Kryptowährung API

Entwicklung einer Web-App zur Abfrage von Kryptowährungen

Info:
Diese Abgabe überschreibt die Abgabe des Frontend (Modul 294), da diese Abgabe die ergänzte Projektdokumentation beinhaltet und zudem noch das Frontend des Modul 294.

Berufsschule: WISS-Schulen für Wirtschaft Informatik Immobilien AG, Bern

Module: 294 – Frontend einer interaktiven Webapplikation realisieren

295 – Backen für Applikationen realisieren

Student: Jason Andrea Termine

Inhalt

1. Projektidee ... 4
   1.1 Elevator Pitch ... 4
   1.2 Projektbeschreibung ... 4
2. Anforderungskatalog ... 5
   2.1 Funktionale Anforderungen ... 5
   2.2 Technische Anforderungen ... 5
3. Datenbank ERD ... 6
4. Storyboard ... 7
   4.1 Home ... 7
   4.2 Suche ... 7
5. Screen-Mockups ... 8
6. REST-Schnittstellen ... 9
   6.1 Datenstruktur ... 9
   6.2 Endpoints ... 10
   6.3 HTTP-Requests mit Insomnia ... 11
   6.4 Maven und Konfiguration ... 12
   6.5 Datenbankstruktur und Kommunikation ... 13
7. Testplan (Frontend) ... 14
8. Testplan (Backend) mit Insomnia ... 16
9. Installationsanleitung (Datenbank) ... 18
   9.1 MySQL installieren ... 18
   9.2 Projekt herunterladen und entpacken ... 18
   9.3 Import Berechtigung anpassen ... 18
   9.4 Datenbank importieren ... 18
10. Installationsanleitung (Backend) ... 19
   10.1 JDK 17 / Java installieren ... 19
   10.2 Projekt herunterladen und entpacken ... 19
   10.3 Eclipse IDE öffnen ... 19
   10.4 Projektordner in Eclipse IDE importieren ... 19
   10.5 Backend starten ... 19
   10.6 Sicherstellen ... 19
11. Installationsanleitung (Frontend) ... 20
   11.1 Node.js installieren ... 20
   11.2 Projekt herunterladen und entpacken ... 20
   11.3 Visual Studio Code öffnen ... 20
   11.4 Projektordner öffnen ... 20
   11.5 Terminal öffnen ... 20
   11.6 Projekt starten ... 20
   11.7 Unit Tests ... 20

12 Installationsanleitung (Insomnia) 21
12.1 Insomnia installieren 21
12.2 Projekt herunterladen und entpacken 21
12.3 Insomnia öffnen und die Konfigurationsdatei importieren 21
13 Hilfestellungen und Quellen 22
13.1 Künstliche Intelligenz 22
13.2 Tools und Programme 22
13.3 API 22
13.4 Dokumentation 22
14 Abbildungsverzeichnis 23
15 Tabellenverzeichnis 23

1. Projektidee

1.1 Elevator Pitch

In meinem Projekt entwickle Ich eine benutzerfreundliche Anwendung, die es Nutzern ermöglicht, in Sekundenschnelle den aktuellen Preis, die Bezeichnung und der Rang von Kryptowährungen abzurufen. Die Anwendung umfasst eine vordefinierte Auswahl der beliebtesten Kryptowährungen sowie eine Suchfunktion, mit der Nutzern Kryptowährungen nach Ihrem Rang (Beliebtheit) such kann.

1.2 Projektbeschreibung

Meine Anwendung wird eine Kryptowährung-API nutzen, um mir sofort folgende Informationen bereitzustellen:

- **Bezeichnung**: Vollständige Namen der Kryptowährungen.
- **Rang (Beliebtheit)**: Aktueller Rang der Kryptowährungen.
- **Preis**: Echtzeit-Preisdaten für Kryptowährungen.

Ich setze Wert auf eine äusserst benutzerfreundliche Benutzeroberfläche, die einen mühelosen Zugriff auf Krypto-Infos ermöglicht. So bleibe ich immer auf dem Laufenden, ohne Zeit für aufwändige Recherchen aufwenden zu müssen.

2. Anforderungskatalog

2.1 Funktionale Anforderungen

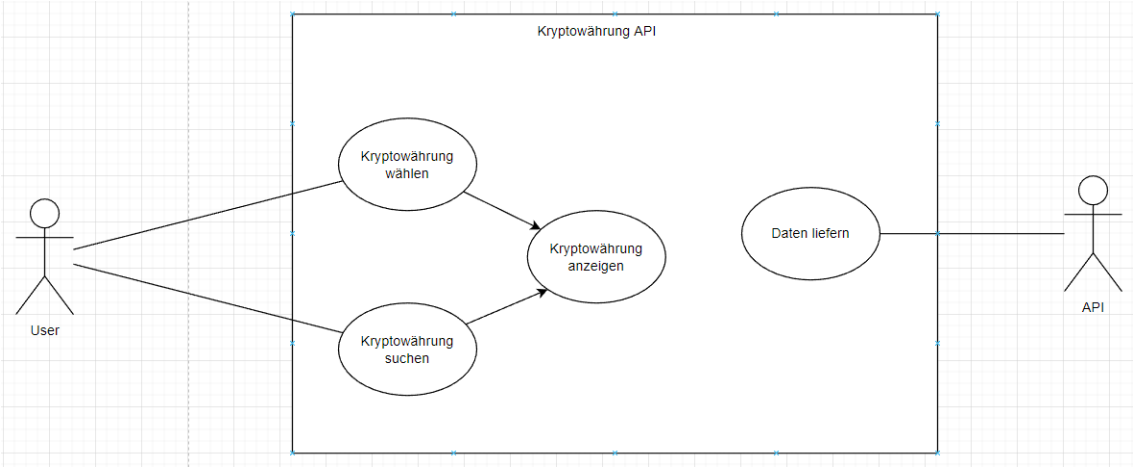

Abbildung 1 (Use-Case Diagramm)

- **Use Case**: Nutzer können mittels Vordefinierte Buttons eine Währung wählen und die API zeigt die Details der gewünschten Währung an. (Abbildung 1)
- **Use Case**: Nutzer können nach einer Währung (Rang der Währung) suchen und die API zeigt die Details der gewünschten Währung an. (Abbildung 1)

Die CoinLore API liefert bei einer http-Anfrage (GET-Methode) die gewünschten Währungen.

