

2021

Philip Baumann  
Dániel Polgár

Leo Scherer

Cedric Gasser

Mock-Stock

# Abstract

Im Rahmen unserer Projektarbeit erarbeiteten wir einen fiktiven Aktienmarkt, welcher als Simulationsspiel für Kundinnen und Kunden unserer Firma dienen soll. Das Spiel soll Kunden auf das Ökosystem des Unternehmens aufmerksam machen und so andere Produkte der Unternehmung vermarkten. Das Projekt, welches den Übernahmen Mock-Stock trägt, agiert auf Basis von Echtzeit-Aktiendaten. So hat das Spiel einen möglichst grossen Realitätsbezug. Eine Rangliste ermöglicht den Benutzern sich untereinander zu vergleichen und ihre Highscores zu schlagen

Um auf die Bedürfnisse unserer Arbeitgeber eingehen zu können und das Projekt auch erfolgreich umsetzen zu können, wurde eine für unsere Bedürfnisse modifizierte Art der agilen Vorgehensmethode Scrum\* angewendet. Wir begannen mit der Definierung und Priorisierung der Anforderungen und hielten diese auf unserem virtuellen Scrum-Board fest.

Die Benutzeroberfläche der Simulation wurde mit dem Angular-Framework umgesetzt. Diesbezüglich haben wir TypeScript\* im Zusammenhang mit HTML und CSS verwendet. Die Benutzeroberfläche kommuniziert über REST mit dem Backend, welches mit dem Python Framework Flask\* implementiert wurde. Die Echtzeit-Daten werden über die von Yahoo öffentlich zur Verfügung gestellte API\* angezogen. Die Käufe, Verkäufe und Saldo werden in einer MongoDB\* persistiert.

Inhalt

[Abstract 1](#_Toc90327802)

[PA Bericht Teil 1 5](#_Toc90327803)

[1 Projektorganisation 6](#_Toc90327804)

[1.1 Experten 6](#_Toc90327805)

[1.2 Beteiligte Personen 6](#_Toc90327806)

[1.3 Dokumente 6](#_Toc90327807)

[1.3.1 Dokument Aufbau 6](#_Toc90327808)

[1.4 Arbeitsort 6](#_Toc90327809)

[1.5 Informationsquellen 6](#_Toc90327810)

[1.6 Hilfestellungen 7](#_Toc90327811)

[1.7 Vorgehensmethode 7](#_Toc90327812)

[1.8 Zeitplanung 7](#_Toc90327813)

[2 Arbeitsjournal 7](#_Toc90327814)

[PA Bericht Teil 2 8](#_Toc90327815)

[3 Ist-Soll Vergleich 9](#_Toc90327816)

[4 Anforderungen 9](#_Toc90327817)

[4.1 Userstories 9](#_Toc90327818)

[4.1.1 Datenstruktur 9](#_Toc90327819)

[4.1.2 Datenmengen 9](#_Toc90327820)

[4.1.3 Applikationsanforderungen 9](#_Toc90327821)

[4.1.4 Bedienung 10](#_Toc90327822)

[4.1.5 Umgebung 10](#_Toc90327823)

[4.2 Wireframes 11](#_Toc90327824)

[4.3 Projektstrukturplan 17](#_Toc90327825)

[5 Systemabgrenzung 18](#_Toc90327826)

[5.1 Backend Architektur 18](#_Toc90327827)

[5.2 API-Definition 19](#_Toc90327828)

[6 Risikoanalyse 21](#_Toc90327829)

[6.1 Datenverlust 21](#_Toc90327830)

[6.2 Coronavirus 21](#_Toc90327831)

[6.3 Fehlerhafter Zeitplan 22](#_Toc90327832)

[6.4 Fehler in 3. Party Framework 22](#_Toc90327833)

[6.5 Risikovergleich 22](#_Toc90327834)

[6.6 Auswertung der Risiken 23](#_Toc90327835)

[7 Scrum 23](#_Toc90327836)

[7.1 Rollenverteilung 23](#_Toc90327837)

[7.1.1 Product Owner 23](#_Toc90327838)

[7.1.2 Scrum Master 24](#_Toc90327839)

[7.1.3 Entwickler 24](#_Toc90327840)

[7.2 Artefakte 24](#_Toc90327841)

[7.2.1 Product Backlog 24](#_Toc90327842)

[7.2.2 Sprint Backlog 24](#_Toc90327843)

[7.2.3 Product Increment 24](#_Toc90327844)

[7.2.4 Sprint 24](#_Toc90327845)

[7.3 Ereignisse 24](#_Toc90327846)

[7.3.1 Refinement 24](#_Toc90327847)

[7.3.2 Sprint Planning 25](#_Toc90327848)

[7.3.3 Daily 25](#_Toc90327849)

[7.3.4 Sprint Review 25](#_Toc90327850)

[7.3.5 Retrospektive 25](#_Toc90327851)

[8 C4 Diagramme 26](#_Toc90327852)

[8.1 Legende 26](#_Toc90327853)

[8.2 System Context Diagramm 27](#_Toc90327854)

[8.3 Component Diagramm 27](#_Toc90327855)

[9 Umsetzung 28](#_Toc90327856)

[9.1 Yfinance Modul 28](#_Toc90327857)

[9.2 Models & Adapter 28](#_Toc90327858)

[9.3 Flask Login 29](#_Toc90327859)

[9.4 Performance Optimierungen 30](#_Toc90327860)

[9.5 Caching 30](#_Toc90327861)

[9.6 Styling 31](#_Toc90327862)

[9.7 Services 31](#_Toc90327863)

[9.8 Komponente 31](#_Toc90327864)

[9.9 Routing in Frontend 31](#_Toc90327865)

[9.10 Auth-Guard 31](#_Toc90327866)

[10 Testkonzept und Protokoll 32](#_Toc90327867)

[10.1 Testziele 32](#_Toc90327868)

[10.2 Teststrategie 32](#_Toc90327869)

[10.3 Umfang Testumgebung 32](#_Toc90327870)

[10.4 Integration in Projektaktivitäten 32](#_Toc90327871)

[10.5 Tests nach den Anforderungen 33](#_Toc90327872)

[11 Setup 36](#_Toc90327873)

[11.1 Docker (Empfohlen) 36](#_Toc90327874)

[11.2 Lokal (Für Entwickelnde) 36](#_Toc90327875)

[12 Reflexion 37](#_Toc90327876)

[13 Glossar 38](#_Toc90327877)

[14 Quellenverzeichnis 40](#_Toc90327878)

[15 Abbildungsverzeichnis 40](#_Toc90327879)

[16 Tabellenverzeichnis 41](#_Toc90327880)

[17 Abbildungsverzeichnis 42](#_Toc90327881)

[18 Selbstständigkeitserklärung 43](#_Toc90327882)

[19 Anhang 43](#_Toc90327883)

# PA Bericht Teil 1

# Projektorganisation

Im folgenden Abschnitt wird die Projektorganisation der simulierten PA beschrieben.

## Experten

|  |  |
| --- | --- |
| Hauptexperte | Nebenexperte |
| Ramon Suter | - |

Tabelle 1: Experten

## Beteiligte Personen

|  |  |  |
| --- | --- | --- |
| Name | Projektrolle | Rolle in der Schule |
| Ramon Suter | Product Owner | Lehrer |
| Leo Scherer | Entwickler, Tech Lead | Lernender Informatik, Fachrichtung Applikationsentwicklung |
| Philip Baumann | Entwickler, Projektmanager | Lernender Informatik, Fachrichtung Applikationsentwicklung |
| Dániel Polgár | Entwickler, Stellvertreter des Projektmanagers | Lernender Informatik, Fachrichtung Applikationsentwicklung |
| Cedric Gasser | Entwickler, Softwarearchitekt | Lernender Informatik, Fachrichtung Applikationsentwicklung |

Tabelle 2: Beteiligte Personen

## Dokumente

Die praktische Arbeit umfasst folgende Dokumente:

* Projektabschlussbericht
* Projektstatusberichte
* Arbeitsjournale
* Programmcode auf [GitHub](https://github.com/the-philip-baumann/mock-stock)

### Dokument Aufbau

Dieses Dokument wird die folgende Formatierung einhalten, um ein einfaches und effizienten Lesen zu ermöglichen. Diese Formatierungen gelten ab diesem Abschnitt des Dokuments.

* Fliesstext
* Kommandozeile
* Programmcode wird als Bild dargestellt und entsprechend gekennzeichnet
* Wörter, welche im Glossar beschrieben sind, werden bei der ersten Nennung mit einem Stern (**\***) gekennzeichnet

## Arbeitsort

Die Arbeit wird hauptsächlich in der Schule durchgeführt. Vereinzelt wird das Team Arbeiten im Homeoffice durchführen.

## Informationsquellen

Als primäre Informationsquelle ist Google zu verwenden, wobei alle Formen der Informationsbeschaffung bei diesem Projekt erlaubt sind. Alle relevanten Informationsbeschaffungen sind im Dokument aufgeführt.

## Hilfestellungen

Hilfeleistungen können von Mitarbeitenden der jeweiligen Firma und Herr Suter bezogen werden. Diese werden jeweils im Arbeitsjournal gekennzeichnet.

## Vorgehensmethode

Während des Projekts arbeiteten wir nach Scrum. Da wir aber nicht die Zeit hatten, welche einem echten Scrum Team zur Verfügung steht, passten wir den Prozess etwas an. Im Grundsatz ist Scrum eine agile Vorgehensmethode zur Entwicklung von Software. Die Entwicklung des Produkts nach Scrum geschieht in Iterationen\*. Weitere Details zu Scrum sind im Kapitel 8 genauer beschrieben.

## Zeitplanung

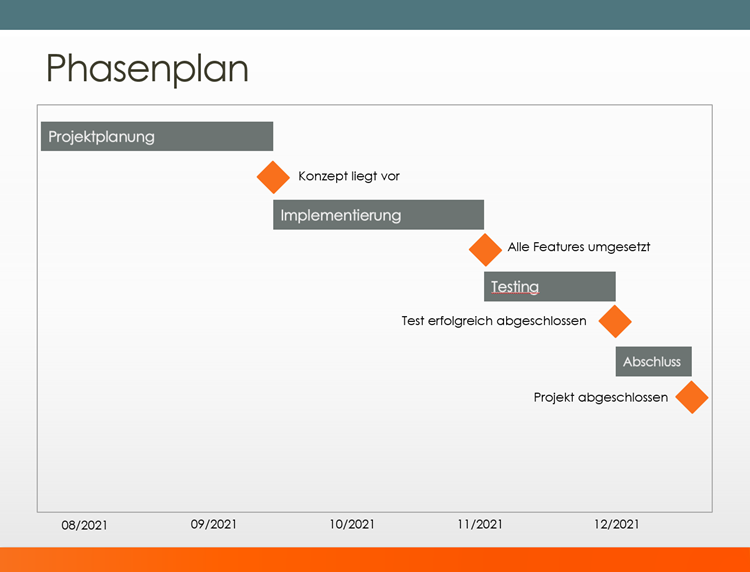


Abbildung 1: Phasenplan

Alle Phasen wurden mit den darauffolgenden Meilensteinen jeweils termingerecht abgeschlossen.

