Abiturjahrgang 2023

Dokumentation

Fach: Informatik

Thema der Arbeit:

Smart Prediction Tool

Verfasser: Bernd Storath und Joshua Pfennig

Kursleiter: StRin Nora Bender

Abgabetermin: 01. Juli 2022

_____ _____

Unterschrift Kursleiter eingegangen am

1. Einführung

Die Börse ist ein ständiges Auf und Ab. Demnach ist ein Tool, welches die neuesten Nachrichten nutzt, um sowohl aktuellen Aktienbesitzern die Entscheidung zu erleichtern, ob sie ihre Aktie verkaufen oder behalten sollten, als auch möglichen zukünftigen Anteilseignern die Wahl zu vereinfachen, ob sie ein Unternehmen mit finanziellen Mitteln unterstützen sollten, sehr sinnvoll.

2. Zielsetzung sowie grober Aufbau des Projekts

Dem Nutzer wird mithilfe einer UI die Möglichkeit geboten, eine Aktie seiner Wahl auszusuchen, deren bisherige Historie angezeigt und zukünftige Entwicklung vorhergesagt werden. Hierbei soll der Aktienpreis mithilfe des Crawlens von Nachrichtenseiten durch das Einlesen dieser News geschätzt werden, indem die Websites mithilfe der Methode der „Sentiment Detection“ komplett analysiert wird. Das Resultat dieser Untersuchung gibt dementsprechend an, wie der Aktienverlauf, zumindest in naher Zukunft, zu erwarten ist.

Nachfolgend sollen die einzelnen oben genannten Schritte noch einmal ausführlicher beschrieben werden.

Die Benutzeroberfläche greift auf die Finnhub API zu, welche, beispielsweise mittels einer json-Datei, die gewünschte Aktie inklusive ihrer bis zu sechsjähriger Historie zur Verfügung stellt. Zudem wird die UI mit Next.JS basierend auf Javascript entwickelt und sorgt so für die Website.

Für das Erhalten des Sentiment Scores sowie der aktuellen Nachrichten wird auf die MarketAux API zugegriffen. Hierbei wird mithilfe der Methode der „Sentiment Detection“ geprüft, ob die Artikel eher positiv oder negativ über die Firmen beziehungsweise deren Aktien berichten. Durch den dadurch herausgefundenen Wert kann zwar nicht vorhergesagt werden, um wieviel Prozent sich die Aktie genau verändern wird, jedoch wird zumindest eine grobe Hypothese aufgestellt, ob diese steigt, fällt oder stabil bleiben wird.