2.2 Technische Anforderungen

- **Browser**:	Firefox (Version 117.0.1 (64-Bit))
- **Programmier- / Markup-sprachen**:	JavaScript, HTML, CSS
- **API**:	CoinLore API (Keine Version vorhanden)
- **Laufzeitumgebung**:	Node.js (Version 18.17.1)
	JRE / JDK 17.0.9
- **Bibliothek**:	React inkl. Router
- **Betriebssystem**:	Windows 10 (Windows 10 Pro 64-Bit)
- **Entwicklungsumgebung / Editor**:	Visual Studio Code (Version 1.82.2)
	Eclipse IDE Enterprise (Version 4.29.0)
- **RDBM/ Datenbank**	MySQL (Version 8.0.34)
- **API Testing Tool**	Insomnia (Version 8.3.0)

Bemerkung: Um das Projekt richtig zu verwenden, ist eine Internetverbindung vorausgesetzt!

3. Datenbank ERD

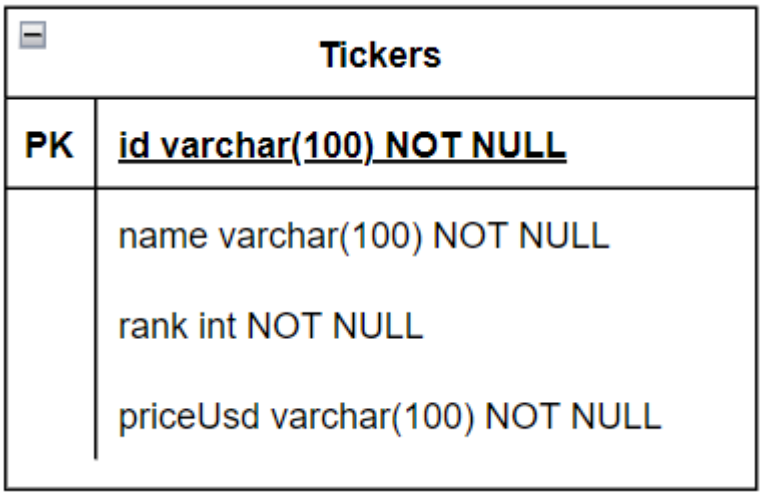

Abbildung 2 (ERD-Datenbank der coinsapi)

Da die Datenbank nur eine Tabelle beinhaltet, gibt es in diesem Fall keine Beziehungen und Kardinalitäten. Die Originale, vollständige API beinhaltet mehr Datensätze, jedoch trotzdem keine weiteren Tabellen.

4. Storyboard

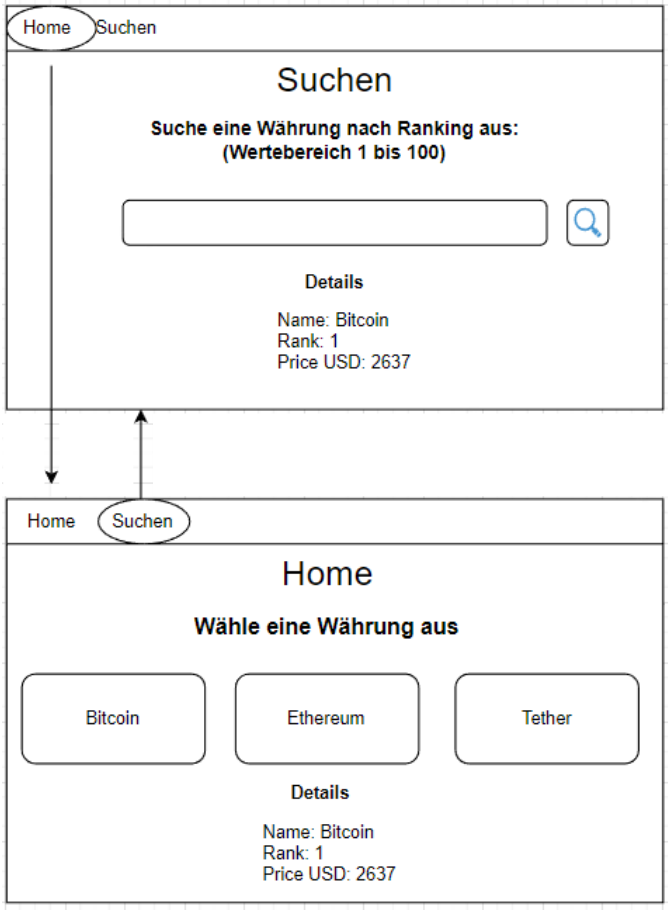

Abbildung 3 (Storyboard)

4.1 Home

- «Home» ***(das untere Bild von Abbildung 3)*** ist die Startseite. Das Betätigen des «Home» Buttons in der Navigation führt immer zu dieser Seite ***(das untere Bild von Abbildung 3)***.
- Besitzt 3 Vordefinierte Buttons:
    - Bitcoin
    - Ethereum
    - Tether
- Das Betätigen eines der Buttons zeigt die Details (Name der Währung, Rang, Preis in USD) der jeweiligen Währungen an.