# Arbeitsjournal

Im Rahmen dieser Projektarbeit werden die Arbeitsjournale separat von jedem Teammitglied abgegeben. Daher werden in dieser Dokumentation die Arbeitsjournale nicht aufgeführt. Folgend wird Grob aufgeführt, wer was umgesetzt hat.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Philip Baumann** | **Dániel Polgár** | **Cédric Gasser** | **Leo Scherer** |
| **09.09. SW3** | Lastenheft | Lastenheft | Business Case | Business Case |
| **16.09. SW4** | Projektauftrag | Projektauftrag | Projektphasenplan | Projektphasenplan |
| **23.09. SW5** | Testkonzept | Design, Schnittstellendesign | Design, Schnittstellendesign | Testkonzept |
| **30.09. SW6** | Projektstrukturplan | Projektstrukturplan | Arbeitspaket | Arbeitspaket |
| **21.10. SW7** | Deployment Script, Deployment auf Server | Setup UI, UI Library einbinden | Setup Flask backend, Aktien Detail API (in Progress) | YFinance evaluation, Anbindung In-memory-DB |
| **28.10.**  **SW8** | PSP 1 | Login/Register UI (in Progress) | Login/Register API, Startkapital API | Buy/Sell API, Logout API |
| **04.11.**  **SW9** | Server Infrastruktur | Login/Register UI, Logout UI | Leaderboard API | Datenbankanbindung |
| **11.11.**  **SW10** | PSP 2 | Detailpage UI (in Progress) | Aktien Detail API (done) | Aktienliste API |
| **18.11.**  **SW11** | Doku Abstract, Scrum | Buy/Sell UI, Detail Page UI | Performance improvements | Aktienliste anzeigen UI |
| **25.11.**  **SW12** | PSP 3 | Eigenes Profil UI, andere Spielerprofile UI | Eigenes Profil API, andere Spielerprofile API | Leaderboard UI |
| **SW13**  **02.12.** | Testprotokoll / IST-SOLL | Testprotokoll / Glossar | Testprotokoll / Zeitplanung | Testprotokoll / Projektorganisation |
| **09.12.**  **SW14** | Reflexion, Überarbeitug Dokument, Scrum, Selbständigkeits-erklärung | Pflichtenheft, Umsetzung UI, alle Verzeichnisse | Umsetzung API, Systemabgrenzung | Risikoanalyse, Setup, C4 Diagramme |

# PA Bericht Teil 2

# Ist-Soll Vergleich

Grundsätzlich ist zu sagen, dass unser Soll-Zeitplan nicht stark ins Detail geht, weshalb eine Auswertung im Nachhinein etwas schwieriger ist. Trotzdem können wir sagen, dass wir sämtliche Meilenstein und damit verbundenen Dokumente rechtzeitig und in entsprechender Ausführung einreichen konnten. Weshalb wir trotzdem sagen können, dass unsere Zeitplanung erfolgreich war.

# Anforderungen

## Userstories

Um die Anforderungen zu definieren, haben wir uns entschieden Userstories zu schreiben. Bei unseren Betrieben werden diese auch verwendet. Ebenfalls passen sie perfekt in den Scrum Prozess.

### Datenstruktur

|  |  |  |
| --- | --- | --- |
| **SWE-Nummer** | **Bezeichnung** | **Priorität** |
| 1000 | Der Spieler kann sich bei der Webseite mit Personalien (Username, E-Mail, Passwort) registrieren. | Hoch |
| 1001 | Der Spieler kann sich bei der Webseite mit (E-Mail, Passwort) einloggen. | Hoch |
| 1002 | Beim Erwerb einer Wertschrift des Spielers werden Informationen (Anzahl, Name, Erwerbspreis, Zeitpunkt des Kaufes) gespeichert. | Hoch |
| 1003 | Die anzubietenden Wertschriften werden mit Namen und Kürzel gespeichert. | Hoch |

Tabelle 3: Userstories - Datenstruktur

### Datenmengen

|  |  |  |
| --- | --- | --- |
| **SWE-Nummer** | **Bezeichnung** | **Priorität** |
| 1004 | Die Applikation muss bis zu 50'000 Spieler verwalten können ohne Performance Einbussen zu erleiden. Bei der Performance ist auf die Reaktionszeit und Verarbeitungszeit der Applikation zu achten. | Hoch |
| 1005 | Kauf und Verkauf von Wertschriften muss mit einer Geschwindigkeit von unter einer Sekunde geschehen. | Mittel |

Tabelle 4: Userstories - Datenmengen

### Applikationsanforderungen

|  |  |  |
| --- | --- | --- |
| **SWE-Nummer** | **Bezeichnung** | **Priorität** |
| 1006 | Mehrere Spieler können gleichzeitig Wertschriften erwerben und verkaufen, ohne die Datenkonsistenz zu verletzen. | Hoch |
| 1007 | Der Spieler kann eine bestimmte Wertschrift zu einem Preis, der dem aktuellen Kurs entspricht, erwerben. | Hoch |
| 1008 | Der Spieler kann eine bestimmte Wertschrift zu einem bestimmten Preis, der dem aktuellen Kurs entspricht zu einem Gewinn oder einem Verlust verkaufen. | Hoch |
| 1009 | Der Spieler startet mit dem Initialkapital von 20'000 Franken, welches nach dem Registrieren direkt verfügbar ist. | Hoch |
| 1010 | Der Spieler kann sich auf einem Leaderboard mit anderen Spielern messen können. Das Leaderboard misst sich am Wert der aktuellen Wertschriften, welche ein Spieler zum gegebenen Zeitpunkt besitzt. | Hoch |
| 1011 | Nach dem Druck des «Verkaufen» Buttons wird die von dem Benutzer ausgewählte Wertschrift verkauft. | Hoch |
| 1012 | Die Daten im Spiel werden gespeichert, dass sie beim nächsten Anmelden nicht verloren gehen. | Hoch |

Tabelle 5: Userstories - Applikationsanforderungen

### Bedienung

|  |  |  |
| --- | --- | --- |
| **SWE-Nummer** | **Bezeichnung** | **Priorität** |
| 1013 | Die Applikation muss mit der Kombination von Tastatur und Maus bedienbar sein. | Hoch |

Tabelle 6: Userstories - Bedienung

### Umgebung

|  |  |  |
| --- | --- | --- |
| **SWE-Nummer** | **Bezeichnung** | **Priorität** |
| 1014 | Die Applikation ist mit dem Google Chrome Browser aufzurufen mit einer Verfügbarkeit von 24/7. | Hoch |
| 1015 | Die Applikation muss auf dem CSS-Lernenden Server lauffähig sein, wonach sich die Performance Massstäbe halten müssen. | Hoch |

