Projektdokumentation

Wertpapiersimulation

Ausbildungsberuf

Fachinformatiker/-in Fachrichtung: Anwendungsentwicklung

Berufsschule

Brühlwiesenschule

Auszubildende

Danny Nicolay Walter Binder, Cristian Felipe Castillo-Barrero, Moritz Heinke

E-Mail-Adresse: moritz.heinke@deutsche-boerse.com

Betrieblicher Ansprechpartner: Jan Patrick Drehwald

E-Mail-Adresse: jan.patrick.drehwald@clearstream.com | Telefon: +49 69 211-1 41 41

Thema der Projektarbeit:

Entwicklung und Implementierung einer plattformunabhängigen Simulation eines Wertpapierdepots.

Inhaltsverzeichnis

[1 Einleitung 3](#_Toc177662760)

[1.1 Vorwort 3](#_Toc177662761)

[1.2 Projektauftrag 3](#_Toc177662762)

[1.3 Projektumfeld 3](#_Toc177662763)

[2 Projektplanung 3](#_Toc177662764)

[2.1 Funktionale Anforderungen 3](#_Toc177662765)

[2.2 Ressourcen und Kostenplanung 4](#_Toc177662766)

[2.2.1 Zeitliche Planung 4](#_Toc177662767)

[2.2.2 Kostenplanung 4](#_Toc177662768)

[3 Projektdurchführung 4](#_Toc177662769)

[3.1 Vorbereitung 4](#_Toc177662770)

[3.2 Defintion 4](#_Toc177662771)

[3.3 Entwurf 5](#_Toc177662772)

[3.4 Implementation 5](#_Toc177662773)

[3.4.1 Entwicklungsumgebung 5](#_Toc177662774)

[3.4.2 Einrichtung der Datenbank 5](#_Toc177662775)

[3.4.3 Verwendung von yfinance 5](#_Toc177662776)

[3.4.4 Aufbau der Webanwendung 6](#_Toc177662777)

[3.5 Testen 6](#_Toc177662778)

[3.5.1 Komponententest 6](#_Toc177662779)

[3.5.2 Integration Test 6](#_Toc177662780)

[3.6 Installation 7](#_Toc177662781)

[4 Fazit 7](#_Toc177662782)

[4.1 Ist/Soll Vergleich 7](#_Toc177662783)

[4.2 Ausblick 7](#_Toc177662784)

[4.2.1 Weitere Funktionalitäten 8](#_Toc177662785)

[4.3 Gewonnene Erkenntniss 8](#_Toc177662786)

[5 Glossar 8](#_Toc177662787)

[6 Anhang 9](#_Toc177662788)

[7 Quellen 14](#_Toc177662790)

1. Einleitung

## Vorwort

Verwendete Fachbegriffe und Abkürzungens sind im Kapitel 5 Gloassr erwähnt und erklärt.

## Projektauftrag

Der Auftrag besteht in der Entwicklung einer Simulationsplatform zum spielerischen Erlernen von praxisnahem und lehrreichen Verständnis für den Handel mit Wertpapieren. Die Plattform ermöglicht es den Benutzern, mithilfe von virtuellem Kapital, Käufe und Verkäufe von Wertpapieren durchzuführen. Das Gesamtkapital des Benutzers wird in Echtzeit basierend auf den aktuellen Börsenkursen berechnet und angepasst. Darüber hinaus soll die Simulation die Dynamiken des echten Börsenhandels visuell abbilden.

## Projektumfeld

Das Projekt wird innerhalb der Brühlwiesenschule, für die Lernfelder zehn bis zwölf, innerhalb des Schulblocks vom 26.08.2024 – 20.09.2024 durchgeführt. Innerhalb des Gruppenprojektes werden die bislang erlernten Fertigkeiten der letzten beiden Lehrjahre genutzt. Neue Konzepte, wie Model-View-Trennung, Barrierefreiheit und Plattformübergreifende Kompatibilität werden durch Eigeninitiative erlernt und angewandt. Das Projekt wird im folgenden Schulblock durch eine Präsentation und eine Live-Demo vorgestellt.