3. Erfolge

a) Auswahl der gewünschten Aktie

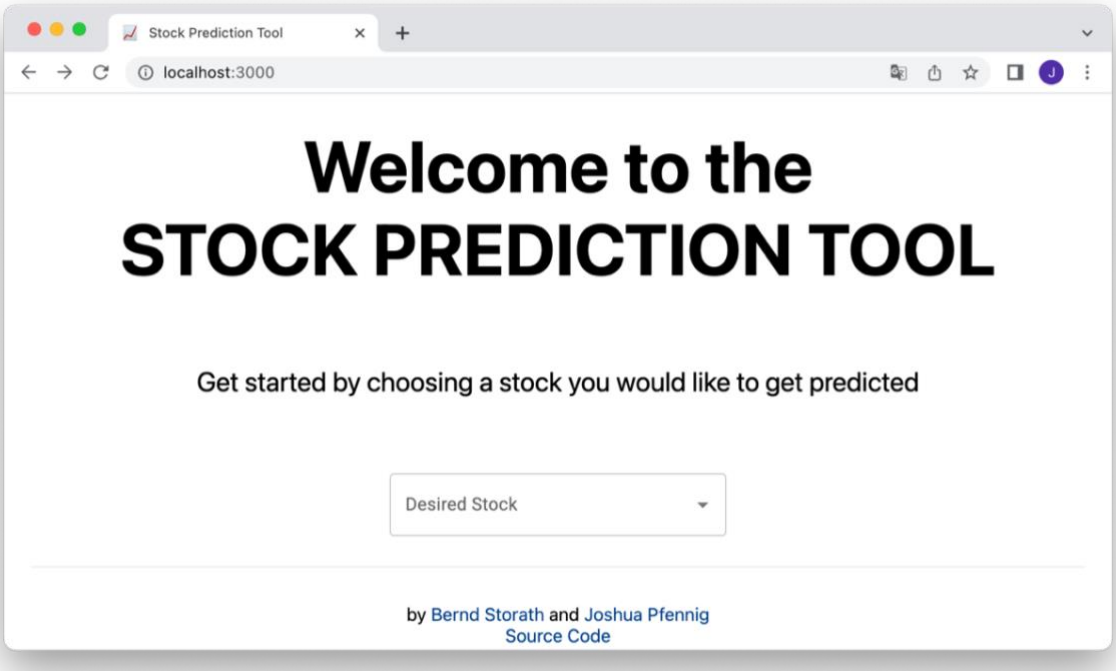

Abbildung 2: Homepage zur Auswahl der gewünschten Aktie

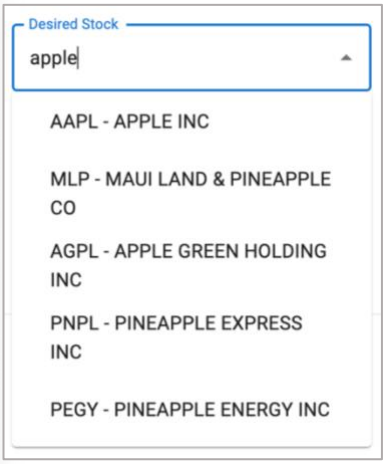

Abbildung 1: Integriertes Dropdown mit Zugriff auf Finnhub-API

Mithilfe eines Dropdown-Menüs (vgl. Abbildung 1) wird sichergestellt, dass der richtige Aktienname vom Nutzer gewählt wird, da auf äquivalente Eingaben reagiert wird. Hierbei wird auf die Finnhub-API zugegriffen. Um die Geschwindigkeit der Anzeige aller gefundenen Aktien zu beschleunigen, wurde die Liste virtualisiert. Hierdurch werden die DOM-Elemente erst geladen, wenn der Nutzer die Positionen dieser im Dropdown erreicht hat.

Zuletzt sei noch zu erwähnen, dass eine Verlinkung sowohl zu dem persönlichen Github-Profil der beiden Team-Mitglieder als auch zum Quellcode unter dem angelegten Git-Repository angegeben ist (vgl. Abbildung 2).

b) Anzeige des bisherigen Aktienverlaufs sowie der Sentiment Analysis

Abbildung 3: Weiterleitung an Unterseite für die gewünschten Informationen

Bestätigt man die Eingabe der Aktie mittels der Enter-Taste, so wird man automatisch an eine ebenfalls selbst designte und entwickelte Seite weitergeleitet, die die Unterdomain mit der Bezeichnung der gewünschten Aktie enthält (siehe Markierung).
Unter dem ebenfalls von der Finnhub-API erhaltenen Logo sowie dem eingetragenen Unternehmensnamennamen verbirgt sich außerdem der Link zur offiziellen Website der jeweiligen Firmen.
Darunter ist das „Herzstück“ des gesamten Projekts zu finden:
Zum einen ist der bisherige Aktienverlauf abgebildet, wobei hier eine dynamische Interaktion möglich ist, um jeden einzelnen Tag genauer zu betrachten (vgl. Abbildung 3).
Außerdem liegt eine Datenbank zugrunde, die die täglich gemessenen Sentiment Scores anzeigt. Ist jedoch, aufgrund der beschränkten Dauer dieses Projekts, kein Datenbankeintrag vorhanden, so wird dies ebenfalls angezeigt.
Neben der Anzeige des Datenbankverlaufs ist zudem der aktuelle Sentiment Score grafisch vorzufinden.
Schließlich werden darunter drei Nachrichten angezeigt, zusammen mit den individuellen Sentiment Scores für die jeweilige Nachricht.