Tabelle 7: Userstories - Umgebung

## Wireframes

A képen szöveg látható

Automatikusan generált leírás

Abbildung 2: Benutzerkonto erstellen

A képen szöveg látható

Automatikusan generált leírás

Abbildung 3: Anmeldung

A képen asztal látható

Automatikusan generált leírás

Abbildung 4: Aktienübersicht

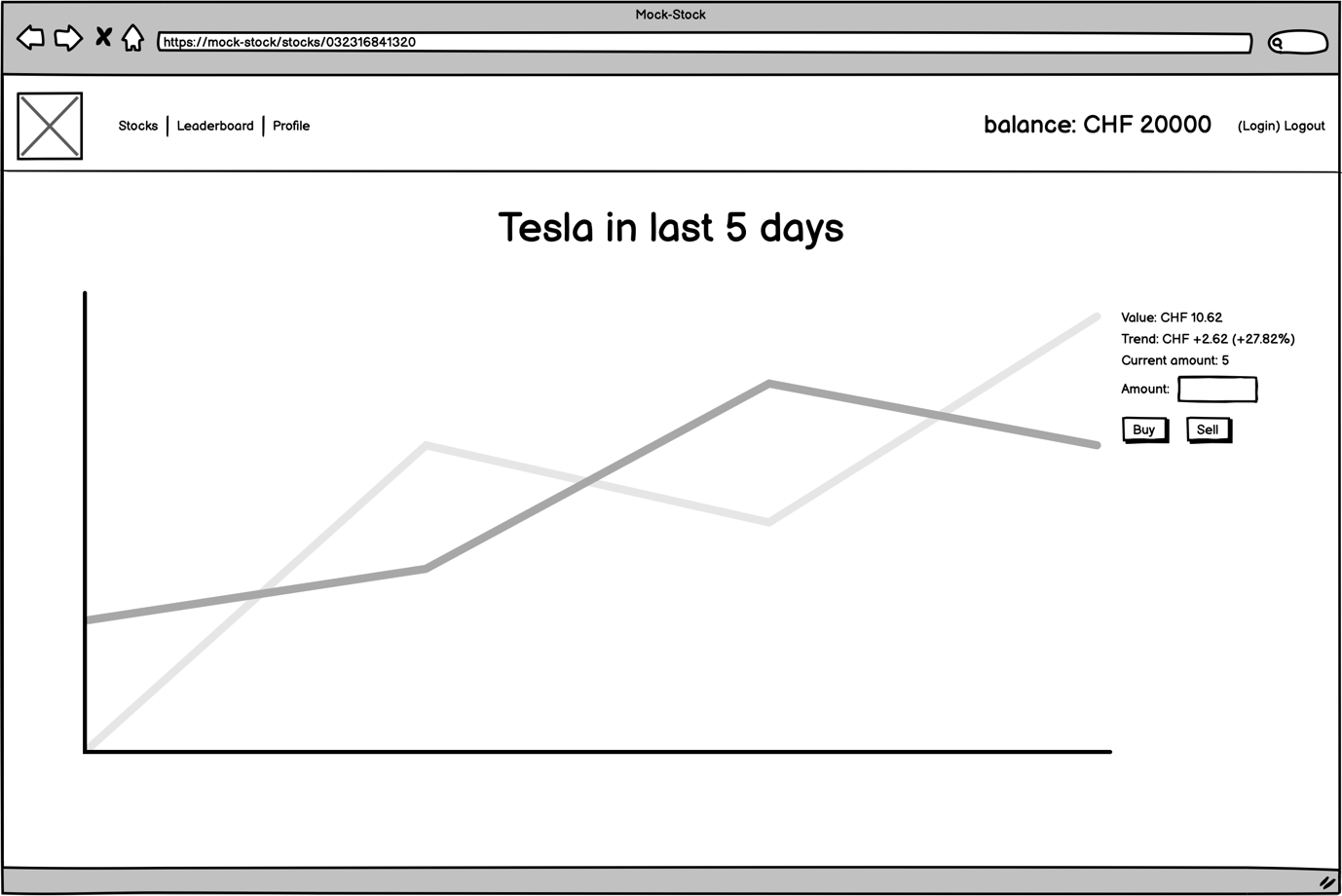


Abbildung 5: Aktien Detailansicht

A képen asztal látható

Automatikusan generált leírás

Abbildung 6: Rangliste

A képen asztal látható

Automatikusan generált leírás

Abbildung 7: Aktien von einem anderen oder eingeloggten Benutzer 1

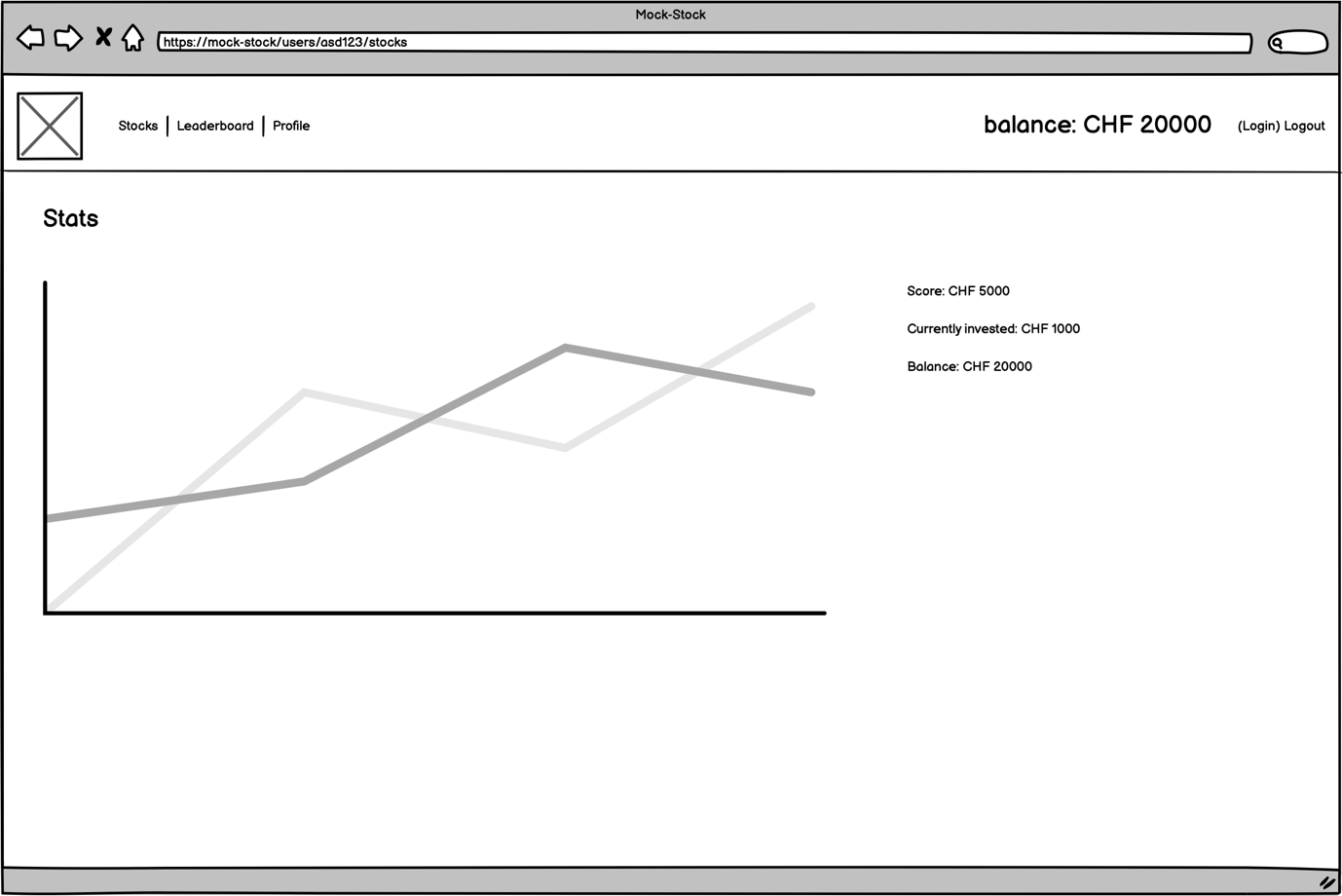


Abbildung 8: Aktien von einem anderen oder eingeloggten Benutzer 2

## Projektstrukturplan

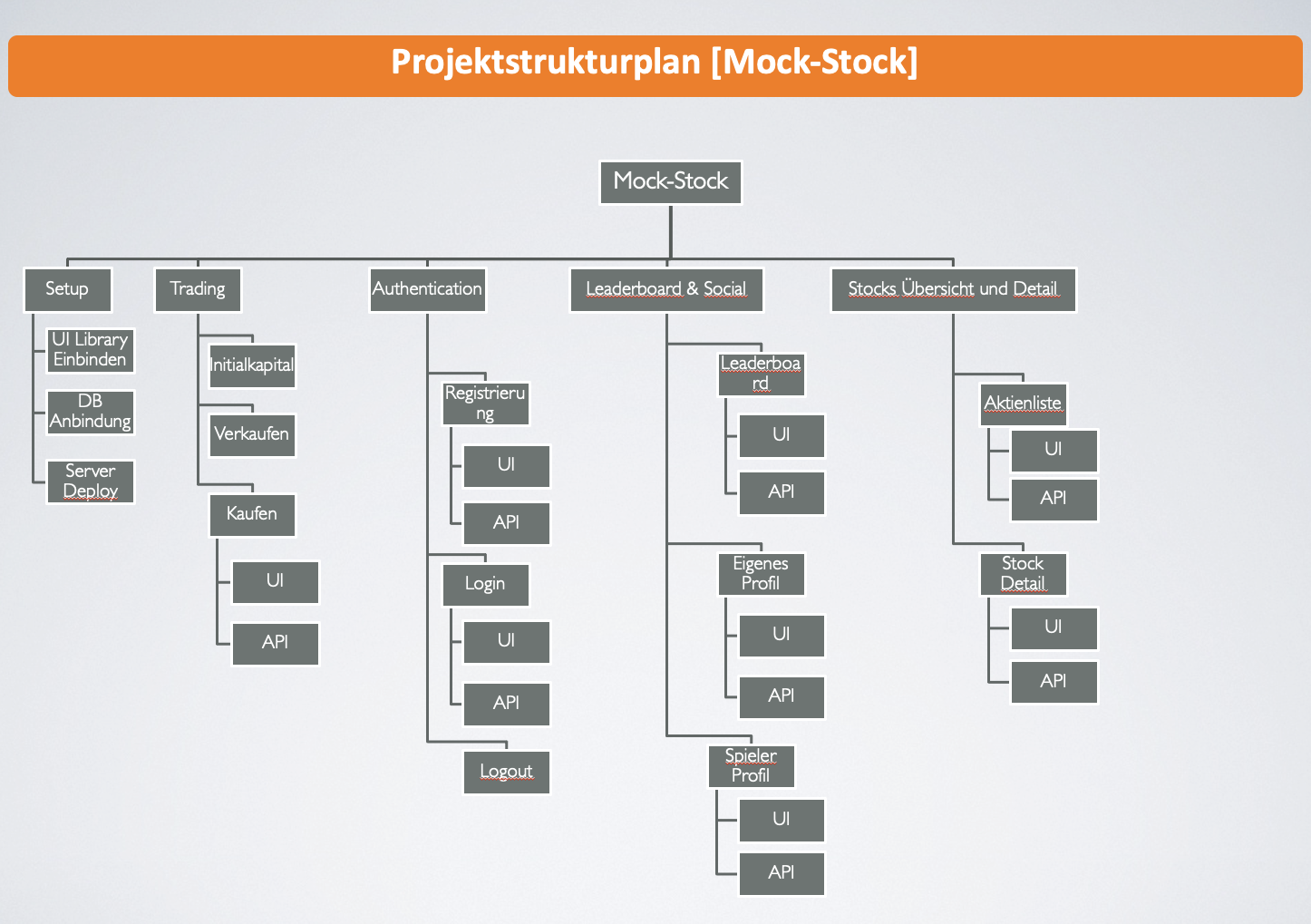


Abbildung 9: Projektstrukturplan

Die Arbeitspakete sind unter dem folgenden Link zu finden:

<https://mockstock.atlassian.net/jira/software/projects/M150/boards/1>

# Systemabgrenzung

Folgende Abschnitte beschreiben die Systemabgrenzungen und die entsprechenden Beziehungen zueinander innerhalb des Projektes.

## Backend Architektur

Durch die Einfachheit des Projektes wollen wir auf eine umfängliche Architektur verzichten. Diese hätte nur unnötige Komplexität und Aufwand bedeutet.

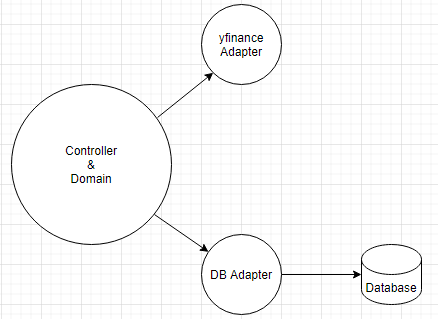


Abbildung 10: Backendarchitektur

Diese Architektur ermöglicht uns auch flexibel eine Datenbank oder einfach eine “In-Memory” Lösung zu verwenden.

## API-Definition

Die Schnittstelle zwischen dem Frontend und dem Backend ist die wichtigste Komponente in unserer Applikation. Anhand einer API-Definition können Frontend und Backend unabhängig voneinander entwickelt werden, was die Effizienz und das Agile Arbeiten fördert. Diese Definition wurde zu Beginn des Projektes in einem OpenAPI\*-File angelegt, eine Standardisierte Methodik zur Schnittstellendefintion. So konnte das Backend unabhängig vom Frontend umgesetzt werden, ohne dass die Schnittstellen während der Umsetzung aufeinander abgestimmt werden müssen.