# Projektplanung

## Funktionale Anforderungen

Im Vorfeld der Anforderungsbesprechung war es notwendig, das Projekt gründlich zu durchdenken und zu planen. Dabei ging es darum, eine klare Vision für die Simulationsplattform zu entwickeln, Es mussten Überlegungen angestellt werden, wie die Benutzererfahrung gestaltet werden kann, welche Funktionen integriert werden sollen und wie die Echtzeit-Berechnung des Kapitals sowie die visuelle Darstellung der Börsendynamiken umgesetzt werden können. Diese Vorarbeit war entscheidend, um sicherzustellen, dass alle Anforderungen und Ziele des Projekts klar definiert und realisierbar sind

## Ressourcen und Kostenplanung

### Zeitliche Planung

Der Zeitplan des Projekts wurde aus dem Projekt Antrag übernommen und definiert den Ablauf des geplanten Projekts (Tabelle 2: Zeitplanung). Wegen der konkreten Anforderungen und dem zeitlichen Horizont eignet sich das erweiterte Wasserfallmodell als Vorgehensmodell. Die geplanten Abläufe werden nach dem erweitertem Wasserfallmodell nacheinander abgearbeitet können aber mit darüberliegenden Phasen interagieren.

### Kostenplanung

Die Gesamtkosten für das Projekt setzten sich aus Personalkosten sowie Dienstelistungskosten zusammen. Die Personalkosten sind von HR festgelegt und belaufen sich für den Arbeitgeber auf 11€ pro Stunde. Die angegebenen Dienstleistungskosten deckt die Nutzung der GCP Infrastruktur ab, die für die Implementierung und das Testen des Programms notwendig waren. Der Stundensatz für die eingesetzte E2-Standardmaschinentypen (4 virtuelle CPUs und 16GB Arbeitsspeicher) ist der [Preistabelle](https://cloud.google.com/compute/all-pricing?hl=de) für die erwähnten Spezifikatioen mit ≈ 0,18$ /0,16€ zu entnehmen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Personal** | **Stundenlohn** | **Stunden** | **Kosten** |
| Auszubildender | 11,00€ | 232 | 2552€ |
| Google-Cloud | 0,16€ | 142 | 22,72€ |
| **Summe** |  |  | **2574,72€** |

Tabelle 1 : Kostenplanung

1. Projektdurchführung

## Vorbereitung

Um eine effiziente und transparente Verwaltung unserer Projektdokumentation zu gewährleisten, setzen wir auf ein Versionskontrollsystem. Hierfür erstellen wir ein Git-Repository. Git ist ein Tool zur Versionskontrolle, das es ermöglicht, Änderungen an unseren Dokumenten im Laufe der Zeit nachzuverfolgen und zu verwalten.

## Defintion

In der Projektdurchführung haben wir uns entschieden, die Yahoo Finance API als Datenquelle zu nutzen und ein Python Backend mit Flask zu entwickeln. **Als Datenbank kommt SQLite zum Einsatz.**

**Yahoo Finance API:** Die Yahoo Finance API bietet einen umfangreichen und leicht zugänglichen Datensatz zu Finanzinformationen. Sie stellt historische Kursdaten und diverse Finanzkennzahlen für eine Vielzahl von Unternehmen bereit. Die API ist gut dokumentiert und wird von einer großen Entwicklergemeinde genutzt, was die Integration in unser Projekt erleichtert.

**Python und Flask:** Python ist eine flexible Programmiersprache, die sich durch ihre Lesbarkeit und die große Auswahl an Bibliotheken auszeichnet. Flask, ein leichtgewichtiges Web-Framework für Python, ermöglicht es, einfache Webanwendungen zu entwickeln. Im Gegensatz zu Django ist es schlanker und wegen des geringen Umfangs schneller zu lernen.

Als Datenbank haben wir uns für SQLite entschieden, da diese für kleinere Projekte wie unseres ideal ist und sich leicht in Python-Anwendungen integrieren lässt. Des Weiteren ist SQLite kostenlos und Kenntnisse sind durch anderes Projekt bereits vorhanden.

Die technische Produktumgebung sowie die abgesprochenen Musskriterien wie Wunschkriterien wurden im Pflichtenheft festgehalten.

## Entwurf

Für die Konzeptionierung des Projekts haben wir eine detaillierte Modellierung durchgeführt. Use-Case-Diagramme dienten dazu, die funktionalen Anforderungen aus Benutzersicht zu erfassen und zu dokumentieren. Dabei wurden alle Use-Cases alle verfügbaren Seiten berücksichtigt. In dieser Doku sind aus Platz und Übersichtsgründe die zwei einflussreichsten mit den meisten Funktionalitäten enthalten. (Abbildung 3: UseCase StartseiteAbbildung 4: UseCase User Depot) Diese Diagramme geben einen klaren Überblick über die von den Benutzern erwarteten Funktionen. Das ERM visualisiert die Entitäten und ihre Beziehungen zueinander. Auf Basis des ERM (Abbildung 1: ERM) konnten wir die Datenbanktabellen entwerfen. Ergänzend dazu haben wir ein Aktivitätsdiagramm (Abbildung 2: Aktivitätsdiagramm Login(Abbildung 2: Aktivitätsdiagramm Login) erstellt, um eine Teilalgorithmik der der Anwendung abzubilden. Durch diese Modellierungstechniken unterstreicht das Vorgehen nach Top-down Prinzip.