4. Starten der Webapplikation

a) Starten des Backends

```
(base) joshuapfennig@MacBook-Pro-von-Joshua stocks-prediction-tool % cd backend
(base) joshuapfennig@MacBook-Pro-von-Joshua backend % make start
flask run
 * Environment: development
 * Debug mode: on
 * Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 121-489-781
```

b) Starten des Frontends

```
(base) joshuapfennig@MacBook-Pro-von-Joshua stocks-prediction-tool % cd frontend
(base) joshuapfennig@MacBook-Pro-von-Joshua frontend % yarn dev
yarn run v1.22.18
$ next dev
ready - started server on 0.0.0.0:3000, url: http://localhost:3000
wait  - compiling...
```

Die markierte Webadresse ist in einen Browser Ihrer Wahl einzugeben. Durch das Hosten des Projekts auf Bernds Server über einen Docker Container sind keine weiteren Installationen mehr durchzuführen.

5. Auszüge aus dem Programmiercode

a) Backend

a. Erhalten der Daten aus der Marketaux API

```python
def getNews(symbol) -> News:
    """
    Get News for Symbol

    :param symbol: Symbol
    """
    conn = http.client.HTTPSConnection('api.marketaux.com')

    params = urllib.parse.urlencode({
        'api_token': os.environ['MARKETAUX_KEY'],
        'symbols': symbol,
        'limit': 50,
        'filter_entities': True,
        'language': 'en'
        })

    conn.request('GET', '/v1/news/all?{}'.format(params))

    res = conn.getresponse()
    data = res.read()

    data = data.decode('utf-8')

    dataAsDictionary = json.loads(data)

    dataAsObject = Box(dataAsDictionary)

    return dataAsObject
```

Zuerst wird für eine Verbindung zur API aufgebaut. Dabei werden Parameter übergeben, welche die Einstellungen zu den erhaltenen Daten sowie den API Key, der einzigartig ist und erst den Zugriff ermöglicht, enthalten.
Hierbei werden die Daten vorerst im json-Format gespeichert, um sie schließlich in ein Objekt umzuwandeln.

b. API-Endpunkte

```python
@app.route('/bars')
def bars():
    symbol = request.args.get('symbol')

    if symbol is None:
        return {"error": "No symbol provided"}, 400

    timeframe = request.args.get('timeframe', 'D')
    fromTimestamp = request.args.get('from', "1609459200")
    toTimestamp = request.args.get('to', "1640908800")
    res = client.stock_candles(symbol, timeframe, fromTimestamp, toTimestamp)

    if (res['s'] == "no_data"):
        return {"error": "No data available"}, 404

    df = pd.DataFrame(res).to_json(orient="records")
    return df
```

Je nach gewünschter Funktion ruft das Frontend eine gewisse Funktion des Backends auf, wobei dies über eine URL-Route stattfindet, die das Backend einließt. Dabei gibt diese Methode die Datenpunkte der Historie der gewünschten Aktie zurück oder wirft einen Fehler, wenn das Unternehmen nicht gefunden wurde.

Analog zu der gezeigten Methode verhält es sich auch bezüglich der Routen „sentiment“, „symbols“, „company“ und „sentiments“.

b) Frontend

a. Methoden zum Erhalt der Daten aus dem Backend

```
export async function getSentiment(stock: string): Promise<Sentiment> {
  const request = await fetch(`${API_URL}/sentiment?symbol=${stock}`);
  if (!request.ok) {
    throw new Error(request.statusText);
  }

  const averageSentiment = await request.json();
  return averageSentiment;
}
```

Analog zu dieser Methode für den Sentiment Score verhalten sich auch die Methoden zum Erhalt des Logos, der Nachrichten sowie der Informationen zum gewünschten Unternehmen, wobei eigene Interfaces als Datentypen implementiert wurden, wie am folgenden Beispiel ersichtlich wird.

```
export interface Sentiment {
  averageSentiment: number | null;
  data: SentimentData[];
}
```

b. Anzeige des Graphen

```
export default function Chart({ data }: ChartProps) {
  return (
    <ResponsiveContainer width="100%" height={300}>
      <LineChart data={data}>
        <Line type="monotone" dataKey="average" stroke="#8884d8" name="Price" />
        <CartesianGrid stroke="#ccc" strokeDasharray="5 5" />
        <XAxis
          dataKey="date"
          tickFormatter={(v: Date) => v.toLocaleDateString('de-DE', { month: 'long' })}
        />
        <YAxis tickFormatter={(v: number) => `$${v}`} />
        <Tooltip content={(props) => <ExtendedTooltip props={props} />} />
      </LineChart>
    </ResponsiveContainer>
  );
}
```

Zuerst wird die Größe des Graphen spezifiziert. Hierbei nimmt die Grafik die komplette Breite sowie 300 Pixel als Höhe ein.
Zudem werden die Daten für die beiden Achsen definiert. Während die X-Achse die Datumswerte der erhaltenen Daten anzeigt, dient die y-Achse zur Anzeige der Werte der Aktien, wobei diesem ein Dollarzeichen vorangestellt wird.