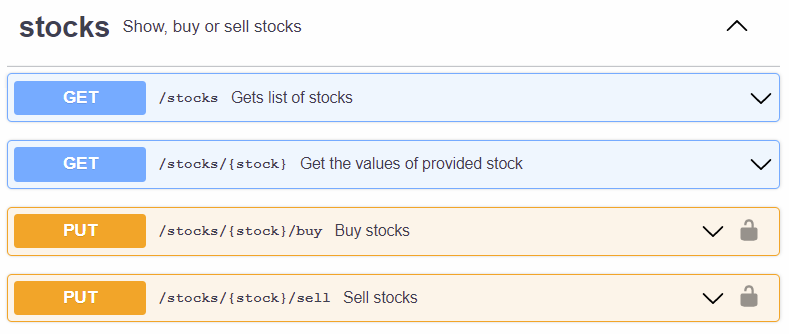


Abbildung 11: OpenAPI /stocks

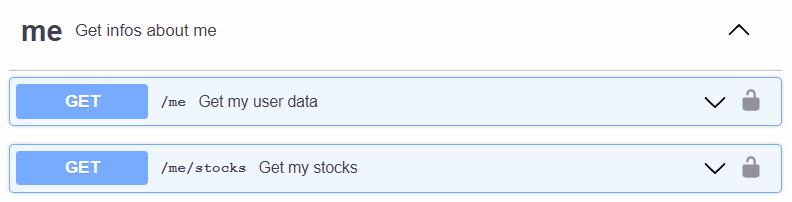


Abbildung 12: OpenAPI /me

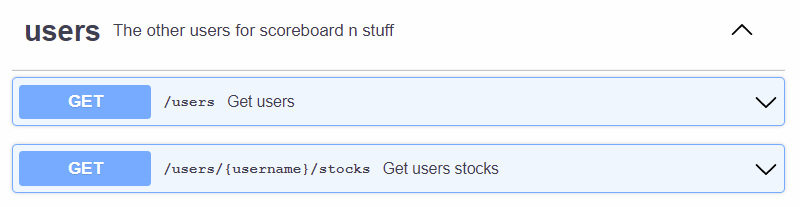


Abbildung 13: OpenAPI /users

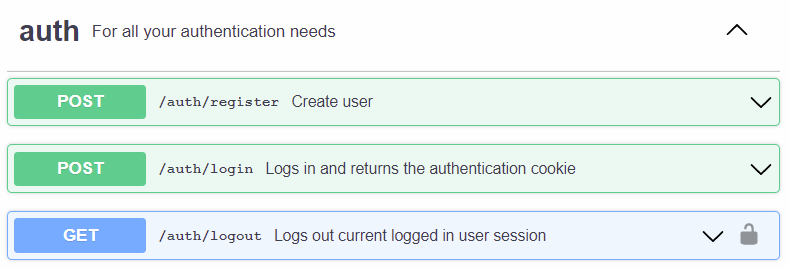


Abbildung 14: OpenAPI /auth

Für genauere Angaben bezüglich der Schnittstelle kann gerne das [OpenAPI-yaml File](https://github.com/the-philip-baumann/mock-stock/blob/master/backend/src/openapi.yaml) konsultiert werden.

# Risikoanalyse

Da jedes Projekt Risiken birgt, ist es wichtig diese im Voraus zu erkennen und wenn möglich präventive Gegenmassnahen zu treffen. Dazu wird eine Risikoanalyse durchgeführt und daraus werden die grössten Risiken mit Gegenmassnahmen versehen. Zudem hilft es auch einen guten Überblick auf die Punkte zu behalten, welche besondere Beachtung benötigen. Alle Risiken werden mit einer potenziellen Schadenshöhe und einer Eintrittswahrscheinlichkeit bewertet. Beide Skalen gehen von 1 (tief) bis 10 (hoch).

## Datenverlust

|  |  |
| --- | --- |
| **Risiko** | Gewisse Dokumentationsabschnitte werden nur lokal gespeichert. Daher kann einen Datenverlust erfolgen. |
| **Auswirkung** | Die ganze simulierte PA\* wäre in Gefahr, denn alles bereits erarbeitete könnte verloren gehen. Bei einem Teilverlust muss mit grossen Zeiteinbussen gerechnet werden und der Zeitplan könnte nicht eingehalten werden. |
| **Ursache** | Die Daten sind nicht genügend gesichert und es existiert kein Backup. |
| **Wahrscheinlichkeit** | 9 – bereits ein versehentliches Löschen kann dazu führen, dass Daten für immer weg sind. |
| **Schadenshöhe** | 6 – Die Wahrscheinlichkeit für einen grossen Datenverlust ist überschaubar, während ein kleiner Datenverlust wahrscheinlich ist. |
| **Präventionsmassnahmen** | Programmcode wird regelmässig auf GitHub gesichert. Dokumentationsdokumente werden regelmässig auf OneDrive hochgeladen. |

Tabelle 8: Risiko - Datenverlust

## Coronavirus

|  |  |
| --- | --- |
| **Risiko** | Ein Projektbeteiligter könnte durch die Ansteckung mit dem Coronavirus oder durch eine Quarantäne ausfallen. |
| **Auswirkung** | Je nach Dauer des Krankheitsausfalls müsste mit dem Hauptexperten eine Lösung gefunden werden, wie man fortfahren will. Das Arbeiten im Homeoffice wäre unter Umständen auch eine Möglichkeit. |
| **Ursache** | Ansteckung durch Coronavirus. |
| **Wahrscheinlichkeit** | 6 |
| **Schadenshöhe** | 6 |
| **Präventionsmassnahmen** | Sich an die Hygienemassnahmen halten. (BAG, 2021) |

Tabelle 9: Risiko - Coronavirus

## Fehlerhafter Zeitplan

|  |  |
| --- | --- |
| **Risiko** | Der zum Anfang des Projekts erstellte Zeitplan könnte fehlerhaft sein und somit könnte die PA nicht rechtzeitig abgeschlossen werden. |
| **Auswirkung** | Wenn nicht früh genug bemerkt wird, dass die Zeit zu knapp berechnet ist, kann dies zur Folge haben, dass die Arbeit nicht fertiggestellt werden kann. Eventuell müsste ein Teil der PA weggelassen werden. |
| **Ursache** | Es ist schwierig alle Aufgaben korrekt zu schätzen und planen, wenn man am Anfang des Projekts steht. Zudem können bei der Umsetzung unvorhersehbare Probleme und Arbeiten auftauchen. |
| **Wahrscheinlichkeit** | 8 – bei der Implementierung tauchen häufig unvorhersehbare und ungeplante Probleme auf. |
| **Schadenshöhe** | 7 |
| **Präventionsmassnahmen** | Reserve einplanen und jede Woche einen Abgleich mit dem Zeitplan machen. |

Tabelle 10: Risiko - Fehlerhafter Zeitplan

## Fehler in 3. Party Framework

|  |  |
| --- | --- |
| **Risiko** | Es kann vorkommen, dass ein benutztes Framework einen Fehler aufweist. Dies kann meine Arbeit negativ beeinflussen und Zeit benötigen. |
| **Auswirkung** | Je nach Grösse des Fehlers könnte die Funktion nicht mehr genutzt werden oder das Framework müsste sogar ausgetauscht werden. |
| **Ursache** | Wie alle Software können auch die verwendeten Frameworks Fehler haben. |
| **Wahrscheinlichkeit** | 4 – die eigesetzten Frameworks sind bekannt und laufen stabil. |
| **Schadenshöhe** | 6 |
| **Präventionsmassnahmen** | Frameworks nutzten, die von vielen anderen auch genutzt werden und z.B. Angular Material einsetzten, da dies auch vom offiziellen Angular Team gewartet wird. |

Tabelle 11: Risiko Fehler in 3. Party Framework

## Risikovergleich

Die verschiedenen Risiken werden verglichen und es wird eine Priorität von 1 (hoch) bis 3 (tief) eingeteilt.

|  |  |  |  |
| --- | --- | --- | --- |
| Risiko | Schadenshöhe | Wahrscheinlichkeit | Priorität |
| Datenverlust | 8 | 9 | 1 |
| Fehlerhafter Zeitplan | 7 | 8 | 1 |
| Coronavirus | 6 | 8 | 2 |
| Fehler in 3. Party Framework | 6 | 4 | 3 |

Tabelle 12: Risikovergleich

## Auswertung der Risiken

Die Risiken wurden im Nachhinein abgeschätzt und teilweise eingetroffen. Durch etwas Hilfe und ein wenig Glück war der Schaden aber jeweils recht klein

|  |  |  |  |
| --- | --- | --- | --- |
| Risiko | Effektive Schadenhöhe | Eintreffen | Bemerkung |
| Datenverlust | 3 | Teilweise | Die finale Form der Zwischendokumentationen war teilweise nicht in der Cloud und somit nicht für alle abrufbar. Daher musste zwischendurch länger auf ein anderes Teammitglied gewartet werden. |
| Coronavirus | 1 | Teilweise | Niemand musste in Quarantäne oder wurde infiziert, jedoch waren Teammitglieder vereinzelt krank. |
| Fehlerhafter Zeitplan | 0 | Nein | Wir waren im Zeitplan |
| Fehlerhafte Frameworks | 3 | Ja | Im Yfinance-Modul schlich sich zwar kein Bug ein, aber das Framework als Ganzes war teilweise unbrauchbar langsam. Somit hatten wir zusatzaufwand durch selbst implementiertes Caching. |

Tabelle 13: Auswertung der Risiken

# Scrum

## Rollenverteilung

Ein Scrum Team setzt sich aus drei Rollen zusammen. Anbei sind die einzelnen Rollen genauer Beschrieben.

### Product Owner

Wie der Name schon sagt, ist der Product Owner der Besitzer der Software. Er dient als Mittelmann zwischen den Anwendern der Software und dem Entwicklungsteam. Es ist in seiner Verantwortung die Anforderungen der Endkunden entgegenzunehmen und sie im sogenannten Product Backlog zu priorisieren. Diese Rolle wurde durch Ramon Sutter, Fachkraft Modul 150.

### Scrum Master

Der Scrum Master ist für die Einhaltung des Scrum Prozesses, sowie das Wohlbefinden des Teams verantwortlich. Er schützt das Team vor auswertigen Störfaktoren und geht bei allfälligen Bedürfnissen auf die einzelnen Teammitglieder ein. Diese Rolle wurde durch den Projektmanager wahrgenommen.

### Entwickler

Zu den Entwickler zählen nicht nur die Entwickler, sondern auch die Tester und Requirement Engineers. Wobei die zwei letztgenannten nicht spezifisch einer Person zu geteilt sein müssen. Es ist dem Team überlassen, wie sie die einzelnen Aufgaben aufteilen. In unserem Fall wurden die Aufgaben auf alle vier Teammitglieder aufgeteilt.