4.2 Suche

- «Suche» ***(das obere Bild von der Abbildung 3)*** ist die Suchseite. Das Betätigen des «Suche» Buttons in der Navigation führt immer zu dieser Seite ***(das obere Bild von der Abbildung 3)***.
- Besitzt ein Suchfeld, in dem der Nutzer nach dem Rang (Ranking) der Währung suchen kann.
    - Der Wertebereich ist auf 1 bis 100 limitiert.
    - Einen Wertebereich kleiner oder grösser als 1 bis 100 gibt eine Fehlermeldung.
- Das Betätigen des «Lupe»-Buttons zeigt die Details (Name der Währung, Rang, Preis in USD) der nach dem Ranking gesuchten Währung an.

5. Screen-Mockups

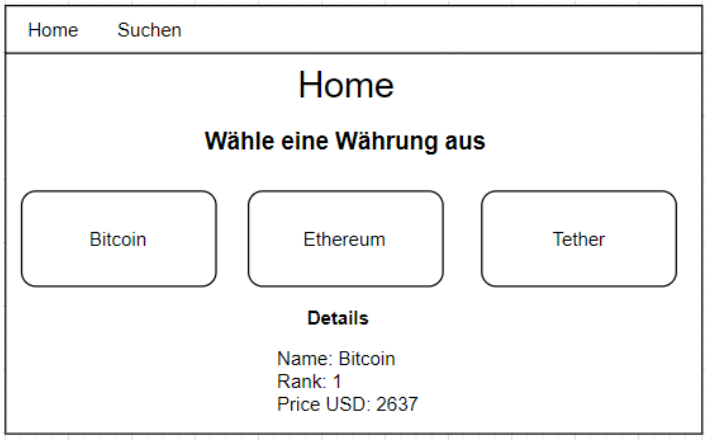

Abbildung 4 (Home-Mockup)

- Das Betätigen eines der Buttons (Bitcoin, Ethereum, Tether) zeigt die Details (Name der Währung, Rang, Preis in USD) der jeweiligen Währungen an *(Abbildung 4).*
- Das Betätigen des «Suche» Buttons in der Navigation führt zu «Abbildung 5».

Abbildung 5 (Suchen-Mockup)

- Das Betätigen des «Lupe»-Buttons zeigt die Details (Name der Währung, Rang, Preis in USD) der nach dem Rang gesuchten Währung an *(Abbildung 5).*
- Das Betätigen des «Home» Buttons in der Navigation führt zu «Abbildung 4»

6. REST-Schnittstellen

```
const response = await fetch(`http://localhost:8080/all`);
```

Abbildung 6 (GET-Methodenvariabel)

Die «`fetch()`»-Funktion ist eine JavaScript-Funktion, die zum Senden von HTTP-Anfragen (GET-Methode) an Server verwendet wird. Sie akzeptiert eine URL als Argument und sendet eine Anfrage an diese URL. Mittels «`await`» wird diese Operation asynchron ausgeführt und wartet zudem, bis die Anfrage abgeschlossen ist und eine Antwort zurückgegeben wurde.

```
const jsonData = await response.json();
```

Abbildung 7 (Response in JSON umwandeln)

«`response.json()`» ist eine Methode, die in JavaScript verwendet wird, um den Inhalt der HTTP-Antwort (die in der Variable «`response`» gespeichert ist) als JSON (JavaScript Object Notation) zu parsen und in ein JavaScript-Objekt umzuwandeln.

```
const filteredCurrency = jsonData.find(
  (currency) => currency.name === currencyName
);
```

Abbildung 8 (Response variable nach Währungsname suchen)

Die Zeile sucht das erste Element in dem Array «`jsonData`», dessen «`name`»-Attribut mit der «`name`»-Variable übereinstimmt, und speichert es in «`filteredCurrency`».

```
console.log("Filtered Response: " + Object.values(filteredCurrency));
```

Abbildung 9 (Ausgabe in Konsole für Debugging / Überprüfung)

Diese Zeile gibt die Werte der Eigenschaften des Objekts «`filteredCurrency`» in der Browser-Konsole aus, zusammen mit der Nachricht "Filtered Response:". Dies dient dazu, die Daten des gefundenen Elements in «`filteredCurrency`» in der Browser-Konsole anzuzeigen und zu überprüfen.

6.1 Datenstruktur

```
▼ 0:
    id:          "90"
    name:        "Bitcoin"
    rank:        1
    priceUsd:    "34100"
```

Abbildung 10 (API-Datenstruktur)

In meinem Projekt speichere ich die gesamte API-Antwort in die Variable «response» / «jsonData» und greife nur auf die Werte von "name", "rank" und "priceUsd" zu.
Die API verwendet für «name», sowie «priceUsd» Strings und der «rank» ist ein Integer.

Bemerkung: Die API-Datenstruktur wurde für dieses Projekt nur auf die benötigten Datensätze gekürzt und stimmt daher nicht mit der Datenstruktur aus der Projektdokumentation vom Modul 294.