6. Mögliche Erweiterungen

Denkbar wäre noch die Funktionalität des Paper Tradings, beispielsweise mit der Alpaca API, wobei hier eine ähnliche Funktionalität wie das Planspiel der Börse denkbar wären, bei dem vor allem SchülerInnen und junge Menschen an den Finanzmarkt herangeführt werden, indem diese Aktien testweise erwerben und genau analysieren können, ob sie bezüglich der Gewinne, oder im schlechten Fall Verluste, die richtigen Entscheidungen vorgenommen haben.

7. Kritische Reflexion und Schluss

Bernd konnte schon umfangreiche Vorkenntnisse mit den verwendeten Tools vorzeigen, wodurch es für ihn möglich war, effizienter zum Projekt beizutragen. Joshua musste sich hingegen erst beispielsweise in next.js mithilfe von Bernd einarbeiten.
Außerdem war die Aufteilung der Aufgaben zwischenzeitlich nicht eindeutig.
Eine Lösung dieses Problems wurde jedoch schnell gefunden und wird im Nachfolgenden beschrieben:

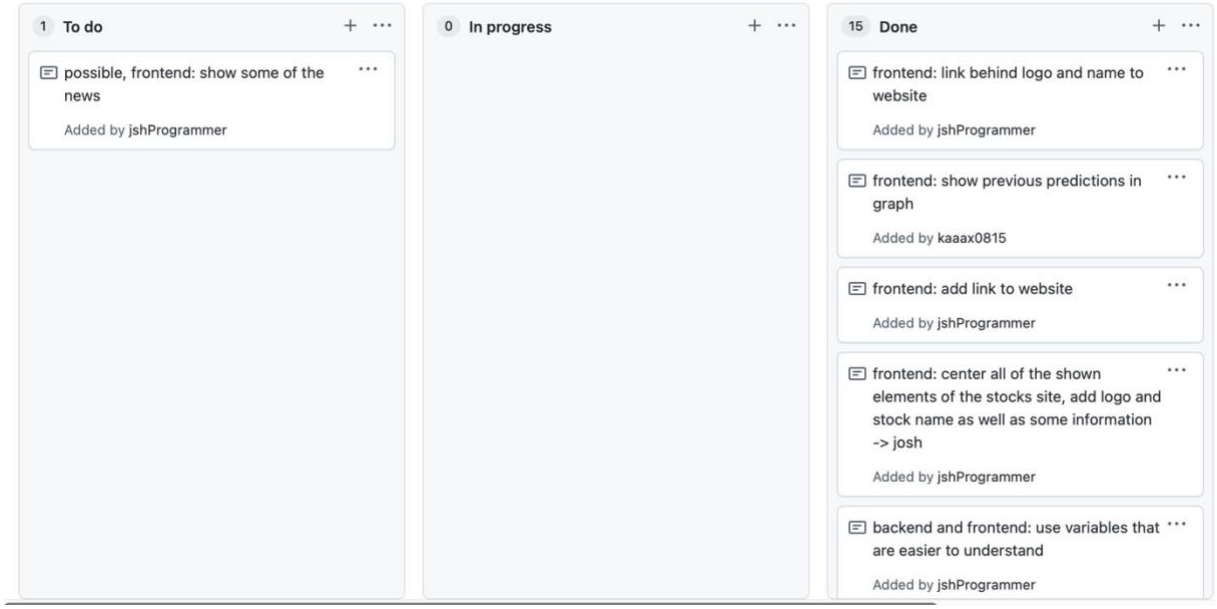

Abbildung 4: Projectboard auf GitHub

Die Einrichtung eines Projectboards auf GitHub stellte eine gute Entscheidung dar, da so übersichtlich und effizient festgestellt werden konnte, welche Aufgaben noch zu erledigen sind beziehungsweise gerade von einem anderen Teammitglied durchgeführt werden.