## Artefakte

Der Scrum Prozess besteht aus drei Artefakten. Anbei sind die einzelnen Artefakte genauer Beschrieben.

### Product Backlog

Der Product Backlog hält in Form einer Liste sämtliche Features, welche umgesetzt werden müssen, aber auch sämtliche Bugs, Änderungen und sonstige Vorgänge. Die Liste wird vom Product Owner verwaltet und kann ständig aktualisiert werden. Dabei kann die Priorität einzelner Vorgänge ändern oder gar neue Vorgänge hinzukommen. (Scrum.org, 2021)

### Sprint Backlog

Der Sprint Backlog beinhaltet sämtliche Vorgänge, welche während eines sogenannten Sprints abgearbeitet werden sollten. Dieser wird am Anfang jedes Sprints zusammengestellt und sollte nicht, während dem Sprint angepasst werden. (Scrum.org, 2021)

### Product Increment

Das Product Increment ist die Summe aller Product Backlog Vorgänge, die im aktuellen Sprint und allen vorherigen Sprints abgearbeitet wurden. Dabei muss das Increment in einem brauchbaren Zustand sein und zugleich der «Definition of Done» entsprechen.

### Sprint

Ein Sprint ist ein Arbeitsschritt, in welchem die Produktimplementierung stattfindet. Der Sprint wird mit einem Planing gestartet und mit einem Review oder einer Retrospektive abgeschlossen. Ein Sprint war in unserem eine Woche. Somit begann ein Sprint am Donnerstagnachmittag und endete am selben Donnerstag vor Unterrichtsbeginn.

## Ereignisse

Folgende Ereignisse werden während eines Sprints abgehalten, damit ein möglichst reibungsloser Ablauf garantiert ist.

### Refinement

Damit ein Sprint geplant werden kann ist es wichtig das die einzelnen Vorgänge nach ihrer Komplexität und dem damit verbundenen Aufwand geschätzt werden. Ebenfalls können die Vorgänge im Product Backlog besprochen werden und allenfalls noch genauer beschrieben werden. Dieses Ereignis wurde aus Zeitgründen nicht durchgeführt. (Scrum.org, 2021)

### Sprint Planning

Das Sprint Planning ist das erste Ereignis und somit auch der «Startschuss» für einen neuen Sprint. Während diesem Ereignis wird der Sprint Backlog zusammengestellt. Der Sprint wird so geplant, dass die Kapazität des Teams den Schätzungen, welche im Refinement vorgenommen wurden, entspricht. Sollte dies nicht der Fall sein können Vorgänge dem Sprint hinzugefügt werden oder im Umkehrschluss aus dem Sprint Backlog entfernt werden. Dieses Ereignis wurde aus Zeitgründen nicht durchgeführt. (Scrum.org, 2021)

### Daily

Wie der Name schon sagt, findet dieses Ereignis täglich statt. Jeder Entwickler geht dabei auf drei Punkte seiner Arbeit ein. Da wir nicht die Möglichkeit hatten jeden Tag an dem Projekt zu arbeiten wandelten wir das Daily in ein Weekly um. Die drei Punkte sind:

* Was habe ich gemacht?
* Wo bin ich auf Probleme gestossen?
* Was werde ich machen?

### Sprint Review

Hier werden die abgeschlossenen Vorgänge den Endkunden präsentiert und zusätzliches Feedback abgeholt. Dieses Ereignis findet nur einmal pro Sprint statt und erfordert die Anwesenheit der Endkunden. Dieses Ereignis wurde aus Zeitgründen nicht durchgeführt.

### Retrospektive

In der einer vom Scrum Master geleiteten Meeting wird der aktuelle Sprint besprochen und es werden eventuellen Massnahmen für Probleme, welche aufgetaucht sind, getroffen. Nach diesem Meeting gilt ein Sprint als abgeschlossen. Dieses Ereignis wurde aus Zeitgründen nicht durchgeführt. (Scrum.org, 2021)

# C4 Diagramme

Folgend wird die Applikation mit dem C4 Architekturdiagramm-Modell dargestellt. Das C4-Modell besteht aus diesen vier Diagrammen:

* **System Context** – *Beschreibt die Einbettung in ein bereits bestehendes Gesamtsystem auf höchstem Fluglevel. Es soll auch fachlich klar werden, wo die Software einzuordnen ist.*
* **Container** – *Beschreibt die zu erstellenden Applikationen (und Datenbanken, …) und deren Umsysteme und gibt an, wie diese kommunizieren sollen. Es wird der Zusammenhang der einzelnen Teilsysteme ersichtlich.*
* **Component** – *Zeigt, wo und wie genau die Teilsysteme zusammenhängen und gemeinsame Komponenten teilen. Zeigt eine detaillierte Architekturdarstellung auf. Es wird erkennbar, wie die Teilsysteme aufgebaut sind, jedoch werden keine Implementierungsdetails (Klassen, Enums, …) dargestellt.*
* **Code** – Das detaillierteste Diagramm zeigt den Code auf. Dies kann beispielsweise mittels Klassendiagramm passieren.

Bei dieser Arbeit wird auf das zweite sowie das vierte «C», Container und Code, gemäss Aufgabenstellung, verzichtet.

## Legende

Die Diagramme sind nachfolgender Notation aufgebaut.

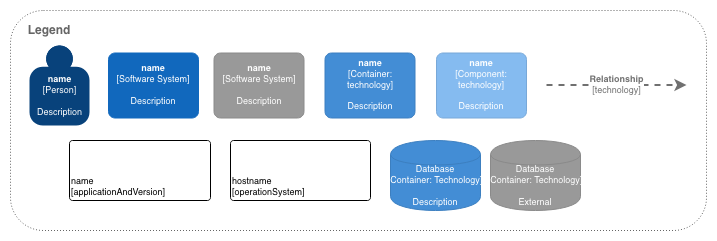


Abbildung 15: Legende C4 Diagramm (diagrams.net, 2020)

## System Context Diagramm

In folgendem Diagramm wird der Systemkontext dargestellt. Hierbei ist vor allem der externe Aufruf zum Yfinance-Modul zu beachten. Diese API wird verwendet, um die aktuellen Marktdaten abzufragen

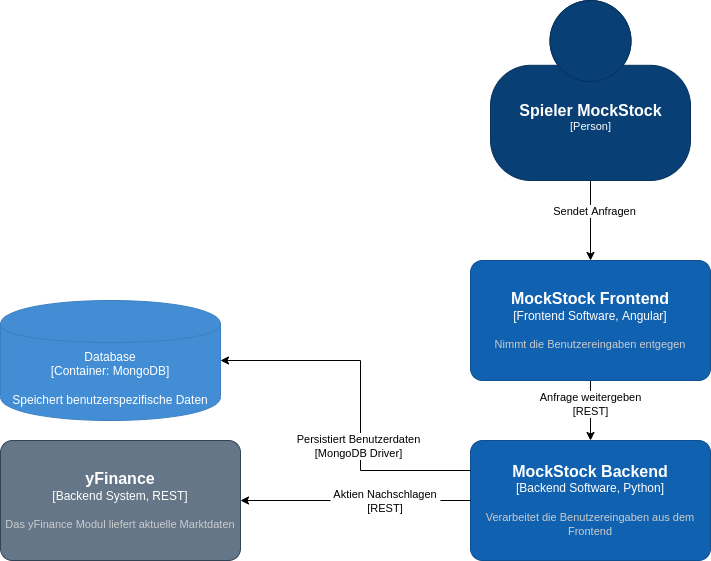


Abbildung 16: C4 Diagramm System Context

## Component Diagramm

In diesem Diagramm wird die Applikation genauer dargestellt. Das Frontend wurde in diesem Diagramm nur beschränkt dargestellt, da die interessanten Architekturentscheidungen im Backend liegen.

Diagram

Description automatically generated

Abbildung 17: C4 Diagramm Component Diagramm

# Umsetzung

## Yfinance Modul

Python ist bekannt dafür, für nahezu alles ein Modul zu haben. Dies ist auch für die Yahoo-Finance-API der Fall. Mithilfe dieses Moduls können wir einfach auf aktuelle Finanzdaten zugreifen.

## Models & Adapter

Nach genauem Abwägen der Grösse und der Komplexität des Backend haben wir den Entschluss gezogen, dass eine aufwändige Backendarchitektur wie z.B. eine hexagonale Architektur, für den Umfang unseres Projektes zu viel Aufwand bedeuten würde. Um jedoch trotzdem ein nötiges Mass an Struktur und Erweiterbarkeit in unser Backend zu schaffen, erstellten wir Models für die einzelnen Entitäten und Adapter für die Datenbank und die Yfinance-API.

Die Models beinhalten beispielsweise Methoden für das Serialisieren der Entitäten. DTOs und Domänen Entitäten haben wir in die Models integriert, um ähnliche Funktionalitäten zusammenzufassen. Bei einem grösseren Projekt würde es Sinn mache, diese Dinge zu trennen.



Abbildung 18: Serialisierungsmethode

Adapter ermöglichen es uns zu jederzeit die Yfinance-API oder auch die ganze Datenbank auszutauschen. Da wir keinerlei Kopplung zu den einzelnen Umsystemen haben. Es herrscht lediglich eine Kopplung zu den von uns bereitgestellten Interfaces. Die Implementierung dahinter ist unserem Backend unbekannt und zur Laufzeit auch nicht relevant.

## Flask Login

Das Flask Login Modul ersparte uns einige Arbeit beim Implementieren einer Registrierungs- und Loginlösung. Mit der Verwendung eines Decorators konnten wir ohne grossen Aufwand, die Zugriffe auf einzelne Endpunkte einschränken. Das Login zuerst umzusetzen, war aus unserer Sicht eine wichtige Entscheidung, da alles andere darauf aufbaut. Wie andere Projekte zeigten, ist es immer schwierig, nachträglich ein Login zu implementieren.

Grundsätzlich wird bei Flask Login eine Funktion benötigt, welche mit einem Username einen User laden kann. Diese Funktion kann, wie praktisch alles andere von Flask-Login, durch einen Decorator definiert werden.

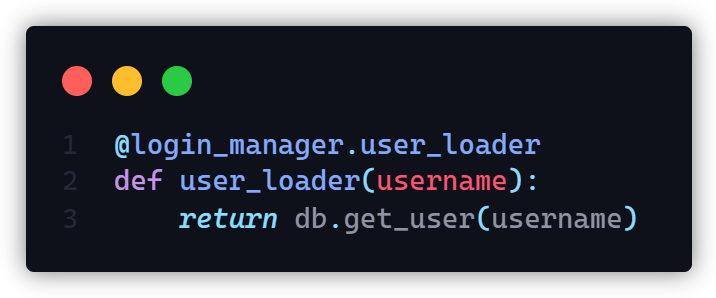


Abbildung 19: User-Loader definieren

Route Guards können dann bei einer Flask-Route einfach mit dem Decorator @login\_required hinzugefügt werden. Das Flask-Login half auch bei der Implementierung des Ein- und Ausloggen. Die beiden Methoden login\_user() und logout\_user() waren dabei von Bedeutung.