6.2 Endpoints

```
@GetMapping("/all")
public ResponseEntity<List<Coin>> getAllCoins() {

    connect.connect();
    List<Coin> coins = connect.getAllCoins();
    connect.close();

    if (coins.isEmpty()) {
        return ResponseEntity.notFound().build();
    } else {
        return ResponseEntity.ok(coins);

    }
}
```

Abbildung 11 (/all Endpoint)

Dieser gibt beim GET-Request an «*https://localhost:8080/all*» alle Währungen von der Datenbank zurück, sortiert nach ***Rank.***

```
@GetMapping("/rank/{rank}")
public ResponseEntity<Coin> findCoinByRank(@PathVariable String rank) {

    connect.connect();
    Coin result = connect.findCoinByRank(rank);
    connect.close();

    if (result == null) {
        return ResponseEntity.notFound().build();
    } else {
        return ResponseEntity.ok(result);
    }
}
```

Abbildung 12 (/rank Endpoint)

Dieser gibt beim GET-Request an «*https://localhost:8080/rank/{rank}*» die nach dem Rang gesuchte Währung. Anstelle von *{rank}* setzt man den gesuchten *Rank,* von 1 bis und mit 100 ein.

```
@GetMapping("/name/{name}")
public ResponseEntity<Coin> findCoinByName(@PathVariable String name) {

    connect.connect();
    Coin result = connect.findCoinByName(name);
    connect.close();

    if (result == null) {
        return ResponseEntity.notFound().build();
    } else {
        return ResponseEntity.ok(result);
    }
}
```

Abbildung 13 (/name Endpoint)

Dieser gibt beim GET-Request an «*https://localhost:8080/name /{name}*» die nach dem Namen gesuchte Währung. Anstelle von *{name}* gibt man den Namen der gesuchten Währung an.

6.3 HTTP-Requests mit Insomnia

« Get all Coins »:
Da die Antwort dieser Abfrage zu gross ist, füge ich keine Abbildung dazu.

« Get Coin by Rank »:

Abbildung 14 (Get Coin by Rank Antwort)

Bei der HTTP-Anfrage wurde das Rank 9 mitgegeben / gesucht und gibt daher in der Response Body, die in der obigen Abbildung dargestellten Daten an.

« Get Coin by Name »:

Abbildung 15 (Get Coin by Name Antwort)

Bei der HTTP-Anfrage wurde der Name *Tether* mitgegeben / gesucht und gibt daher in der Response Body, die in der obigen Abbildung dargestellten Daten an.

6.4 Maven und Konfiguration

Dependencies:

```xml
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter</artifactId>
    </dependency>
</dependencies>
```

Abbildung 16 (pom.xml Dependencies)

Folgende Dependencies werden für dieses Projekt verwendet.

Um die Datenbankverbindung zu gewährleisten, ist die zweite Dependency «mysql-connector-java» notwendig.

Um die Applikation über Springboot laufen zu lassen, werden die Springboot Dependencies «org.springboot.boot» benötigt.

Plugins:

```xml
<plugins>
    <plugin>
        <artifactId>maven-clean-plugin</artifactId>
        <version>3.1.0</version>
    </plugin>
    <plugin>
        <artifactId>maven-resources-plugin</artifactId>
        <version>3.0.2</version>
    </plugin>
    <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.0</version>
    </plugin>
    <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.22.1</version>
    </plugin>
    <plugin>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.0.2</version>
    </plugin>
    <plugin>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
    </plugin>
    <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.2</version>
    </plugin>
    <plugin>
        <artifactId>maven-site-plugin</artifactId>
        <version>3.7.1</version>
    </plugin>
    <plugin>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>3.0.0</version>
    </plugin>
    <plugin>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
</plugins>
```

Abbildung 17 (pom.xml Plugins)

Folgende Plugins werden für diese Projekt verwendet.

Beim Erstellen des Maven Projekts, wurden diese Plugins automatisch hinzugefügt. Einige Plugins und Dependencies wurden beim Verwenden von Annotations wie *@SpringBootApplication* automatisch hinzugefügt.

6.5 Datenbankstruktur und Kommunikation

id	name	rank	price_usd
90	Bitcoin	1	34100
80	Ethereum	2	1790.78
518	Tether	3	1
2710	Binance Coin	4	227.15
33285	USD Coin	5	1
58	XRP	6	0.546218
46971	Staked Ether	7	1787.81
48543	Solana	8	32.22
257	Cardano	9	0.293839
2	Dogecoin	10	0.069219
2713	TRON	11	0.094082
77729	The Open Network	12	2.05
2751	ChainLink	13	11.49
33422	Wrapped Bitcoin	14	34070.93
45219	Polkadot	15	4.19
1	Litecoin	16	67.93
2321	Bitcoin Cash	17	245.41
45088	Shiba Inu	18	0.000008
48591	Binance USD	19	1
44883	Avalanche	20	10.92
33833	UNUS SED LEO	21	3.97
89	Stellar	22	0.11359
28	Monero	23	161.85
32479	TrueUSD	24	1
33531	OKB	25	45.18
33830	Cosmos	26	7.17
47305	Uniswap	27	4.12
118	Ethereum Classic	28	16.3
46981	Lido DAO	29	1.83
32607	Filecoin	30	3.67
12377	Maker	31	1444.51
2741	VeChain	32	0.019171
33536	Matic Network	33	0.621894
33085	Quant	34	107.81
93847	Arbitrum	35	0.929604
46018	Aave	36	80.48
48563	NEAR Protocol	37	1.24
46183	Injective Protocol	38	13.31
70485	Kaspa	39	0.050909
33234	Bitcoin SV	40	49.25
70497	Optimism	41	1.38
46968	Frax	42	1
48569	Stacks	43	0.658192
48561	The Graph	44	0.100877
44863	Render Token	45	2.36
36447	THORChain	46	2.54
34406	Algorand	47	0.101616
69801	USDD	48	0.995407
2679	EOS	49	0.631102
3682	Tezos	50	0.727205
32360	Theta Token	51	0.671469
258	Decentraland	52	0.349405
133	Neo	53	9.31
33644	Fantom	54	0.234502
67117	Bitget Token	55	0.46107
45161	The Sandbox	56	0.329974
46575	Axie Infinity	57	4.88
32893	Red Pulse Phoenix	58	0.724362
62645	Mina Protocol	59	0.622865
33819	Crypto.com Chain	60	0.059277
48589	Flow	61	0.530518
64671	Pax Dollar	62	1
51539	eCash	63	0.000028
45930	APEcoin	64	1.37
42227	PAX Gold	65	1990.57
48555	Hedera Hashgraph	66	0.05196
42855	Tether Gold	67	2004.03
93841	Pepe	68	0.000001
33213	Rocket Pool	69	24.02
34391	Chiliz	70	0.06516
44035	Tokenize Xchange	71	5.77
134	Zcash	72	27.84
45577	Gala	73	0.018178
44178	Trust Wallet Token	74	1.03
447	IOTA	75	0.154752
64675	Kava	76	0.638757
48547	FTX Token	77	1.27
42531	Klaytn	78	0.131883
48581	Curve DAO Token	79	0.475593
77735	GMX	80	46.17
45467	Elrond eGold	81	29.18
32351	Huobi Token	82	2.34
48537	Terra Classic	83	0.000065
48703	Casper	84	0.033084
54377	dYdX	85	2.37
47304	Compound	86	45.6
51947	Floki Inu	87	0.000037
32604	Nexo	88	0.613668
46642	Wemix Token	89	1.07
100423	First Digital USD	90	1
8	Dash	91	28.12
42441	Arweave	92	4.79
237	Qtum	93	2.97
93845	Sui	94	0.459898
184	Basic Attention Token	95	0.201279
48711	Liquity USD	96	0.99531
32334	Zilliqa	97	0.018614
46966	1inch	98	0.289648
33718	Fetch	99	0.348855
32354	SingularityNET	100	0.230202

