzahl an Orders im Orderbuch stehen. Diese Orders sollten immer nahe am aktuellen Kurs sein, damit andere Marktteilnehmer zu normalen Preisen die Aktie kaufen können und Limit-Orders mit unerwarteten Preisen nicht zum matchen kommen.
Für die Befüllung des Orderbuches werden fünf Buy- und fünf Sell-Limit-Orders jeweils ein paar Prozent über und unter dem aktuellen Preis gesetzt. Dabei wird eine Aufteilung von 80 zu 20 Prozent verwendet. Das heißt, dass 80 Prozent des Ordervolumens in die beiden

matchenden Orders aufgeteilt werden, während die restlichen 20 Prozent für Orders zum Befüllen des Orderbuches benutzt werden.
Da für die Anzahl der Aktien die Originaldatei von der Börse verwendet werden und damit eine sehr hohe Anzahl an Aktien gehandelt werden, ist der Aktienkurs vor privaten Anlegern (Privatbroker) schützt. Durch das kleine Handelsvolumen dieser im Vergleich zur Simulation haben diese keinen großen Einfluss auf den Preis. Dies bedeutet, dass eine Marktmanipulation durch private Anleger mit einer geringen Anzahl an Aktien nicht möglich ist. Bei den Geschäftskunden kann es unter Umständen dazu führen, dass es einen kleinen Kursanstieg bzw. Kursfall verursacht, sofern eine große Menge an Aktien gekauft bzw. verkauft wird. Dafür müssen jedoch genug Aktien gekauft werden, die mehrere Einträge aus dem Orderbuch erfüllen, und damit einen Preissprung erreichen.

4.3 Benachrichtigungen über Marktereignisse

Nachdem bei einer Simulation nicht nur die Daten simuliert werden sollen, sondern ein normaler Handelstag, ist es sinnvoll auch Nachrichten zu dem Unternehmen der Aktie zu veröffentlichen. Dabei können automatisiert Nachrichten über Telegram versendet werden, wenn der Kurs beispielsweise an einer bestimmten Stelle angekommen ist. Für unsere Szenarien wäre zum Beispiel Nachrichten wie: „Unternehmen XY veröffentlicht Quartalszahlen. Gewinneinbruch von 20%!“ sinnvoll. Diese Nachrichten werden dann durch einen Nachrichtenbot versendet. Anschließend kann beobachtet werden, dass der Aktienkurs sinkt. Diese Möglichkeit hat für Marktteilnehmer den Vorteil, dass sie reagieren können und nicht rein spekulativ handeln müssen.

5 Implementierung

5.1 Technologiewauswahl

Im folgenden Abschnitt wird thematisiert, wieso wir uns für die Technologien entschieden haben, die innerhalb des Simulationsprojekts verwendet werden. Da wir hinter den als Gesamtprojekt definierten Schnittstellen sehr unabhängig waren und keine Limitationen durch bereits vorhandenen Code oder Technologien hatten, konnte die Technologiewahl nahezu ohne Einschränkungen stattfinden. Ziel ist es, den Entscheidungsprozess näher zu beleuchten und die finale Wahl zu begründen.

Wahl der Programmiersprache

Eine der grundlegenden Entscheidungen, die es bei jedem Start einer Entwicklungsaufgabe zu treffen gilt, ist die Wahl der Programmiersprache. Dabei spielen mehrere Faktoren eine wichtige Rolle. So gilt es immer das richtige Tool für die Aufgabe zu finden. Diese Suche muss allerdings mit den im Entwicklerteam vorhandenen Fähigkeiten abgeglichen werden, sodass am Ende nicht nur eine Programmiersprache verwendet werden kann, die den Anwendungsfall möglichst optimal lösen kann, sondern auch durch Wissen innerhalb des Teams gestützt ist.

Beginnend mit der Analyse der Anforderungen an die Programmiersprache lässt sich feststellen, dass es keine besonderen Anforderungen bzgl. Multithreading oder sonstigen Paradigmen, wie OOP, gab. Dies ermöglicht eine große Wahlfreiheit und es konnte eine Programmiersprache gewählt werden, bei der das Wissen innerhalb des Teams möglichst umfangreich war.
Dabei fiel die Wahl auf JavaScript, im speziellen NodeJS als serverseitige Laufzeitumgebung. Ein besonderer Vorteil bei NodeJS war für uns das durchgängige Verwenden einer Programmiersprache zwischen der Backend-Anwendung und der Frontend-Anwendung.