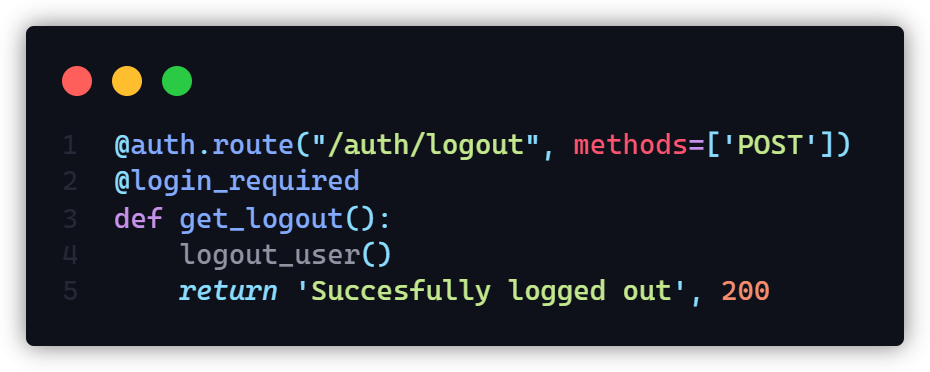


Abbildung 20: Geschützte Route

## Performance Optimierungen

Nach unseren ersten Tests haben wir herausgefunden, dass die Ticker des Yfinance Modul sehr langsam sind. Um umfassende Infos zu einer einzigen Wertschrift zu holen, mussten wir ungefähr drei Sekunden warten. Wir haben dann als Verbesserung die Download-Funktion des Yfinance Moduls verwendet. Sie lädt gezielt nur die nötigen Daten herunter. Sie kann zugleich die Daten von mehreren Wertschriften ohne grosse Verzögerung auf einmal laden. Diese Funktion konnte uns die Beschreibungen der einzelnen Wertschriften nicht mitgeben. Dabei half ein lokales CSV File, in welchem die Beschreibungen und Keys von 4704 verbreiteten Stocks stehen.

## Caching

Für einzelne Stock Abfragen verwendeten wir das Ticker Objekt von Yfinance. Dieses dauert eine Weile zum Erstellen, liefert jedoch detailreiche Informationen. Um diese Objekte nicht jedes Mal neu erstellen zu müssen, cachen wir mit dem folgenden Code ein Objekt nach erstmaligem Aufrufen. Bei künftigen Anfragen auf eine bestimmte Wertschrift kann lediglich auf den Cache zugegriffen werden. Dies hat zur Folge, dass die Abfrage viel schneller geschieht.

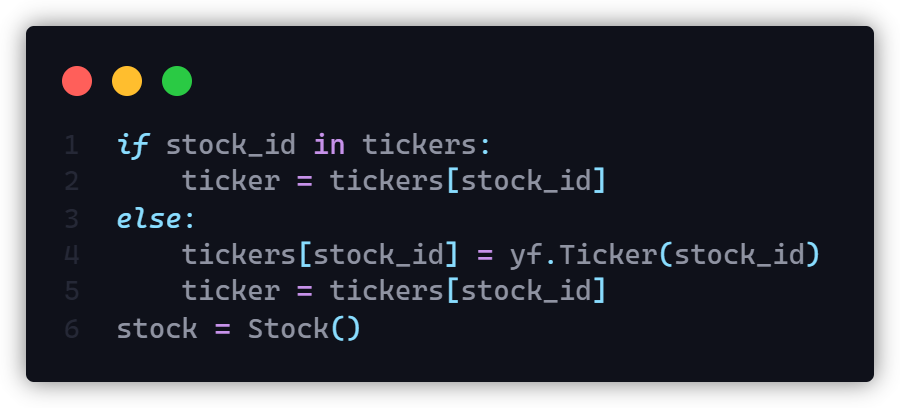


Abbildung 21: Caching-Implementierung

## Styling

Damit wir beim Theming und beim Responsive Design Zeit sparen, haben wir externe Style Frameworks eingebunden. Da wir auch ein Bedürfnis für Diagramme hatten verwendeten wir ganze Komponenten-Libraries. PrimeNG und Angular Material kamen dabei zum Einsatz.

## Services

Für jeden Backendkontext existiert ein eigener Service im Frontend, welcher alle Schnittstellen im gegebenen Backendkontext aufruft. Für den Datentransfer aus dem Backend wird der von Angular zu Verfügung gestellter http\*-Client verwendet.

## Komponente

Die App-Komponente dient als Root-Komponente, welche alle anderen Komponenten beinhaltet. Die Komponenten werden nicht als alleinstehende Komponenten dargestellt, sondern immer nur als Teil der App-Komponente. Nur so können mehrere Komponenten gleichzeitig dargestellt werden.

## Routing in Frontend

Der Router ist dafür verantwortlich, dass auf allen Seiten die richtigen Komponenten dargestellt werden. Die entsprechenden Router-Links sind in der URI\*-Bar im Browser sichtbar.

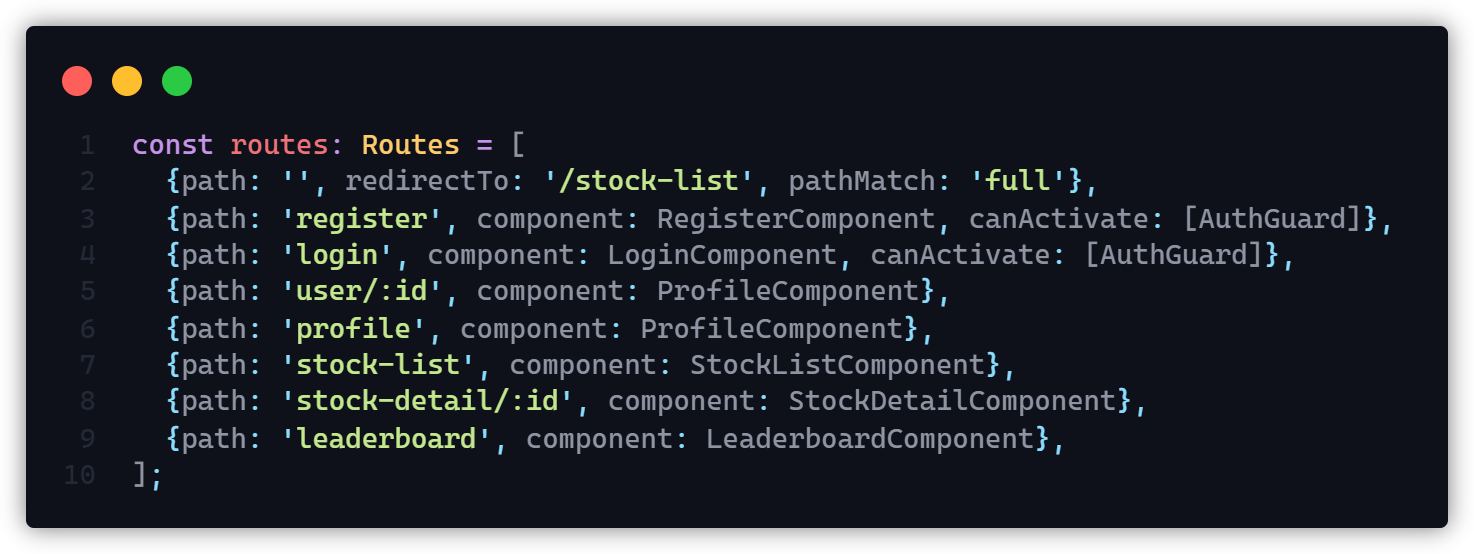


Abbildung 22: Routen im Frontend

Der Code zeigt, wie die einzelnen Pfade mit den entsprechend zu ladenden Komponenten festgehalten. Wenn es im Browser kein Path gibt, wird automatisch auf die Aktien-Übersichtsliste weitergeleitet.

## Auth-Guard

Der Auth-Guard entscheidet, welche Seiten im Frontend durch authentifizierte bzw. unauthentifizierte Benutzer besucht werden dürfen. Zum Beispiel, wenn ein Benutzer eingeloggt ist, soll er nicht mehr die Möglichkeit haben eine Login- oder Registrierungsseite aufzurufen.

# Testkonzept und Protokoll

In diesem Kapitel wird das Testkonzept, die Teststrategie, die Testfälle und dessen Durchführung genauer beschrieben.

## Testziele

|  |  |  |  |
| --- | --- | --- | --- |
| **Nr.** | **Beschreibung** | **Messgröße** | **Priorität\*** |
| *1* | *Schnittstellentest erfolgreich* | *Datenaustausch korrekt* | *M* |
| *2* | *Systemarchitektur erfüllt die Anforderungen des Auftraggebers* | *Übereinstimmung zu geplanter Architektur* | *2* |
| *3* | Webseite ist abrufbar | *Erreichbarkeit der Webseite* | *M* |
| *4* | *Datenintegrität* | *Abdeckung echte Marktdaten* | *3* |
| ***\* Priorität: M = Muss / 1 = hoch, 2 = mittel, 3 = tief*** | | | |

Tabelle 14: Testziele

## Teststrategie

Eine effiziente Strategie ist in unserem Falle eine Agile Teststrategie, bei welcher sehr individuell nach Notwendigkeit und Sinn Tests gemacht werden können.

Vom Entwickler wird direkt die Anforderung aus der Aufgabe vom Jira\*-Board während dem Entwickeln getestet. Dabei werden auch allfällige davon abhängige Funktionen getestet, um sicherzustellen, dass es dort keine negativen Effekte gibt. Der Entwickler pusht nur funktionierenden und getesteten Code auf das GitHub\* Repository.

Die Entwickelnden werden, falls es Sinn macht, auch Unittests schrieben. Modultests oder UI-Tests werden von den Entwicklern nicht implementiert.

Ergänzend zu den User-Stories können wir die Anforderungen aus dem Lastenheft ebenfalls als Testfälle verwenden. So können wir nach Abschluss der Implementierungsphase Abnahmetests durchführen, um zu überprüfen, ob alle Anforderungen umgesetzt wurden. Der Testaufwand ist in den geschätzten Story Punkte inbegriffen und ist somit teil unsere “Definition of Done”.