Abbildung 18 (MySQL Datenbank)

Von der Ursprünglichen [API CoinLore], benötige ich nur die in der Abbildung dargestellten und im [Unterkapitel 6.1] erwähnten Daten.

Da die Intention dieses Projekts, die beliebtesten Währungen anzuzeigen, werden die Daten in der Datenbank nach dem *rank* sortiert.

7. Testplan (Frontend)

ID / Bezeichnung	T-001	Buttons Rendering
Beschreibung	Überprüfe, ob die Buttons auf Startseite korrekt gerendert werden.	
Testschritte	1. Starte die Anwendung 2. Klicke auf den «Home» Link in der Navigationsleiste	
Erwartetes Ergebnis	Die Startseite soll angezeigt werden und enthält die Buttons mit den vordefinierten Buttons der Währungen.	
Effektives Ergebnis	Die Startseite wird angezeigt und enthält die Buttons mit den vordefinierten Buttons der Währungen.	
Massnahme	Keine	

Tabelle 1 (Frontend Test T-001)

ID / Bezeichnung	T-002	Navigation zu «Suchen» Page
Beschreibung	Überprüfe, ob die Navigation zu «Suchen» Seite funktioniert.	
Testschritte	1. Starte die Anwendung 2. Klicke auf den «Suchen» Link in der Navigationsleiste	
Erwartetes Ergebnis	Die «Suchen» Seite soll angezeigt werden.	
Effektives Ergebnis	Die «Suchen» Seite wird angezeigt.	
Massnahme	Keine	

Tabelle 2 (Frontend Test T-002)

ID / Bezeichnung	T-003	Navigation zu «Home» Page
Beschreibung	Überprüfe, ob die Navigation zu «Home» Seite funktioniert.	
Testschritte	1. Starte die Anwendung 2. Klicke auf den «Suchen» Link in der Navigationsleiste 3. Klicke auf den «Home» Link in der Navigationsleiste	
Erwartetes Ergebnis	Die «Home» Seite soll angezeigt werden.	
Effektives Ergebnis	Die «Home» Seite wird angezeigt.	
Massnahme	Keine	

Tabelle 3 (Frontend Test T-003)

ID / Bezeichnung	T-004	Suche nach einer Währung mit einem gültigen Rang
Beschreibung	Überprüfe, ob die Suche nach einer Währung mit einem gültigen Rang korrekt funktioniert und die Details der Währung angezeigt werden.	
Testschritte	1. Starte die Anwendung 2. Klicke auf den «Suchen» Link in der Navigationsleiste 3. Gib ein gültiger Rang in das Suchfeld ein und klicke auf die Lupe	
Erwartetes Ergebnis	Die Details der Währung mit dem entsprechenden Rang soll angezeigt werden.	
Effektives Ergebnis	Die Details der Währung mit dem entsprechenden Rang werden angezeigt.	
Massnahme	Keine	

Tabelle 4 (Frontend Test T-004)

ID / Bezeichnung	T-005	Suche nach einer Währung mit einem ungültigen Rang
Beschreibung	Überprüfe, ob eine Suche nach einer ungültigen Währung (z.B. nicht vorhandenes Rang) angemessen behandelt wird.	
Testschritte	1. Starte die Anwendung 2. Klicke auf den «Suchen» Link in der Navigationsleiste 3. Gib ein ungültiger Rang in das Suchfeld ein und klicke auf die Lupe	
Erwartetes Ergebnis	Es soll eine angemessene Fehlermeldung angezeigt werden, dass die Eingabe der Währung im Wertebereich sein muss.	
Effektives Ergebnis	Es wird eine angemessene Fehlermeldung angezeigt, dass die Eingabe der Währung im Wertebereich sein muss.	
Massnahme	Keine	

Tabelle 5 (Frontend Test T-005)

8. Testplan (Backend) mit Insomnia

ID / Bezeichnung	T-001	Wiedergabe von allen Währungen
Beschreibung	Überprüfe, ob alle Währungen vom Backend zurückgegeben werden.	
Testschritte	1. Starte die Anwendung Insomnia 2. Erstelle eine HTTP-Request and die localhost:8080/all URL 3. Überprüfe, ob alle Währungen korrekt zurückgegeben werden.	
Erwartetes Ergebnis	Das Backend soll alle Währungen korrekt zurückgeben.	
Effektives Ergebnis	Das Backend gibt alle Währungen korrekt zurück.	
Massnahme	Keine	

