Kurzdokumentation

(TINF20C, WebEngineering2 - Team 1)

Project: *Aktien Website und Aktien Telegram Bot*

Team:

Linus Eickhoff (Rripped) (wi20034@lehre.dhbw-stuttgart.de)
Florian Kellermann (NormalParameter) (inf20141@lehre.dhbw-stuttgart.de)
Florian Kaiser (H4CK3R-01) (inf20155@lehre.dhbw-stuttgart.de)
Kevin Pauer (kevinpauer) (inf20003@lehre.dhbw-stuttgart.de)

Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	03.05.2022	Linus Eickhoff	Dokument angelegt und bearbeitet
1.0	08.05.2022	Alle	Fertiggestellt

CONTENTS

1. Vorwort

Dieses Dokument ist nur eine Kurzdokumentation des Projektes im Repository und in der ZIP-Datei sind ausführlichere Dokumentationen und Informationen im Ordner „documentation“ und in den README.md – Dateien zu finden.
Das Projekt ist im Rahmen der Vorlesung „Web Engineering 2“ im Kurs TINF20C entstanden. Das Projekt umfasst eine Datenbank, eine API, eine Website und einen Telegram Chat Bot. Die Webseite und der Telegram Bot können dafür genutzt werden, seine Aktien, Käufe und Verkäufe zu verwalten. Der Telegram Bot wird dazu genutzt regelmäßige Updates über das Portfolio und in Form von Nachrichten an das mobile Gerät des Users zu senden. Die Webseite und der Bot lassen sich miteinander verknüpfen, sodass man auch über ein mobiles Gerät den Account und die Transaktionen verwalten kann.
Das Projekt ist auf GitHub unter https://github.com/WebEngineering2/TelegramAktienBot zu finden.

2. Teamübersicht

Name und Github	Email und Matr. Nr.	Aufgaben
Linus Eickhoff (*Rripped*)	wi20034@lehre.dhbw-stuttgart.de (1943478)	Projektstruktur, Datenbankdesign, Dokumentation, Telegram Bot, API-Handler, News-fetching
Florian Kellermann (*NormalParameter*)	inf20141@lehre.dhbw-stuttgart.de (8838597)	Projektstruktur, Datenbankdesign, Dokumentation, Telegram Bot, Update-Scheduler, Stocks-fetching
Florian Kaiser (*H4CK3R-01*)	inf20155@lehre.dhbw-stuttgart.de (9829423)	Projektstruktur, Datenbankdesign, Dokumentation, Backend (Datenbank, API, Server, CI-CD)
Kevin Pauer (*kevinpauer*)	inf20003@lehre.dhbw-stuttgart.de (1199719)	Projektstruktur, Datenbankdesign, Dokumentation, Frontend (Website)

Tabelle 1: Qualitativer und quantitativer Projektnutzen

3. Setup und Zugangsdaten

Die Website ist über https://gruppe1.testsites.info/ erreichbar. Der Telegram Bot lässt sich am Smartphone über t.me/projektaktienbot hinzufügen. Im Login-Bereich der Website lässt sich dann ein neuer Account registrieren, anschließend lassen sich in den Account-Einstellungen Telegram und Website Account mit der Telegram ID verknüpfen. Die Telegram ID findet man über /id oder /auth im Bot heraus. Als User lassen sich nun alle Funktionalitäten nutzen, ADMIN Funktionen wie das Ausgeben aller User oder das Setzen von ADMIN-Rechten sind nur mit Admin Account möglich.
Um das Projekt lokal zu testen gibt es Setup-Anleitungen in den README Dateien der entsprechenden Ordner. Für das lokale Testen des Bots steht der Debug Bot (t.me/mynewdebugbot) zu Verfügung. Auf Anfrage senden wir gerne die .env Datei mit den entsprechenden API Keys und Daten für ein lokales Starten des Bots.

4. Zeitrahmen

Grober Zeitaufwand nach Person und Aufgabe, da die Commit Historie über 600 Commits lang ist haben wir ein Dokument mit allen Commits generiert zu finden im Ordner „documentation“ unter (ERGÄNZEN!!)

	Linus Eickhoff	Florian Kellermann	Florian Kaiser	Kevin Pauer
Dokumentation	15	15	15	15
Projektstrukturierung	5	5	5	5
Telegram Bot	40	40	5	5
Frontend (Website)	5	5	5	40
Backend (DB, API)	5	5	40	5

Tabelle 2: Projektplan

5. Projektanforderungen

Anforderungen an Projekt

1. **Entwickeln Sie eine Webanwendung auf Basis eines vorhandenen Webframeworks Ihrer gewählten Programmiersprache (bspw. ASP.NET MVC, Rails.js, CakePHP, Django, o.ä.)**

2. **Verbinden Sie eine Datenbank via einer ORM Komponente (bspw. Entity Framework, SQLAlchemy, Sequelize, o.ä.) und Implementieren Sie mindestens einmal die Funktionen „Insert, Update, Delete, Select"**

3. **Realisieren Sie mindestens eine einfache Benutzeroberfläche für die Interaktion mit Ihrer Applikation mit einem vorhandenen CSS-Framework wie bspw. Bootstrap (MVC-Ansatz) oder eine „Single Page Application" mit entsprechenden Datenschnittstellen.**