## Umfang Testumgebung

Die Abnahmetests werden direkt in der Live-Umgebung gemacht. Durch eine vollautomatische CI/CD Pipeline wird jeder Commit auf dem produktiven Server gepusht. Wenn also alle Tests lokal erfolgreich waren, kann direkt auf dem Server überprüft werden, ob in der Produktivumgebung auch alles funktioniert.

## Integration in Projektaktivitäten

Sobald eine brauchbare Version der Applikation vorhanden ist, werden wir weitere Nutzer als sogenannte “Beta-Tester” einladen, um direktes Nutzerfeedback zu erhalten. Daraufhin können weitere Verbesserungen an der Applikation gemacht werden.

## Tests nach den Anforderungen

Die Tests wurden nach dem Testkonzept durchgeführt.

|  |  |
| --- | --- |
| Test Nr. | 1000 |
| Datum | 05.12.2021 |
| Tester | Dániel Polgár |
| Resultat | Erfolgreich |
| Bemerkungen | Der Benutzer kann sich registrieren. |

Tabelle 15: Protokoll Test 1000

|  |  |
| --- | --- |
| Test Nr. | 1001 |
| Datum | 05.12.2021 |
| Tester | Dániel Polgár |
| Resultat | Erfolgreich |
| Bemerkungen | Der Spieler kann sich einloggen. |

Tabelle 16: Protokoll Test 1001

|  |  |
| --- | --- |
| Test Nr. | 1002 |
| Datum | 05.12.2021 |
| Tester | Dániel Polgár |
| Resultat | Erfolgreich |
| Bemerkungen | Die Daten des Erwerbes werden gespeichert. |

Tabelle 17: Protokoll Test 1002

|  |  |
| --- | --- |
| Test Nr. | 1003 |
| Datum | 05.12.2021 |
| Tester | Dániel Polgár |
| Resultat | Erfolgreich |
| Bemerkungen | Die Wertschriften haben entsprechende Bezeichnungen. |

Tabelle 18: Protokoll Test 1003

|  |  |
| --- | --- |
| Test Nr. | 1004 |
| Datum | 05.12.2021 |
| Tester | Dániel Polgár |
| Resultat | Erfolgreich |
| Bemerkungen | Anzahl Spieler ist nicht ausschlaggebend für die Performance Einbussen, die wir durch die Yfinance-API erleiden. |

Tabelle 19: Protokoll Test 1004

|  |  |
| --- | --- |
| Test Nr. | 1005 |
| Datum | 05.12.2021 |
| Tester | Leo Scherer |
| Resultat | Erfolgreich |
| Bemerkungen | Kaufen und Verkaufen erfolgt in unter einer Sekunde. |

Tabelle 20: Protokoll Test 1005

|  |  |
| --- | --- |
| Test Nr. | 1006 |
| Datum | 05.12.2021 |
| Tester | Leo Scherer |
| Resultat | Erfolgreich |
| Bemerkungen | Die Spieler können ohne Konflikte die Aktien erwerben. |

Tabelle 21: Protokoll Test 1006

|  |  |
| --- | --- |
| Test Nr. | 1007 |
| Datum | 05.12.2021 |
| Tester | Leo Scherer |
| Resultat | Erfolgreich |
| Bemerkungen | Der Benutzer hat erfolgreich eine Wertschrift zum angegebenen Preis erwerben können. |

Tabelle 22: Protokoll Test 1007

|  |  |
| --- | --- |
| Test Nr. | 1008 |
| Datum | 05.12.2021 |
| Tester | Leo Scherer |
| Resultat | Erfolgreich |
| Bemerkungen | Der Benutzer hat erfolgreich eine Wertschrift zum angegebenen Preis verkaufen können. |

Tabelle 23: Protokoll Test 1008

|  |  |
| --- | --- |
| Test Nr. | 1009 |
| Datum | 05.12.2021 |
| Tester | Cédric Gasser |
| Resultat | Erfolgreich |
| Bemerkungen | Der Spieler hat nach der Registrierung ein Kapital von 20'000 US-Dollar. |

Tabelle 24: Protokoll Test 1009

|  |  |
| --- | --- |
| Test Nr. | 1010 |
| Datum | 05.12.2021 |
| Tester | Philip Baumann |
| Resultat | Fehlgeschlagen |
| Bemerkungen | Das Leaderboard misst sich nicht am Wert der Aktien, sondern am Wert der Aktien und dem Geld, welches er neben den Aktien noch besitzt. |

Tabelle 25: Protokoll Test 1010

|  |  |
| --- | --- |
| Test Nr. | 1011 |
| Datum | 05.12.2021 |
| Tester | Philip Baumann |
| Resultat | Erfolgreich |
| Bemerkungen | Die Wertschriften wurden innert weniger Sekunden, meist sofort verkauft. |

Tabelle 26: Protokoll Test 1011

|  |  |
| --- | --- |
| Test Nr. | 1012 |
| Datum | 05.12.2021 |
| Tester | Philip Baumann |
| Resultat | Erfolgreich |
| Bemerkungen | Die Daten wurden nach dem Wiederöffnen des Browsers persistiert, sodass der Spieler ohne Unterbrechung nach dem Einloggen weiterfahren konnte. |

Tabelle 27: Protokoll Test 1012

|  |  |
| --- | --- |
| Test Nr. | 1013 |
| Datum | 05.12.2021 |
| Tester | Cédric Gasser |
| Resultat | Erfolgreich |
| Bemerkungen | Die Applikation kann mit Maus und Tastatur, sowie auch alternativ mit einem Touchscreen bedient werden. |

Tabelle 28: Protokoll Test 1013

|  |  |
| --- | --- |
| Test Nr. | 1014 |
| Datum | 05.12.2021 |
| Tester | Cédric Gasser |
| Resultat | Erfolgreich |
| Bemerkungen | Die Applikation ist im Chrome-Browser aufrufbar. Die Verfügbarkeit ist im Zusammenhang mit der Serverwartung verbunden. Grundsätzlich aber 24/7 verfügbar. |

Tabelle 29: Protokoll Test 1014

|  |  |
| --- | --- |
| Test Nr. | 1015 |
| Datum | 05.12.2021 |
| Tester | Philip Baumann |
| Resultat | Erfolgreich |
| Bemerkungen | Die Applikation ist auf dem Lehrlingsserver deploymentfähig und somit auch über dem Browser aufrufbar. |

Tabelle 30: Protokoll Test 1015

Hier mit bestätigt das Entwicklungsteam die Echtheit der Testergebnisse wie sie hier im Testprotokoll festgehalten worden. Sämtliche Tests wurden mit ausführlich durchgeführt und falls nötig mit entsprechenden Kommentar vermerkt.

Philip Baumann:



Dániel Polgár:



Leo Scherer:

A képen olló látható

Automatikusan generált leírás

Cédric Gasser:

A képen nyíl látható

Automatikusan generált leírás

# Setup

Die Applikation kann in Docker und auch Lokal ausgeführt werden. Die Datenbank muss zwingend in einem Docker-Container laufen.

## Docker (Empfohlen)

Installiert sein muss Npm, Docker und Docker-Compose.

Folgende Befehle können in der Konsole ausgeführt werden:

git clone <https://github.com/the-philip-baumann/mock-stock>

cd mock-stock/frontend/

npm run build

cd ../

docker-compose up -d

Danach kann die Applikation unter localhost:80 aufgerufen und verwendet werden.

## Lokal (Für Entwickelnde)

Installiert sein muss Python3, Pip3, Node, Npm, Angular, Docker und Docker-Compose.

Folgende Befehle können in der Konsole ausgeführt werden:

git clone <https://github.com/the-philip-baumann/mock-stock>

cd mock-stock

docker-compose up -d database

cd frontend/

npm run start

Danach kann in einem neuen Konsolenfenster das Backend gestartet werden.

cd mock-stock/backend

pip install -r requirements.txt

python src/app.py

Danach ist die Applikation im Browser unter localhost:4200/ erreichbar.

# Reflexion

Die Arbeit am Backend war eine neue Erfahrung für uns. Vor Arbeitsbeginn hatte noch niemand im Team Erfahrungen mit Flask machen können. Nichtsdestotrotz haben wir uns schnell einarbeiten und auch schnell Fortschritte erzielen können. Probleme hatten wir allerdings trotzdem, denn wir waren auf eine kostenlose Finanz-API von Yahoo angewiesen. Es sollte uns die nötigen Daten liefern, jedoch stellte sich während der Implementierung heraus, dass die Yfinance-API sehr langsam ist. Doch auch dieses Problem konnten wir durch kreatives Denken umgehen.

Ein weiteres Problem, auf welches wir zwischenzeitlich gestossen waren, war CORS. CORS steht für Cross-Origin Request Sharing und ist einer vom Browser implementierter Sicherheitsmechanismus zur Sicherstellung einer Server-Identität. Immer wieder wurden Anfragen, aufgrund einer CORS-Regel Verletzung, von unserem Frontend blockiert. Erst nach intensiver Recherche konnten wir CORS so konfigurieren, dass keine weiteren validen Anfragen geblockt werden.

Hingegen konnten wir bei der von uns angepassten Vorgehensmethode eine gute Bilanz ziehen. Die Weeklies wurden immer effizient durchgeführt. Auch die Kommunikation ausserhalb des Klassenzimmers war sehr zufriedenstellend. Über Gruppenchats konnten wir bei Problemen miteinander in Kontakt treten und sie so als Team lösen.

Während des Projekts konnten wir viele positive Erfahrungen machen. Die Arbeit an einem gemeinsamen Projekt war eine Interessante Erfahrung für uns alle. Von der Planung über die Umsetzung bis hin zum Testen zeigte jedes einzelne Teammitglied viel Engagement und Leidenschaft. Das Knowhow einzelner konnte gut auf die anderen übertragen werden, sodass ein hochwertiges Produkt in kurzer Zeit von Grund auf entwickelt werden konnte. Das Projekt kam aber auch mit diversen Schwierigkeiten, welche wir aber mit viel Ausdauer und Durchsetzungswillen bewältigen konnten. Diese Einstellung spiegelt sich im Endprodukt unserer Arbeit wider.