Mit der finalen Wahl von TypeScript als Programmiersprache sind wir noch einen zusätzlichen Schritt gegangen und erweiterten JavaScript mit der Fähigkeit des compile-time Typechecking. Das bedeutet Typensicherheit ist gewährleistet und Laufzeitfehler können

minimiert werden. Zusätzlich stellt TypeScript die Grundlage der im Backend und Frontend verwendeten Frameworks dar.

Wahl des Backendframeworks

Mit der Wahl von NestJS zur Implementierung der Backendfunktionalität haben wir ein Framework gewählt, welches in vielen Unternehmen bereits produktiv verwendet wird. Zusätzlich dazu verbindet es die besten Eigenschaften verschiedener populärer Frameworks (z. B. Spring-Boot, Ruby on Rails, Laravel, Symphony) und nutzt die Learnings dieser Frameworks, um selbst ein optimaleres Framework zu sein. Wichtige Eigenschaften sind dabei eine starke Gewichtung des „Separation of Concerns“ Paradigmas. Zusätzlich ermöglicht NestJS ein einfaches System zur Realisierung von Dependency Injection, sodass die Kommunikation zwischen einzelnen Komponenten ohne Probleme möglich ist. Dies vereinfacht die tatsächliche Implementierung und ermöglicht diese jederzeit zu ändern.
Außerdem orientiert sich die Architektur einer Nest-Applikation sehr stark an den Prinzipien einer Angular-Webapplikation, was auch ein ausschlaggebender Faktor für die Wahl des Frontendframeworks war.

Die Geschwindigkeit durch die mithilfe von NestJS eine RESTful-Applikation programmiert werden kann ist sehr hoch. Das Team hatte bereits durch die Realisierung vorheriger Projekte positive Erfahrungen mit dem Framework gesammelt, welches die Entwicklung positiv beeinflussen konnte.

Wahl des Frontendframeworks

Bei der Umsetzung des Frontends kam es uns auch darauf an, viele der bisher getroffen Entscheidungen miteinfließen zu lassen, sodass auch hier ein optimaler Einsatz des Wissens stattfinden kann und eine hohe Entwicklungsgeschwindigkeit (Velocity) erreicht wird. Bei der Wahl eines Frontendframeworks stellt sich oftmals nur die Frage, welches der drei großen Frameworks (VueJS, Angular oder React) am sinnvollsten zur Lösung des Problems beiträgt. In einem einfachem Anwendungsfall, der durch das Simulationsfrontend dargestellt wird, Schnitt keines der genannten Frameworks besser als ein anderes ab. Allerdings, wie bereits zuvor erwähnt, orientiert sich die Architektur von NestJS sehr stark an dem Aufbau von Angular-Applikationen. Unsere Wahl viel daher auf Angular, da somit

eine höchstmögliche Konsistenz zwischen den verschiedenen Komponenten gewahrt werden kann und es jedem Teammitglied möglich ist, mit demselben Know-How sowohl zu Aufgaben im Frontend als auch im Backend beizutragen.