## Implementation

### Entwicklungsumgebung

Die Einrichtung einer virtuellen Python-Umgebung und die Installation der erforderlichen Pakete waren entscheidende Schritte für die Implementation. Durch diese Maßnahmen haben besteht eine kontrollierte Umgebung, in der die Abhängigkeiten der Anwendung präzise verwalten können. Dies minimiert das Risiko von Konflikten, die durch unterschiedliche Paketversionen oder globale Installationen entstehen könnten. Notwendig ist die Installation von python3-venv weitere Installationsschritte sind im Kapitel Installation erwähnt (Installation) erwähnt.

### Einrichtung der Datenbank

Das Fundament unserer Datenbankstruktur bildet die Datei schema.sql. Diese Datei fungiert als Gerüst und definiert die Tabellen, Spalten und Beziehungen, aus denen die Datenbank besteht. Die Methode init\_db\_command() erstellt eine Datenbank, sofern keine Datenbank vorhanden ist, indem es die sqlite DB-API verwendet und mit dem Keyword detect\_types=sqlite3.PARSE\_DECLTYPES die schema.sql einliest. Zu erwähnen ist, dass der Befehl ON DELETE CASCADE Löschenanomalien verhindert indem beim Löschen eines Datensatzes in einer Tabelle eine eine Kaskade von Löschvorgängen in anderen, verknüpften Tabellen ausgelöst wird. Desweiteren werden durch den init-Befehl die Stammdaten, der angebotenen Produkte, in der Tabelle Produkte eingetragen.

### Verwendung von yfinance

Die Aktualisierung der Chartbilder erfolgt primär über die Funktion yfinance.download(),die eine Schnittstelle zur Yahoo Finance API bietet. Durch Übergabeparameter können spezifische Wertpapiere und Zeiträume ausgewählt werden. In Kombination mit der Visualisierungsbibliothek plotly können so historische Kursdaten visuell dargestellt werden.Für die Abfrage des aktuellen Kurses einzelner Wertpapiere wird hingegen die Funktion yfinance.Ticker eingesetzt. Diese Funktion ist effizienter für die Beschaffung einzelner Kursdatenpunkte und liefert die Informationen in einem strukturierten Format, das leicht weiterverarbeitet werden kann.

### Aufbau der Webanwendung

In dem Projekt ist Flask die Kernkomponente, um Webanwendung zu erstellen Entscheidungen war die Implementierung von Routen, die es uns ermöglicht haben, verschiedene URL-Pfade unserer Anwendung gezielt zu steuern und zu verarbeiten. Durch die Definition von Routen konnten wir spezifische Funktionen zu den jeweiligen URLs zuordnen. Die unterschiedlichen HTTP-Methoden wie GET und POST strukturieren die Webanwendung. Bei, Die GET-Anfragen haben die Formularseite geladen, während die POST-Anfragen die übermittelten Daten verarbeitet und die Antwort zurückgegeben haben.

Über die Template-Engine Jinja2 werden die HTML-Seiten dynamisch mit den Daten aus der jeweiligen Route befüllt. Der Datenaustausch zwischen Frontend und Backend ist im JSON-Format, was einem Python-Dictionary ähnelt. So konnte eine saubere Trennung zwischen Logik und Präsentation beibehalten werden.

An einem Flaskobjekt(app) kann die Methode app.run() aufgerufen werden, welche die Webanwendung zum laufen bringt (in Development Localhost IP-Adreese: 127.0.01). Zusätzlich kann mit dem Keyword ssl\_context=('cert.pem', 'key.pem') ein SSL-Zertifikat und den dazugehörigen privaten Schlüssel übergeben werden. Sobald eine Anfrage eingeht, überprüft app, ob eine sichere Verbindung gewünscht wird (HTTPS). Ist dies der Fall, wird mit dem bereitgestellten SSL-Zertifikat und dem Schlüssel die Verbindung verschlüsselt. Im Development wird der cert.pem und key.pem selbstsigniert, sodass andere Browser diese Verbindung nicht als sicher ansehen. In einer Produktionsumgebung müssten diesen Dateien von einer Authentifikation signiert werden. Weitere Informationen stehen im Kapitel Installation (3.6).