# Glossar

|  |  |
| --- | --- |
| Begriff | Erklärung |
| API | Application Programming Interface ist ein Set von Funktionen, welche von einem Service zur Verfügung gestellt werden. |
| Caching | Zwischenspeicher für Daten, welche während der Laufzeit häufig aufgerufen werden. Führt zu einem schnelleren Zugriff auf die entsprechenden Daten. |
| DDD | Domain Driven Design, siehe  <https://www.dev-insider.de/was-ist-domain-driven-design-a-724421/> |
| DevOps | DevOps ist eine Philosophie, bei welcher ein Team für die Entwicklung und den Betrieb einer oder mehreren Applikationen zuständig ist. DevOps ist genauer beschrieben im Kapitel «Methodik – DevOps» |
| Docker | Docker ist eine Software für die Container Virtualisierung von Software. |
| Dockerfile | Dockerfiles sind Dateien (meist Konfigurationen), welche einen Build eines mit Docker kompatiblen Container definieren. |
| Flask | Ein Framework für Python RESTful Services, im Kapitel 9 beschrieben. |
| Git | Software zur Versionsverwaltung. Genauer beschrieben im Kapitel «Methodik – Git» |
| GitHub | Ein von Microsoft angebotener Git-Server, auf welches das Projekt gepusht wurde. |
| HTTP | Hyper Text Transfer Protocol – Siehe: <https://developer.mozilla.org/en-US/docs/Web/HTTP> |
| IDE | Integrated Development Environment. Software, in welcher programmiert wird. Beispielsweise IntelliJ, Eclipse oder Webstorm. |
| Impediment | Ein anderes Problem, welches einen aktuellen Task blockiert (z.B. ein fehlendes Umsystem) |
| Iteration | Als Iteration wird ein immer wiederkehrender Prozess verstanden. |
| Jira | Jira ist ein Tool, mit welchem Arbeitsschritte getrackt werden können. |
| MongoDB | Eine Dokument Orientierte Datenbank, wo die Daten ohne Relationen gespeichert werden können. |
| MVP | MVP ist ein Akronym für Minimum Viable Product und das Minimum der benötigten Funktionen einer Software. |
| PA | Praktische Arbeit |
| PO | Product Owner, Rolle in Scrum |
| POC | Proof of Concept, Zeigt mithilfe einer möglichst einfachen Softwarelösung, dass ein grösseres Projekt umsetzbar ist. |
| REST | Representational State Transfer. Schnittstellendefinition, welche Daten über das HTTP/S-Protokoll überträgt. |
| SOAP | “Simple Object Access Protocol”. Wird zur Kommunikation zwischen Maschinen verwendet. Protokoll des BSFW. |
| Stakeholder | Jemand, der die entwickelte / zu entwickelnde Applikation verwendet und ein Interesse daran hat. |
| Staging | Teil eines Workflows, bei den Software zuerst eine (oder mehrere) Testumgebungen durchläuft, bevor die Software auf die Produktion kommt. |
| Scrum | Eine Projektvorgehensmethode, im Kapitel 7 beschrieben. |
| Typescript (TS) | Eine Programmiersprache, welche zu JavaScript transpiliert wird. |
| UI | User Interface. Die Schnittstelle zum Endbenutzer, meist eine grafische Oberfläche. |
| URI | Uniform Rescource Identifier, siehe: <https://en.wikipedia.org/wiki/Uniform_Resource_Identifier> |
| Yfinance | Python Modul, dass Echtzeit Aktiendaten zur Verfügung stellt. Ist eine Abkürzung für Yahoo Finance. |

*Tabelle 31: Glossar*

# Quellenverzeichnis

BAG. (2021). *Bundesamt für Gesundheit*. Von Informationen Coronavirus: https://www.bag.admin.ch/bag/en/home/krankheiten/ausbrueche-epidemien-pandemien/aktuelle-ausbrueche-epidemien/novel-cov/so-schuetzen-wir-uns.html abgerufen

diagrams.net. (29. 07 2020). *diagrams.net.* Von Create C4 Models and diagrams: https://www.diagrams.net/blog/c4-modelling abgerufen

Scrum.org. (13. 12 2021). *Scrum.org*. Von Scrum.org: https://www.scrum.org/resources/what-is-scrum abgerufen

# Abbildungsverzeichnis

[Abbildung 1: Phasenplan 7](#_Toc90327884)

[Abbildung 2: Benutzerkonto erstellen 11](#_Toc90327885)

[Abbildung 3: Anmeldung 12](#_Toc90327886)

[Abbildung 4: Aktienübersicht 13](#_Toc90327887)

[Abbildung 5: Aktien Detailansicht 14](#_Toc90327888)

[Abbildung 6: Rangliste 15](#_Toc90327889)

[Abbildung 7: Aktien von einem anderen oder eingeloggten Benutzer 1 16](#_Toc90327890)

[Abbildung 8: Aktien von einem anderen oder eingeloggten Benutzer 2 17](#_Toc90327891)

[Abbildung 9: Projektstrukturplan 18](#_Toc90327892)

[Abbildung 10: Backendarchitektur 19](#_Toc90327893)

[Abbildung 11: OpenAPI /stocks 20](#_Toc90327894)

[Abbildung 12: OpenAPI /me 20](#_Toc90327895)

[Abbildung 13: OpenAPI /users 20](#_Toc90327896)

[Abbildung 14: OpenAPI /auth 21](#_Toc90327897)

[Abbildung 15: Legende C4 Diagramm (diagrams.net, 2020) 26](#_Toc90327898)

[Abbildung 16: C4 Diagramm System Context 27](#_Toc90327899)

[Abbildung 17: C4 Diagramm Component Diagramm 28](#_Toc90327900)

[Abbildung 18: Serialisierungsmethode 29](#_Toc90327901)

[Abbildung 19: User-Loader definieren 29](#_Toc90327902)

[Abbildung 20: Geschützte Route 30](#_Toc90327903)

[Abbildung 21: Caching-Implementierung 30](#_Toc90327904)

[Abbildung 22: Routen im Frontend 31](#_Toc90327905)

# Tabellenverzeichnis

[Tabelle 1: Experten 6](#_Toc90327906)

[Tabelle 2: Beteiligte Personen 6](#_Toc90327907)

[Tabelle 3: Userstories - Datenstruktur 9](#_Toc90327908)

[Tabelle 4: Userstories - Datenmengen 9](#_Toc90327909)

[Tabelle 5: Userstories - Applikationsanforderungen 10](#_Toc90327910)

[Tabelle 6: Userstories - Bedienung 10](#_Toc90327911)

[Tabelle 7: Userstories - Umgebung 10](#_Toc90327912)

[Tabelle 8: Risiko - Datenverlust 21](#_Toc90327913)

[Tabelle 9: Risiko - Coronavirus 22](#_Toc90327914)

[Tabelle 10: Risiko - Fehlerhafter Zeitplan 22](#_Toc90327915)

[Tabelle 11: Risiko Fehler in 3. Party Framework 22](#_Toc90327916)

[Tabelle 12: Risikovergleich 23](#_Toc90327917)

[Tabelle 13: Auswertung der Risiken 23](#_Toc90327918)

[Tabelle 14: Testziele 32](#_Toc90327919)

[Tabelle 15: Protokoll Test 1000 33](#_Toc90327920)

[Tabelle 16: Protokoll Test 1001 33](#_Toc90327921)

[Tabelle 17: Protokoll Test 1002 33](#_Toc90327922)

[Tabelle 18: Protokoll Test 1003 33](#_Toc90327923)

[Tabelle 19: Protokoll Test 1004 33](#_Toc90327924)

[Tabelle 20: Protokoll Test 1005 34](#_Toc90327925)

[Tabelle 21: Protokoll Test 1006 34](#_Toc90327926)

[Tabelle 22: Protokoll Test 1007 34](#_Toc90327927)

[Tabelle 23: Protokoll Test 1008 34](#_Toc90327928)

[Tabelle 24: Protokoll Test 1009 34](#_Toc90327929)

[Tabelle 25: Protokoll Test 1010 35](#_Toc90327930)

[Tabelle 26: Protokoll Test 1011 35](#_Toc90327931)

[Tabelle 27: Protokoll Test 1012 35](#_Toc90327932)

[Tabelle 28: Protokoll Test 1013 35](#_Toc90327933)

[Tabelle 29: Protokoll Test 1014 35](#_Toc90327934)

[Tabelle 30: Protokoll Test 1015 36](#_Toc90327935)

[*Tabelle 31: Glossar* 39](#_Toc90327936)

# Abbildungsverzeichnis

[Abbildung 1: Phasenplan 7](#_Toc90327937)

[Abbildung 2: Benutzerkonto erstellen 11](#_Toc90327938)

[Abbildung 3: Anmeldung 12](#_Toc90327939)

[Abbildung 4: Aktienübersicht 13](#_Toc90327940)

[Abbildung 5: Aktien Detailansicht 14](#_Toc90327941)

[Abbildung 6: Rangliste 15](#_Toc90327942)

[Abbildung 7: Aktien von einem anderen oder eingeloggten Benutzer 1 16](#_Toc90327943)

[Abbildung 8: Aktien von einem anderen oder eingeloggten Benutzer 2 17](#_Toc90327944)

[Abbildung 9: Projektstrukturplan 18](#_Toc90327945)

[Abbildung 10: Backendarchitektur 19](#_Toc90327946)

[Abbildung 11: OpenAPI /stocks 20](#_Toc90327947)

[Abbildung 12: OpenAPI /me 20](#_Toc90327948)

[Abbildung 13: OpenAPI /users 20](#_Toc90327949)

[Abbildung 14: OpenAPI /auth 21](#_Toc90327950)

[Abbildung 15: Legende C4 Diagramm (diagrams.net, 2020) 26](#_Toc90327951)

[Abbildung 16: C4 Diagramm System Context 27](#_Toc90327952)

[Abbildung 17: C4 Diagramm Component Diagramm 28](#_Toc90327953)

[Abbildung 18: Serialisierungsmethode 29](#_Toc90327954)

[Abbildung 19: User-Loader definieren 29](#_Toc90327955)

[Abbildung 20: Geschützte Route 30](#_Toc90327956)

[Abbildung 21: Caching-Implementierung 30](#_Toc90327957)

[Abbildung 22: Routen im Frontend 31](#_Toc90327958)

# Selbstständigkeitserklärung

Hiermit erklären wir, dass wir die vorliegende Arbeit sowie das darin dokumentierte Produkt selbstständig und ohne unerlaubte fremde Hilfe angefertigt, keine anderen als die angegebenen Quellen und Hilfsmittel verwendet und die den verwendeten Quellen und Hilfsmitteln wörtlich oder inhaltlich entnommenen Elemente (bspw. Code) als solche kenntlich gemacht haben.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | **Ort, Datum** | **Unterschrift** | **Name** | | Kriens, 13.12.2021 |  | Philip Baumann | | Attinghausen, 13.12.2021 |  | Dániel Polgár | | Ebikon, 13.12.2021 | A képen olló látható  Automatikusan generált leírás | Leo Scherer | | Lungern, 13.12.2021 | A képen nyíl látható  Automatikusan generált leírás | Cédric Gasser | |  |
|  |  |

# Anhang

Folgender Link führt zum GitHub Repository, wo unser Code abgelegt ist.

* https://github.com/the-philip-baumann/mock-stock