5.2 Architektur

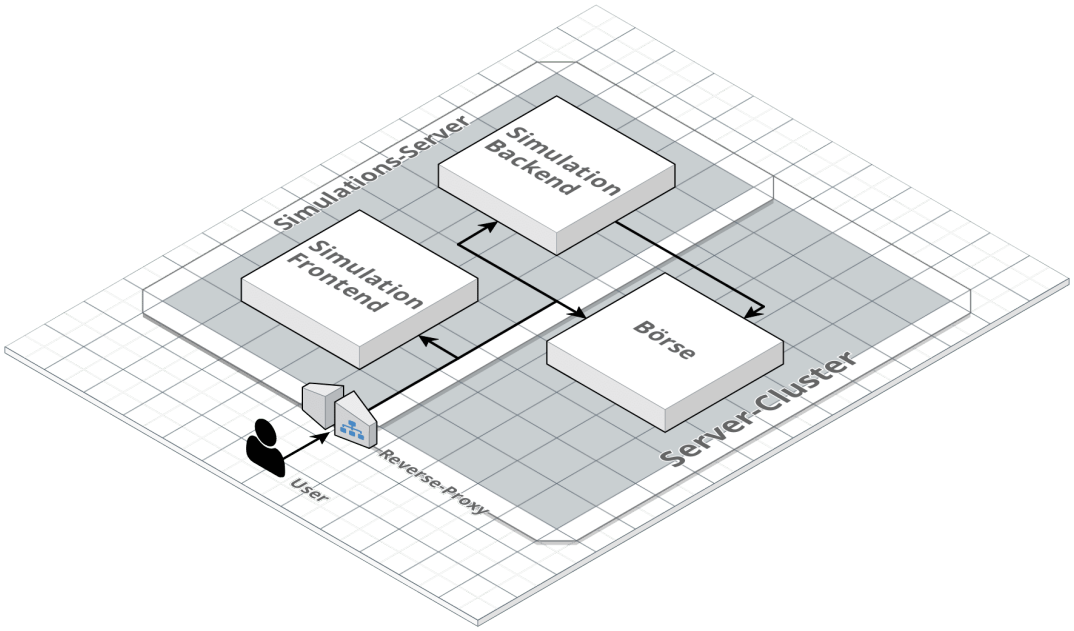

Abbildung 5.1: Architektur

In Abbildung 5.1 ist die Architektur der Anwendung dargestellt. Alle benötigten Server sind in einem Server-Cluster organisiert. Hier werden die Börse, die Kernlogik der Simulation, und das Nutzerinterface betrieben. Zentrales Eintrittstor ist dabei ein Reverse-Proxy, welcher die Aufgabe hat, die Daten an die richtigen internen Server weiterzuleiten und Daten, die für den Nutzer bestimmt sind zu verschlüsseln. Innerhalb des Clusters kommuniziert der Simulations-Backend-Server direkt mit der Börse. Dadurch kann dieser Verkehr nicht von außen beeinflusst werden und mögliche Aktieninformationen (beispielsweise welche Person welche Aktie handelt) sind geschützt.

Das Simulations-Frontend dient als grafische Benutzeroberfläche, mit dem das eigentliche Simulations-Backend gesteuert wird. Weitere Informationen dazu finden sich in Kapitel 6.

Unsere Architektur trennt dabei streng die Szenario-Logik vom Nutzerinterface. Hat der Nutzer ein Szenario ausgewählt, alle Einstellungen getroffen und das Szenario gestartet,

wird eine Anfrage an den Simulations-Backend-Server gesendet. Daraufhin beginnt dieser das entsprechende Szenario auszuführen und mit der Börse zu handeln. Dies hat ein paar Vorteile gegenüber der Implementierung sämtlicher Logik innerhalb des Nutzerinterfaces:

1. Zentrale Nutzersteuerung
   Der Backend-Server kann Ausführungen von Szenarien steuern. So werden beispielsweise Kollisionen vermieden, wenn mehrere Szenarien von verschiedenen Nutzern gestartet werden würden.
2. Code-Qualität
   Durch die Trennung der Logik nach dem Separation of Concern und Clean-Code Methodiken wird der Quelltext wartbarer.
3. Unterbrechungen
   Der Nutzer kann ein Szenario starten, welches eine längere Dauer hat (z. B. mehrere Tage). Anschließend muss er nicht seinen Computer eingeschaltet lassen. Das Szenario wird im Hintergrund ausgeführt.

5.3 Schnittstellen

Die Simulation nutzt verschiedene Schnittstellen und bietet selbst auch einige an. Besonders durch die Aufteilung der Simulation in Frontend- und Backend-Server sind Schnittstellen notwendig. Das Frontend nutzt dabei die Schnittstellen anderer Teams (vgl. Abschnitt 3.4) und des Backend-Servers.

Die Schnittstellen der Börse sind in https://boerse.moonstonks.space/docs/ und die Schnittstellen der Simulation in https://simulation.moonstonks.space/docs/ dokumentiert.

Die Simulation bietet dabei besonders die drei nachfolgenden Schnittstellen an:

- Verfügbare Szenarios
  Gibt alle verfügbaren Szenarios zurück. Jedes Szenario gibt den Namen und die einzelnen Datenpunkte zurück, die bezogen auf die Uhrzeit die Kursveränderungen definieren.
- Szenario starten
  Diese Schnittstelle kann genutzt werden, um ein Szenario zu starten.

- Szenario Status abfragen
  Diese Schnittstelle dient zur Abfrage, wie weit fortgeschritten ein Szenario ist. Hierbei wird ein Prozent-Wert zurückgegeben, der die simulierte Zeit repräsentiert. Sollte kein Szenario im Gange sein, gibt diese Schnittstelle 100 zurück.
- Szenario stoppen
  Mit dieser Schnittstelle kann ein Szenario gestoppt werden. Die Simulation unterbricht alle Orders und geht in einen Leerlauf.

Interne Schnittstellen und Funktionalitäten werden außerdem getestet. Dafür ist eine Pipeline definiert, welche bei jeder Anpassung des Quellcodes diesen auf Fehler überprüft. Die Ergebnisse werden anschließend unter https://stonks2moon.github.io/Simulation/coverage/ publiziert.