Tabelle 6 (Backend Test T-001)

ID / Bezeichnung	T-002	Wiedergabe eines Coins mittels Namens suche
Beschreibung	Überprüfe, ob die Währung durch Namenssuche vom Backend zurückgegeben wird.	
Testschritte	1. Starte die Anwendung Insomnia 2. Erstelle eine HTTP-Request and die localhost:8080/name/{name} URL 3. Füge anstelle des {name}, den Namen eines Coins ein (Bsp. «Bitcoin»)	
Erwartetes Ergebnis	Das Backend soll die gesuchte Währung zurückgeben.	
Effektives Ergebnis	Das Backend gibt die gesuchte Währung zurück.	
Massnahme	Keine	

Tabelle 7 (Backend Test T-002)

ID / Bezeichnung	T-003	Wiedergabe eines Coins mittels Ranges suche
Beschreibung	Überprüfe, ob die Währung durch Rang suche vom Backend zurückgegeben wird.	
Testschritte	1. Starte die Anwendung Insomnia 2. Erstelle eine HTTP-Request and die localhost:8080/rank /{rank} URL 3. Füge anstelle des {rank}, den Namen eines Coins ein (Bsp. 1)	
Erwartetes Ergebnis	Das Backend soll die gesuchte Währung zurückgeben.	
Effektives Ergebnis	Das Backend gibt die gesuchte Währung zurück.	
Massnahme	Keine	

Tabelle 8 (Backend Test T-003)

ID / Bezeichnung	T-004	Wiedergabe der Index Seite
Beschreibung	Überprüfe, ob die Index Seite mit der Begrüssung «Hello, this is my LBM_295» erscheint	
Testschritte	1. Starte die Anwendung Insomnia 2. Erstelle eine http-Request an die localhost:8080/ URL	
Erwartetes Ergebnis	Die Index Seite soll mit dem Text «Hello, this is my LB_M295» wird angezeigt werden.	
Effektives Ergebnis	Die Index Seite wird mit dem Text «Hello, this is my LB_M295» wird angezeigt.	
Massnahme	Keine	

Tabelle 9 (Backend Test T-004)

ID / Bezeichnung	T-005	Backend Projekt starten
Beschreibung	Überprüfe, dass beim Starten des Backends keine Fehlermeldungen erscheinen oder nicht abstürzt.	
Testschritte	1. Starte die Anwendung Eclipse IDE 2. Klicke auf das grüne Play-Button, um das Backend Projekt zu starten 3. Beobachte, ob in der Konsole keine Fehlermeldungen erscheinen.	
Erwartetes Ergebnis	Das Backend Projekt soll ohne Fehlermeldungen oder Abstürze gestartet werden.	
Effektives Ergebnis	Das Backend Projekt wird ohne Fehlermeldungen oder Abstürze gestartet.	
Massnahme	Keine	

Tabelle 10 (Backend Test T-005)

9. Installationsanleitung (Datenbank)

9.1 MySQL installieren

- Falls MySQL noch nicht installiert ist, kannst du dies von der Offiziellen Webseite herunterladen [siehe Kapitel 13 – Hilfestellungen und Quellen]
- Führe den Installer für MySQL aus und folge dem Installationsprozess.

9.2 Projekt herunterladen und entpacken

- Lade das ZIP-Archiv «**LB-Projekt-M294-295_Jason-Termine.zip**» herunter.
- Entpacke das Archiv in einen Ordner deiner Wahl auf deinem Computer.

9.3 Import Berechtigung anpassen

Windows 10 / 11:

- Öffne die Befehlseingabe (CMD).
- Navigiere zum Ordner zur Konfigurationsdatei von MySQL mit folgendem Befehl:
    - CD C:\ProgramData\MySQL\MySQL Server 8.0\
- Öffne die Datei «my.ini» und suche mittels der Suchfunktion [CTRL + F] «*secure-file-priv*».
- Ersetze den Pfad, welches im «secure-file-priv» steht, mit Leeren Anführungszeichen:
    - secure-file-priv=""
- Speichere die Datei ab.

Ubuntu 22.04 & 23.04

- Öffne den Terminal / die Konsole und wechsle in den Root User.
- Navigiere zum Ordner zur Konfigurationsdatei von MySQL mit folgendem Befehl:
    - CD /etc/mysql/
- Öffne die Datei «my.cnf» und suche mittels integrierter Suchfunktion nach «[mysqld]»
- Füge unterhalb von «[mysqld]» diesen Befehl:
    - secure-file-priv=""
- Falls «[mysqld]» nicht vorhanden ist, muss man dies nachträglich hinzufügen.

Hinweis: Diese Konfiguration ist **notwendig**, da MySQL ansonsten einen Error ausgibt, dass man nicht dazu berechtigt ist, Dateien zu importieren.

9.4 Datenbank importieren

- **In Windows**: Öffne den «MySQL 8.0 Command Line Client» und melde dich an.
- **In Linux***:* Öffne den Terminal und gib folgender Befehl ein, um dich in MySQL anzumelden:
    - mysql -u root -p
- Melde dich mit deinem Passwort in MySQL an.
- Die Import-Datei «coinsapi.sql» für die Datenbank befindet sich im Projektordner unter «**Ressourcen**».
- Mit folgendem Befehl kannst du die vorbereitete «*coinsapi*» Datenbank importieren:
    - SOURCE *dein/Pfad/zu/dieser/Datei/**coinsapi.sql***
- Wenn die Datenbank erfolgreich importiert wurde, solltest du folgende Meldung erhalten:

```
Query OK, 100 rows affected (0.00 sec)
Records: 100  Duplicates: 0  Warnings: 0
```

Abbildung 19 (MySQL Erfolgreiche Meldung)

10. Installationsanleitung (Backend)

10.1 JDK 17 / Java installieren

- Falls Java und JDK 17 noch nicht installiert sind, kannst du diese von den jeweiligen Offizielle Webseiten herunterladen [siehe Kapitel 13 – Hilfestellungen und Quellen]
- Führe den Installer aus und folge dem Installationsprozess.