## Testen

Um die Qualität der Anwendung sicherzustellen, wurde als erste Testmaßnahme ein Abnahmetest durchgeführt. Dabei wurde die Anwendung auf einem unabhängigen System nach Installationsanweisung (3.6)vorbereitet und anschließend ausgeführt. Die die Anforderungen aus dem Projektantrag sowie Lastenheft wurden überprüft.

Durch Vorerfahrung haben wir uns dazu entschieden das Python-Modul unittest zu verwenden, um Testfälle zu definieren und automatisiert laufen zu lassen. Diese Testfälle erzeugen beim Start eine eigene Testdatenbank, sodass die Tests unabhängig von der Produktion ausgeführt werden können. Dafür erbt die Testklasse von unittest.TestCase, sodass die Testklasse die Funktionalität von unitest enthält.Eine setup()-Methode kreiert eine Testumgebung. Mithilfe von Assertions (z.B. assertEqual, assertTrue) vergleichen wir das erwartete Ergebnis mit dem tatsächlichen Ergebnis der Funktion.

### Komponententest

Der Komponenentest dienen dazu, einzelne, isolierte Funktionen oder Methoden zu testen. In unserem Testbereich liegt die user\_db\_requests.py-Datei, die Schnittstelle der Flask.Routen und der Datenbank ist.

### Integration Test

Integrationstests überprüfen, wie verschiedene Komponenten einer Anwendung zusammenarbeiten. Dafür werden GET und POST Request erstellt und an die Webanwendung gesendet. Ein POST Request übermittelt Informationen an die Routen im JSON-Format. Die response enthält ein Feedback, inwiefern der Request erfolgreich war. So können Login sowie Registrierung getestet werden. Der GET Request testet die Erreichbarkeit aller Routen. Für die Erreichbarkeitsprüfung des Depotbereich ist ein eingeloggter User Voraussetzung.

## Installation

Damit andere Menschen die Webanwendung nutzen können, muss die Anwendung auf einem Server aufgesetzt werden, der ständig online ist und von überall aus erreichbar ist.

Da der Quellcode in Python geschrieben ist, ist die Installation von einem Python-Interpreter Vorraussetzung. Darüber hinaus müssen benötigten Pakte und Module installiert werden. Die zu installiede Pakete sind im requirements.txt File aufgelistet. Dabei besteht die Möglichkeit die Installation in einer virtuellen Umgebung vorzunehmen um ungewünschte Seiteneffekte zu vermeiden. Dies geht mit den Befehlen:

* pip install virtualenv
* python3 -m venv
* source .venv/bin/activate

Um eine verschlüsselte Verbindung zu gewährleisten ist es zwigend notwenig im Vorfeld authentifizierte Zertifkate(cert.pem) und einen private Key(key.pem) in das Verzeichnis instance/certs abzulegen. Der Anbieter Let’s Encrypt bietet mit dem Programm Certbot eine kostenlose Möglickeit an.

Als nächstes muss die Datenkbank mit ihren Stammdaten aufgesetzt werden:

* flask --app app init-db

Die Anwendung kann im Development mit Debug-Level und ohne vorhandene Zeritfikate gestratet werden:

* flask --app main.py --debug run

Die Anwenung sollte mit diesem Befehl in Produktion aufgesetzt werden:

* python3 -m app.main

# Fazit

## Ist/Soll Vergleich

Während der Durchführung des Projekts gab es an wenigen Stellen Abweichungen zum Projektantrag. Der geplante zeitliche Ablauf hat sich leicht verschoben, jedoch ohne den gesamten Zeitrahmen zu verändern (). Die Implementation des Projekts hat durch eingetroffene Probleme mit GitHub mehr Zeit in Anspruch genommen als geplant war. Hauptursache waren Merge-Konfilkte, \_\_pycache\_\_ Datein und unterschiedliche Versionen der Datenbank. Beim Testen und dem Probelauf lief die Durchführung schneller als geplant ab und es konnte Zeit zurückgewonnen werden. Auch die

## Ausblick

Aufgrund der Abschlussprüfungen muss das Projekt TradeHub pausiert werden. Wir laden jeden ein, mitzumachen und gemeinsam an diesem Projekt zu wachsen. Das [GitHub-Repository](https://github.com/McLovin-81/trade-hub/tree/main) ist öffentlich und für jeden frei zugänglich.