5.4 Szenario

Wie in Abschnitt 5.2 beschrieben kümmert sich der Backend Server um die Ausführung der Szenarien.

Im Kern basiert die Szenarioausführung auf einem Agenten-Ansatz. Agenten sind Computerelemente, die sich in bestimmten Umgebungen befinden und in der Lage sind eigenständige Entscheidungen zu treffen. Die Umgebung ist dabei der Aktienmarkt. Da ein hoher Fokus auf eine realitätsnahe Simulation gelegt wird, orientiert sich die Implementierung dieser Agenten an einem Black-Box-Modell. Wie in der Realität haben alle Agenten eigenständige Strategien und können sich jederzeit frei entscheiden wann welche Order erstellt wird. Dafür haben die Agenten ein eigenes, asynchrones Gehirn. Weder andere Agenten noch der Server haben Informationen über das aktuelle Handeln eines einzelnen Agenten.

Sobald die Anfrage durch die API (Application Programming Interface) im Server eingetroffen ist startet der Server direkt mit der Szenarioausführung. Dafür existiert ein Lifecycle, der mehrere Schritte nacheinander durchläuft (vergleiche „Endlicher Automat“):

1. Server Start
   Der Server wurde gestartet. Ab diesem Zeitpunkt kann der Server anfragen entgegennehmen.
2. Szenarios geladen
   Die Szenarios wurden aus den Quelldaten geladen und können nun gestartet werden.

3. Szenario wird gestartet
   Über die Application Programming Interface wurde der Start eines Szenario angefordert und der Server initialisiert das Szenario.

   3.1. Agenten Initialisierung
   Die Agenten werden erstellt und bekommen ihre Logik. Dafür werden neue Instanzen dieser Agenten erstellt. Zusätzlich wird ihnen ein Verhalten zugewiesen, welches sie später ausführen sollen.

   3.2. Markt Initialisierung
   Die Marktüberwachung wird gestartet, sodass die Agenten Informationen über den aktuellen Aktienpreis haben.

   3.3. Daten Initialisierung
   Die Agenten werden mit Daten, z. B. den Szenarieninformationen oder anderen Einstellungen befüllt. Möglich wäre hier auch eine Implementierung von existierenden Depots dieser Agenten.

   3.4. Agenten beleben
   Die Agenten werden belebt, sodass diese mit ihrer Logik beginnen und Orders erstellen.

4. Szenario wird ausgeführt
   Das Szenario wird aktuell ausgeführt und die Agenten arbeiten.

5. Szenario wird gestoppt
   Alle Agenten werden aufgefordert ihr Verhalten zu stoppen und das Szenario wird beendet.

Da dieser Life-cycle bereits einige Schritte hat, ergibt sich eine erhöhte Komplexität, welches die Implementierung verlangsamen könnte. Um dies vorzubeugen wurde der Code so gestaltet, dass es mit sehr geringem Aufwand möglich ist neue Verhalten für Agenten zu implementieren. So gibt es innerhalb des Quellcodes eine zentrale Schnittstelle, die eingebunden wird. Dies ist als eine abstrakte Klasse definiert. Sobald Entwickler diese benutzen, wird von der Entwicklungsumgebung bereits sämtlicher Code generiert. Zusätzlich ist diese Schnittstelle ausführlich innerhalb des Quellcodes dokumentiert, wodurch Entwickler bereits während des Entwickelns über IntelliSense ausführliche Informationen über die richtige Verwendung bekommen.

Für die Szenarioumsetzung ist primär das Szenario-Gehirn zuständig. Dessen Entscheidungsprozess läuft kontinuierlich nach dem folgenden Muster ab:

1. Daten finden
   Der Agent vergleicht die aktuell simulierte Zeit mit den Szenariendaten.
2. Zielpreis ermitteln
   Konnte ein Datenpunkt identifiziert werden, wird der Zielpreis, welchen die Aktie annehmen sollen, ermittelt. Dabei wird der aktuelle Marktpreis von der Börse mit dem Datenpunkt verrechnet.
3. Matching Order erstellen
   Die in Abschnitt 4.1 beschriebenen Orders werden basierend auf dem Datenpunkt erstellt.