10.2 Projekt herunterladen und entpacken

- Lade das ZIP-Archiv «**LB-Projekt-M294-295_Jason-Termine.zip**» herunter.
- Entpacke das Archiv in einen Ordner deiner Wahl auf deinem Computer.

10.3 Eclipse IDE öffnen

- Falls du Eclipse IDE noch nicht installiert hast, kannst du von der Offiziellen Website [siehe Kapitel 13 – Hilfestellungen und Quellen] den Installer herunterladen und im Installer dann die «Eclipse IDE for Enterprise» wählen und den Installationsprozess folgen.

10.4 Projektordner in Eclipse IDE importieren

- Starte Eclipse IDE.
- Klicke auf "**Datei**" (File) in der oberen linken Ecke von Eclipse.
- Wähle "**Import**" aus und navigiere dann zum «**Maven**» Ordner und wähle «**Existing Maven Projects**». Klicke dann auf «**Browse**» / «**Durchsuchen**» und navigiere zu dem entpackten Archiv und wähle anschliessend den Projektordner («Back-End») aus.
- Klicke anschliessend auf «**Finish**» / «**Fertig**».
- Öffne danach die Klasse «**App.js**» und füge in den Feldern «**[BENUTZERNAME]**» und «**[PASSWORT]**» die Anmeldedaten deiner Datenbank ein:

```
18    public final Connect connect = new Connect("[BENUTZERNAME]", "[PASSWORT]");
```

Abbildung 20 (Anmeldedaten der Datenbank festlegen)

- Öffne dann die «**src/main/resources/application.properties**» Datei in und füge deine Datenbank-Anmeldedaten ebenfalls in «**[BENUTZERNAME]**» und «**[PASSWORT]**» ein:

```
1 spring.jpa.hibernate.ddl-auto=update
2 spring.datasource.url=jdbc:mysql://localhost:3306/coinsapi
3 spring.datasource.username=[BENUTZERNAME]
4 spring.datasource.password=[PASSWORT]
5 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

Abbildung 21 (Eigenschaften der Spring und JDBC-Konfiguration)

10.5 Backend starten

- **Voraussetzung [Kapitel 9 – Installationsanleitung Datenbank]:** Der MySQL Dienst muss gestartet sein, da das Backend eine Verbindung zu der Datenbank benötigt.
- Navigiere zum App.java und klicke in der Oberen Menu-leiste das grüne Play-Button:

Abbildung 22 (Play-Button im Eclipse IDE)

10.6 Sicherstellen

- Überprüfe, dass in der Konsole **keine** Fehlermeldungen erscheinen. Um die Verbindung zu testen, kannst du in deinem Browser auf localhost:8080/all gehen und sicherstellen, dass alle Einträge von der Datenbank abgerufen werden.

11. Installationsanleitung (Frontend)

11.1 Node.js installieren

Falls Node.js noch nicht installiert ist, kannst du von der Offiziellen Node.js Webseite [siehe Kapitel 13 – Hilfestellungen und Quellen] die neuste LTS-Version [siehe Unterkapitel 2.2] herunterladen und installieren.

11.2 Projekt herunterladen und entpacken

- Lade das ZIP-Archiv «**LB-Projekt-M294-295_Jason-Termine.zip**» herunter.
- Entpacke das Archiv in einen Ordner deiner Wahl auf deinem Computer.

11.3 Visual Studio Code öffnen

- Falls du Visual Studio Code noch nicht installiert hast, kannst du von der Offiziellen VS-Code-Website die «Stable Build» Version [siehe Unterkapitel 2.2] herunterladen und installieren.

11.4 Projektordner öffnen

- Starte Visual Studio Code.
- Klicke auf "**Datei**" (File) in der oberen linken Ecke von VS-Code.
- Wähle "**Ordner öffnen**" (Open Folder) aus und navigiere zum Ordner, in dem das Projekt entpackt wurde.
- Navigiere zum Projektordner («Front-End») und klicke auf "**Ordner auswählen**" (Select Folder).

11.5 Terminal öffnen

- Öffne in Visual Studio Code das integrierte Terminal, indem du auf "**Terminal**" in der oberen Menüleiste und dann auf "**Neues Terminal**" (New Terminal) klickst.

11.6 Projekt starten

- **Wichtig**: Führe im Terminal den folgenden Befehl aus: **`npm install`**
- Um das Projekt zu starten, kannst du anschliessend folgenden Befehl im Terminal ausführen: **`npm start`**

11.7 Unit Tests

- Um die Unit Tests auszuführen, stelle sicher, dass das Projekt **nicht** gestartet ist.
- Anschliessend führe folgenden Befehl im Terminal aus, um die Tests zu starten: **`npm test`**
- Die Tests gelten als bestanden, wenn die folgende Meldung in der Konsole angezeigt wird:

```
PASS  src/__tests__/Home.test.js
PASS  src/__tests__/Suchen.test.js
PASS  src/__tests__/GlobalNavigation.test.js