### Weitere Funktionalitäten

Für eingeloggte User soll es möglich sein in dem Fenster Detailpage (Darstellung des Aktienkurses) direkt Aktien kaufen oder verkaufen zu können. Damit kann der Benutzer schneller handeln und spart sich so das Navigieren zum Ordermanager und das Eingeben des Aktienamen.

Eine vielversprechende Erweiterung für unser Projekt wäre die Integration eines SMTP-Servers. Dies würde es uns ermöglichen, E-Mails zu versenden und somit eine Vielzahl von Anwendungsfällen zu realisieren:

* Passwort-Wiederherstellung: Bei einem vergessenen Passwort könnte dem Nutzer automatisch eine E-Mail mit einem Link zum Zurücksetzen des Passworts zugesendet werden.
* Bestätigung der Registrierung: Neue Nutzer könnten eine E-Mail erhalten, um ihre E-Mail-Adresse zu bestätigen und den Registrierungsprozess abzuschließen.

## Gewonnene Erkenntniss

Die Arbeit an diesem Projekt war für uns eine wertvolle Erfahrung. Wir haben alle gelernt, wie wichtig es ist, auf die Stärken und Schwächen der anderen Teammitglieder einzugehen. Kommunikation war in unserem Fall unvermeidlich wichtig, da jeder unterschiedliche Ansichten und Ideen hat. Die Erkenntnis, dass nicht alles planbar ist und dass Flexibilität eine Schlüsselqualifikation ist.

# 5 Glossar

**GCP**: Die Google Cloud Platform, kurz GCP, ist eine umfassende Sammlung von Cloud-Computing-Diensten, die von Google bereitgestellt wird.

**API**: Application Programming Interface ist im Grunde eine Schnittstelle, die es verschiedenen Softwareanwendungen ermöglicht, miteinander zu kommunizieren.

**ERM:** Entity-Relationship-Modell. Es ist eine grafische Darstellung, die dazu dient Datenstruktur zu beschreiben.

**Top-down:** Durch die Abstraktion am Anfang erhält man einen klaren Überblick über das gesamte System und kann komplexe Probleme besser verstehen.

**Stammdaten:** Stammdaten liefern grundlegende Information und werden in der Zukunft wenig bis garnicht geändert.

**SSL**: Secure Sockets Layer ist ein Sicherheitsprotokoll, das die Kommunikation über das Internet verschlüsselt.

**Localhost:** Diese Adresse wird als Loopback-Adresse bezeichnet und leitet Anfragen zurück an das eigene System. Keine anderen Nutzer können die Webanwendung erreichen.

**IP-Adresse**: Internet Protocol-Adresse ist eine numerische Kennung, die jedem Gerät in einem Netzwerk zugewiesen wird, um dessen Standort und Kommunikation zu ermöglichen.

**SMTP-Servers:** Simple Mail Transfer Protocol, ist das zugrunde liegende Protokoll, das den Versand von E-Mails im Internet regelt. Es definiert die standardisierten Regeln und Verfahren, nach denen E-Mail-Nachrichten zwischen verschiedenen Mailservern übertragen werden.