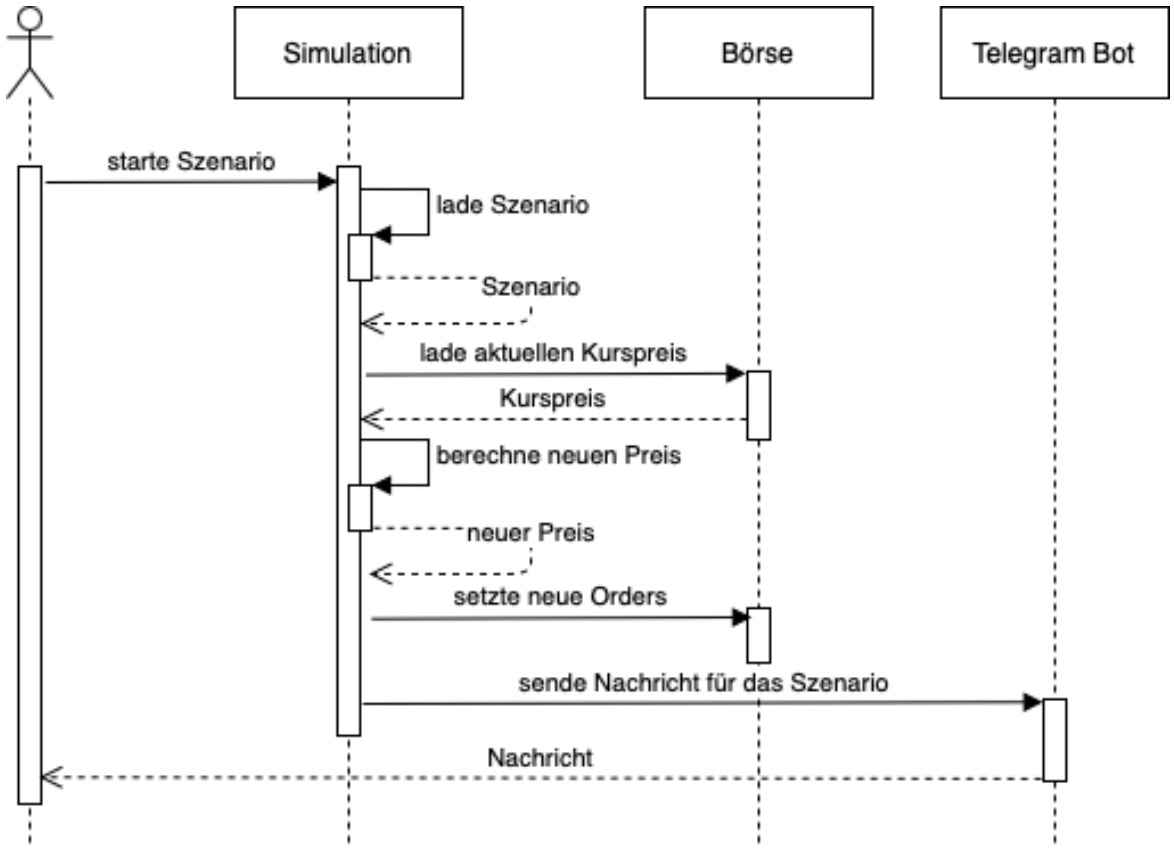

Abbildung 5.2: Sequenzdiagramm

6 Nutzerhandbuch

Startet ein Nutzer die Anwendung, so wird er zu Beginn nach einem Token gefragt. Dieser Token ist der für die Kommunikation mit der Börse benötigte API-Token und gewährleistet, dass mit der Börse nur autorisierte Nutzer interagieren können. In unserer Anwendung wurde auf ein Login verzichtet, da bereits durch die Eingabe des API-Tokens gewährleistet wird, dass nur berechtigte Personen Simulationen starten können.