Test Suites: 3 passed, 3 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        2.325 s
Ran all test suites.
```

Abbildung 23 (Jest-Unit Tests)

12 Installationsanleitung (Insomnia)

Voraussetzung:
[Kapitel 9 – Installationsanleitung Datenbank] und [Kapitel 10 - Installationsanleitung Backend]

12.1 Insomnia installieren

- Falls Insomnia noch nicht installiert ist, kannst du dies von der Offiziellen Webseite herunterladen. [siehe Kapitel 13 – Hilfestellungen und Quellen]

12.2 Projekt herunterladen und entpacken

- Lade das ZIP-Archiv «**LB-Projekt-M294-295_Jason-Termine.zip**» herunter.
- Entpacke das Archiv in einen Ordner deiner Wahl auf deinem Computer.

12.3 Insomnia öffnen und die Konfigurationsdatei importieren

- Öffne Insomnia und navigiere zum «Filter» Input-Feld und klicke auf das Plus-Zeichen, rechts neben dem Input-Feld:

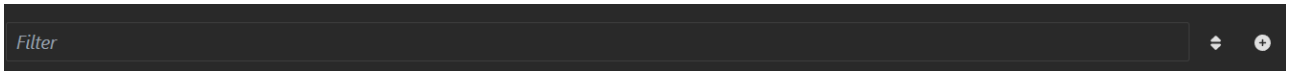

Abbildung 24 (Insomnia Konfigurationsdatei importieren)

- Klicke anschliessend auf «Import» und navigiere zum entpackten Projektordner. Im Unterordner «Ressourcen», befindet sich die JSON-Datei zum Importieren: «**Insomnia_Config_LB_M295_Jason_Termine.json**»
- Folge dem Importprozess und klicke anschliessend auf das Neu importierte Dokument im Insomnia:

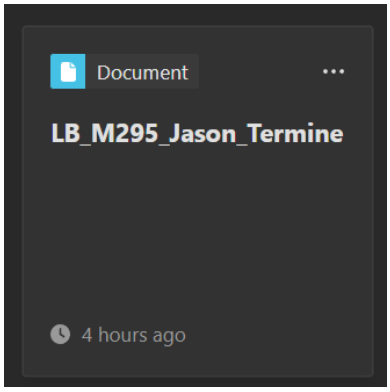

Abbildung 25 (Insomnia Konfigurationsdatei öffnen)

- Anschliessend kann man die vorbereiteten HTTP-GET-Requests:
  - «Get All Coins»
  - «Get Coin by Rank»
  - «Get Coin by Name»

  betätigen und die verschiedenen Endpoints testen, indem man recht neben der URL-Eingabe auf «**Senden**» klickt.

13 Hilfestellungen und Quellen

13.1 Künstliche Intelligenz

- https://chat.openai.com/
  - Teile vom Code von folgenden Dateien im Projekt wurden mithilfe von Chat-GPT generiert:
  - App.js
  - Suchen.js
  - Suchen.test.js
  - GlobalNavigation.test.js

Bemerkung: Die von ChatGPT generierten Zeilen sind mit folgenden Kommentaren gekennzeichnet: /********** ChatGPT ab dieser Linie bis und mit [betroffene Linie] *********/

13.2 Tools und Programme

- https://www.mozilla.org/de/firefox/new/ - Firefox Browser
- https://code.visualstudio.com/download - Visual Studio Code
- https://www.drawio.com/ - Draw.io
- https://nodejs.org/ - Zur Installation von Node.js
- https://www.oracle.com/ch-de/java/technologies/downloads/ - JRE & JDK 17
- https://www.eclipse.org/downloads/packages/installer - Eclipse IDE Installer
- https://dev.mysql.com/downloads/mysql/ - MySQL 8.0.34
- https://insomnia.rest/download - Insomnia 8.3.0

13.3 API

- https://api.coinlore.net/api/tickers/ - CoinLore API

13.4 Dokumentation

- https://react.dev/learn - React Dokumentation
- https://www.w3schools.com/js/ - JavaScript Dokumentation

14 Abbildungsverzeichnis

Abbildung 1 (Use-Case Diagramm) 5
Abbildung 2 (ERD-Datenbank der coinsapi) 6
Abbildung 3 (Storyboard) 7
Abbildung 4 (Home-Mockup) 8
Abbildung 5 (Suchen-Mockup) 8
Abbildung 6 (GET-Methodenvariabel) 9
Abbildung 7 (Response in JSON umwandeln) 9
Abbildung 8 (Response variable nach Währungsname suchen) 9
Abbildung 9 (Ausgabe in Konsole für Debugging / Überprüfung) 9
Abbildung 10 (API-Datenstruktur) 9
Abbildung 11 (/all Endpoint) 10
Abbildung 12 (/rank Endpoint) 10
Abbildung 13 (/name Endpoint) 10
Abbildung 14 (Get Coin by Rank Antwort) 11
Abbildung 15 (Get Coin by Name Antwort) 11
Abbildung 16 (pom.xml Dependencies) 12
Abbildung 17 (pom.xml Plugins) 12
Abbildung 18 (MySQL Datenbank) 13
Abbildung 19 (MySQL Erfolgreiche Meldung) 18
Abbildung 20 (Anmeldedaten der Datenbank festlegen) 19
Abbildung 21 (Eigenschaften der Spring und JDBC-Konfiguration) 19
Abbildung 22 (Play-Button im Eclipse IDE) 19
Abbildung 23 (Jest-Unit Tests) 20
Abbildung 24 (Insomnia Konfigurationsdatei importieren) 21
Abbildung 25 (Insomnia Konfigurationsdatei öffnen) 21

15 Tabellenverzeichnis

Tabelle 1 (Frontend Test T-001) 14
Tabelle 2 (Frontend Test T-002) 14
Tabelle 3 (Frontend Test T-003) 14
Tabelle 4 (Frontend Test T-004) 15
Tabelle 5 (Frontend Test T-005) 15
Tabelle 6 (Backend Test T-001) 16
Tabelle 7 (Backend Test T-002) 16
Tabelle 8 (Backend Test T-003) 16
Tabelle 9 (Backend Test T-004) 17
Tabelle 10 (Backend Test T-005) 17