1. Anhang

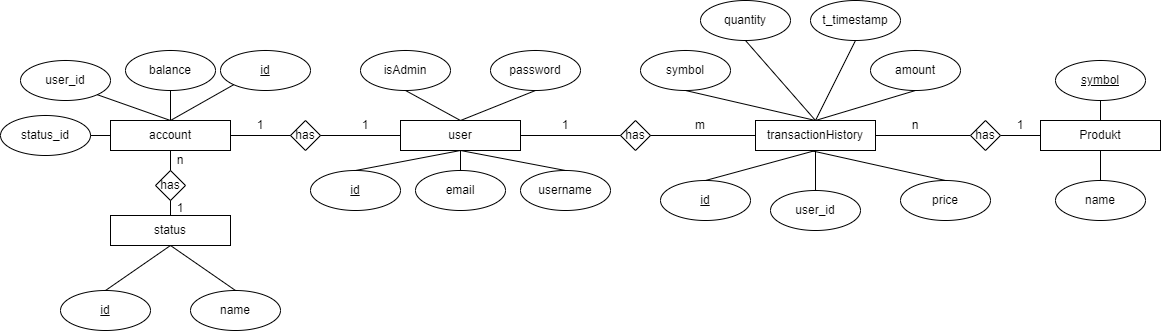


Abbildung 1: ERM

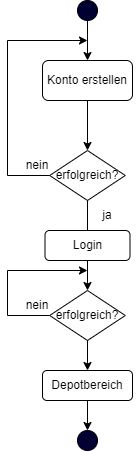


Abbildung 2: Aktivitätsdiagramm Login



Abbildung 3: UseCase Startseite



Abbildung 4: UseCase User Depot

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Aufgabe | Cristian (Std.) | Danny (Std.) | Moritz (Std.) | Gesamtstunden  pro Phase (Std.) |
| Planung | Projektantrag | 5 | 7 | 6 | 18 |
| Definition | Pflichtenheft | 1 | 1 | 1 | 22 |
|  | Auswahl Schnittstellen | 2 | 2 | 2 |  |
|  | Auswahl Software | 1 | 1 | 1 |  |
|  | Auswahl Frameworks | 1 | 1 | 1 |  |
|  | Auswahl Bibliotheken | 1 | 1 | 1 |  |
|  | Erstellung Softwareplan | 1 | 2 | 1 |  |
| Entwurf | Use-Case Diagramm | 1 | 1 | 1 | 20 |
|  | Klassendiagramme | 3 | 2 | 3 |  |
|  | Aktivitätsdiagramm | 2 | 3 | 2 |  |
|  | ER-Modell | - | 1 | 1 |  |
| Implementierung | Klassen | 3 | 3 | 4 | 92 |
|  | Methoden | 7 | 6 | 7 |  |
|  | Datenbank aufsetzen, Kommunikation zum Backend, Definition der Queries | - | - | 12 |  |
|  | User Interface Design | 30 | - | - |  |
|  | API-Anbindung | - | 20 | - |  |
| Tests | Testplanung, -vorbereitung und -spezifikation | 13 | 13 | 14 | 50 |
|  | Komponententests | 1 | 1 | 2 |  |
|  | Integrationstests | 1 | 1 | 1 |  |
|  | Systemtests | 1 | 1 | 1 |  |
| Dokumentation | Dokumentation | 10 | 10 | 10 | 30 |
|  |  |  |  |  | 232 |

Tabelle 2: Zeitplanung

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Phase | Aufgabe | Cristian (Std.) | Danny (Std.) | Moritz (Std.) | Gesamtstunden  pro Phase (Std.) |
| Planung | Projektantrag | 5 | 7 | 6 | 18 |
| Definition | Pflichtenheft | 1 | 1 | 1 | 22 |
|  | Auswahl Schnittstellen | 2 | 2 | 2 |  |
|  | Auswahl Software | 1 | 1 | 1 |  |
|  | Auswahl Frameworks | 1 | 1 | 1 |  |
|  | Auswahl Bibliotheken | 1 | 1 | 1 |  |
|  | Erstellung Softwareplan | 1 | 2 | 1 |  |
| Entwurf | Use-Case Diagramm | 1 | 1 | 1 | 20/6 |
|  | Klassendiagramme | 3/0 | 2/0 | 3/0 |  |
|  | Aktivitätsdiagramm | 2/0 | 3/0 | 2/1 |  |
|  | ER-Modell | - | 1 | 1 |  |
| Implementierung | Klassen | 3 | 3 | 4 | 92/117 |
|  | Methoden | 7 | 6/10 | 7 |  |
|  | Datenbank aufsetzen, Kommunikation zum Backend, Definition der Queries | 0/5 | 0/12 | 12/15 |  |
|  | User Interface Design | 30 | - | - |  |
|  | API-Anbindung | - | 20 | 0/1 |  |
| Tests | Testplanung, -vorbereitung und -spezifikation | 13/5 | 13 | 14 | 50/39 |
|  | Komponententests | 1/0 | 1 | 2 |  |
|  | Integrationstests | 1/0 | 1 | 1 |  |
|  | Systemtests | 1/0 | 1 | 1 |  |
| Dokumentation | Dokumentation | 10/5 | 10/5 | 10/40 | 30 |
|  |  |  |  |  | 232 |

Tabelle 3: Sol/Ist Zeitplanung

# 7 Quellen

GCP-Preistabelle: https://cloud.google.com/compute/all-pricing?hl=de

Certbotbot: https://certbot.eff.org/

GitHub: https://github.com/McLovin-81/trade-hub/tree/main