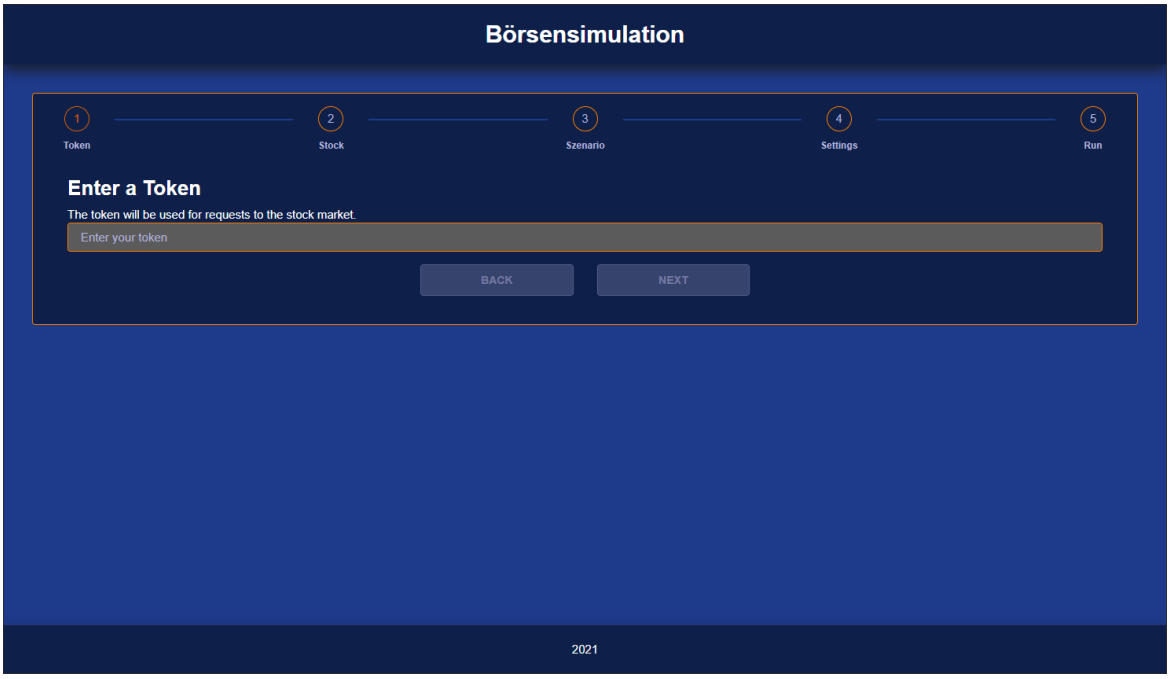

Abbildung 6.1: Start der Anwendung

Das Überspringen dieses Schrittes ist wie in der Abbildung ersichtlich nicht möglich, da der Button „next“ erst aktiviert wird, wenn ein Token eingegeben wurde. Dies wird aus Abbildung 6.1 und Abbildung Abbildung 6.2 ersichtlich. Zudem bietet das gewählte Verfahren den Vorteil, dass der API-Token nirgends in der Simulation gespeichert werden muss. So kann dieser auch nicht durch Dritte ausgelesen werden.

Im nächsten Schritt (Abbildung 6.3) wählt der Nutzer eines der zur Verfügung stehenden Wertpapiere aus, für das er eine Simulation starten möchte. Hier stehen alle Wertpapiere zur Auswahl, die an der Börse gehandelt werden. Nimmt die Börse weitere Wertpapiere in den Handel auf, so sind diese sofort in dieser Liste aufgenommen.

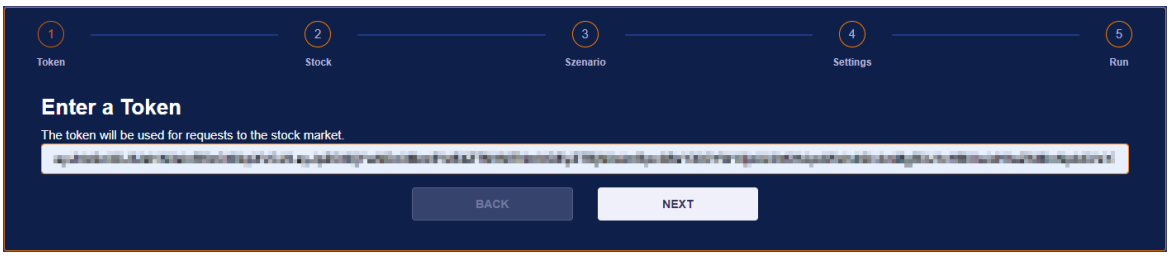

Abbildung 6.2: Eingabe eines Tokens

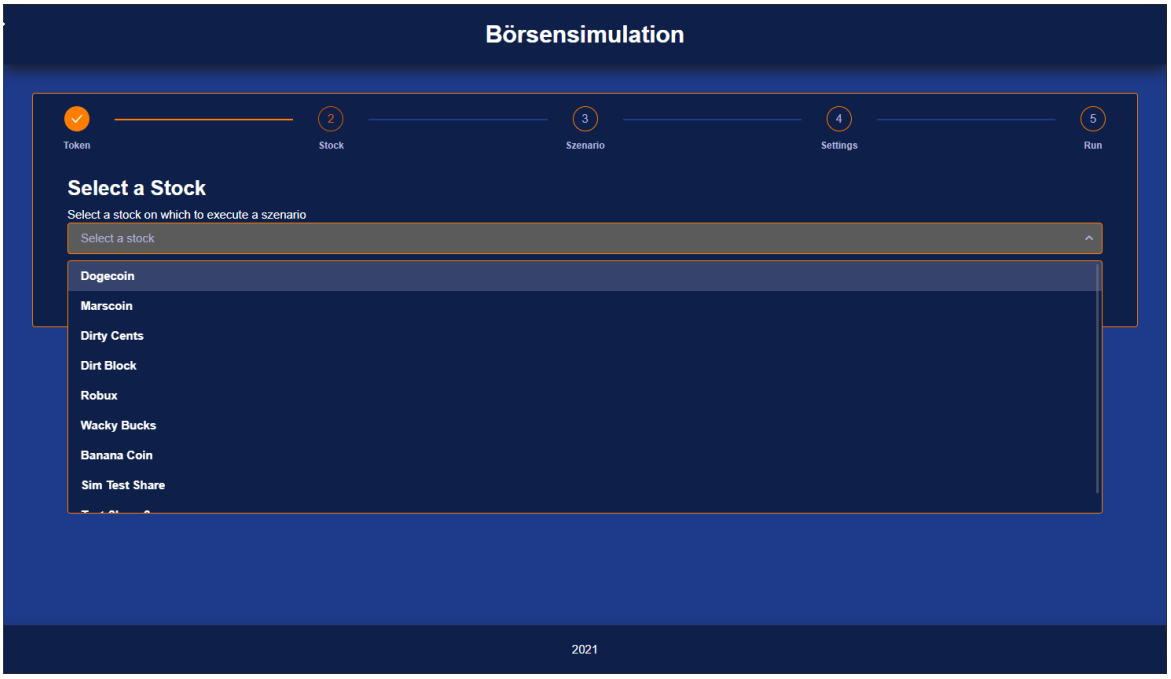

Abbildung 6.3: Auswahl eines Wertpapiers

Mit dem Bestätigen des „next“ Buttons, gelangt der Nutzer zur Auswahl der Szenarien. Aktuell stehen dabei folgende Szenarien zur Verfügung:

- Normaler Handelstag
- Positive Nachricht
- Geringes gehandeltes Volumen
- Kursabsturz durch Übernahme
- Hohes gehandeltes Volumen

Bei der Auswahl eines dieser Szenarien, wird darunter das simulierte Marktgeschehen dieses Szenarios angezeigt (Abbildung 6.4). Die Preise werden dabei als Delta aufgelistet, da

sie sich an die Kurswerte der einzelnen Wertpapiere anpassen und nicht die Preise auf ein gleiches Niveau drücken. Anschließend muss auch hier mit „next“ bestätigt oder mit „back“ zur Auswahl eines Wertpapieres zurückgekehrt werden.

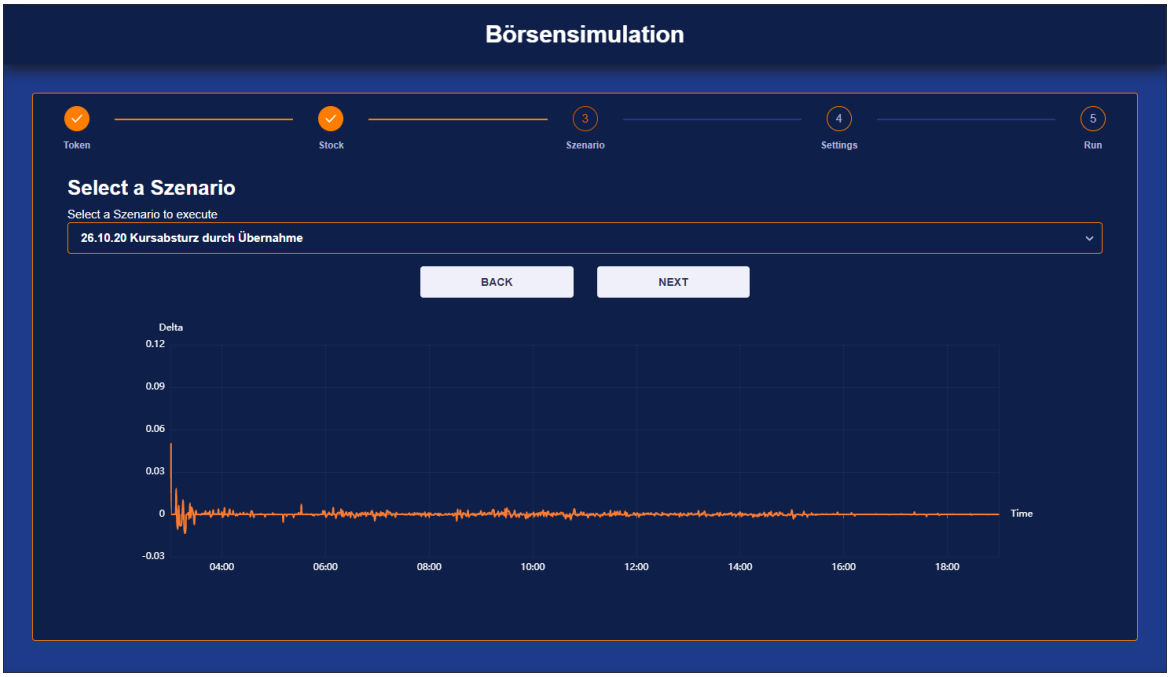

Abbildung 6.4: Auswahl eines Szenarios

Der Speedmultiplikator (Abbildung 6.5) definiert den Beschleunigungsfaktor, der bestimmt, wie viel schneller ein Szenario im Vergleich zur Echtzeit ablaufen soll. Aus Performance Gründen empfehlen wir diesen nicht über 10 und maximal 15 zu setzen, da es ansonsten zu Performance Einschränkungen der Börse kommen kann. Dies würde alle weiteren Broker beeinflussen.

In Abbildung 6.6 ist die Ansicht dargestellt, die während der Szenarioausführung angezeigt wird. Der Balken mit der Inschrift „Running“, zeigt den Fortschritt des ausgewählten Szenarios. Über den „Stop“ Button kann das Szenario jederzeit wieder gestoppt werden. Die Wirkung des Szenarios und das Testen von eigenen Strategien kann nun mit den durch die Broker und die Börse bereitgestellten Daten und Tools bewertet werden.

Abbildung 6.5: Auswahl der Settings

Abbildung 6.6: Laufendes Szenario

7 Zusammenfassung

In der vorliegenden Arbeit haben wir uns damit auseinandergesetzt, wie sich Börsenverläufe simulieren lassen können. Nach der theoretischen Einarbeitung, wurde ein Tool entwickelt, das reale Marktdaten nutzt und damit den Verlauf diverser Wertpapiere simuliert und das Handeln an einer Börse unter Markt ähnlichen Bedingungen ermöglicht. Dabei sei darauf hinzuweisen, dass wir den Anwendungsfall hauptsächlich für Bildungszwecke sehen. Zwar orientieren sich die enthaltenen Szenarien an echten Börsenverläufen, jedoch sind reale Börsenzusammenhänge sehr komplex und die Marktsituationen heute können sich von den von uns gewählten historischen Situationen unterscheiden. Eine in der Simulation gut funktionierende Strategie muss daher nicht zwingend auf einem realen Markt funktionieren. Außerdem distanzieren wir uns von sämtlichen anlageberaterischen Tätigkeiten. Zweck der Anwendung ist ausschließlich das spielerische Erlangen eines besseren Verständnisses über den Aktienmarkt und unterschiedliche Strategien. Dabei ermöglichen wir eine risikofreie Testung von unterschiedlichen Strategien und Ansätzen, da die durch die Simulation durchgeführten Transaktionen keinen Bezug zu realen Währungen und den damit verbundenen finanziellen Verlusten haben.

7.1 Fazit

Insgesamt sind wir mit der Ausarbeitung und der Arbeit innerhalb unserer Teilgruppe sehr zufrieden. Dennoch hat das Projekt auch bei uns einige Learnings hervorgebracht. Diese liegen im Wesentlichen in der Teamübergreifenden Arbeit und dem Projektgeschehen.

- Früher POCs und MVPs nutzen:
  Dies hätte uns potenziell die Mehraufwände und Einschränkungen durch die Performance und Lastprobleme zwischen der Simulation und der Börse erspart, die sich auch auf die anderen Broker ausgewirkt hat.
- Effektivere Kommunikation in den Abstimmungsmeetings (Agenda, o. Ä.)
- Klarere Aufgabendefinitionen:
  Hätte dazu geführt Aufwände bei der theoretischen Einarbeitung und Implementierungen zu reduzieren.

- Früher Feedback vom Kunden:
  Auch hier hätte uns schnelleres Feedback unter Umständen Aufwände ersparen können.
- Definierte Ansprechpartner im Fehlerfall

In der von uns entwickelten Anwendung sehen wir vor allem den Vorteil, dass diese sehr skalierbar ist. So können einfach weitere Aktien hinzugefügt und simuliert werden. Auch das Ergänzen von weiteren Szenarien ist ohne weitere Implementierungsaufgaben möglich. Außerdem kann unsere Anwendung auch mit höheren Frequenzen, also mehr Trades pro Sekunde, und deutlich größeren Volumina arbeiten.