4. **Implementieren Sie eine sichere Benutzerregistrierung (Felder: Name / E-Mail / Kennwort / Kennwort wiederholen) sowie einen Benutzerlogin (Name oder E-Mail und Kennwort)**

   Erweiterung #1: Bieten Sie zusätzlich die Registrierung via oAuth oder WebAuthn an

5. **Implementieren Sie eine API (SOAP oder REST) zur Kommunikation mit einem Client. Unterstützen Sie dabei mindestens den Abruf von Informationen aus Ihrer Datenbank.**

   Erweiterung #1: Implementieren Sie die Autorisierung via JWT und stellen Sie ein dokumentiertes Postman Testprojekt zur Verfügung

   Erweiterung #2: Stellen Sie zusätzlich das Updaten und Löschen von Datensätze mit Postman Testprojekt um

6. **Richten Sie einen Webserver/Reverse Proxy in Ihrer Testumgebung ein und sichern Sie diese mit einem „Let's Encrypt" Zertifikat ab. Prüfen Sie Ihre Konfiguration mit „ssllabs.com" – es muss mindestens ein A als Bewertung erreicht werden.**

   Erweiterung #1: Erreichen Sie A+ mit Ihrer Konfiguration
   Erweiterung #2: Konfigurieren Sie http/2.0

7. **Ihre Website muss öffentlich erreichbar sein (Notwendiger DNS-Eintrag und Server wird gestellt) – die Einrichtung des Servers erfolgt in Gruppenarbeit – direkte Installation (Webserver, ggf. Runtimes und Datenbank) ist möglich, alternativ Nutzung von Docker**

8. Optimieren der Ressourcen: „Minify" von Javascript- und CSS-Dateien; Verkleinerte Grafiken. Prüfen Sie Ihre Optimierung mit „gtmetrix.com", erreichen Sie mindestens die Bewertung C.

   Erweiterung #1: Erreichen Sie eine Bewertung „A"

9. **Präsentation Ihres Projekts im Zeitumfang von max. 10 Minuten (Wird später noch verbindlich mitgeteilt) mit den Themen**
   a. Kurze Übersicht Team sowie Zeitaufwand und Verteilung auf Mitglieder
   b. Welche Frameworks und Technologien haben Sie verwendet? (Skriptsprache, Frameworks, Webserver, ggf. Protokolle)
   c. Kurzvorstellung des Projektes (Interaktive Vorstellung, ca. 10 Minuten)
   d. **Lessons Learned / Was waren die größten Probleme bei der Umsetzung?**

      Erweiterung #1: Empfehlung an andere Gruppen, bspw. Welche Tools/Komponenten haben besonders geholfen oder hatten einen Mehrwert, Welche Tipps & Tricks würden Sie an andere Gruppen weitergeben?

   e. Rückblick und Fazit

10. Optional: Verwenden Sie eine Quellcodeverwaltung

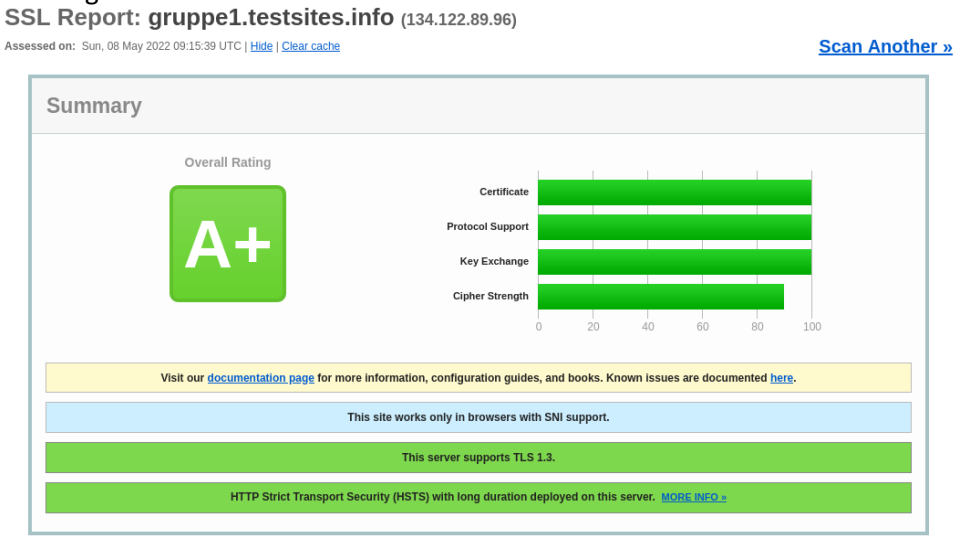
11. **Dokumentieren Sie bedeutende Stellen Ihres Quellcodes mit erklärendem Kommentartext**

12. **Kurzdokumentation**
    a. **Kurze Übersicht Team sowie Zeitaufwand und Verteilung auf Mitglieder mit Selbsteinschätzung der Projektnote**
    b. **Test-Zugangsdaten sowie Url (und weitere Informationen sofern notwendig) für die Begutachtung des Projekts**
    c. **Screenshots der Testergebnisse von gtmetrix.com und ssllabs.com der Projektseite**
    d. **Falls verwendet: Screenshot der Commit-Historie der Quellcodeverwaltung**

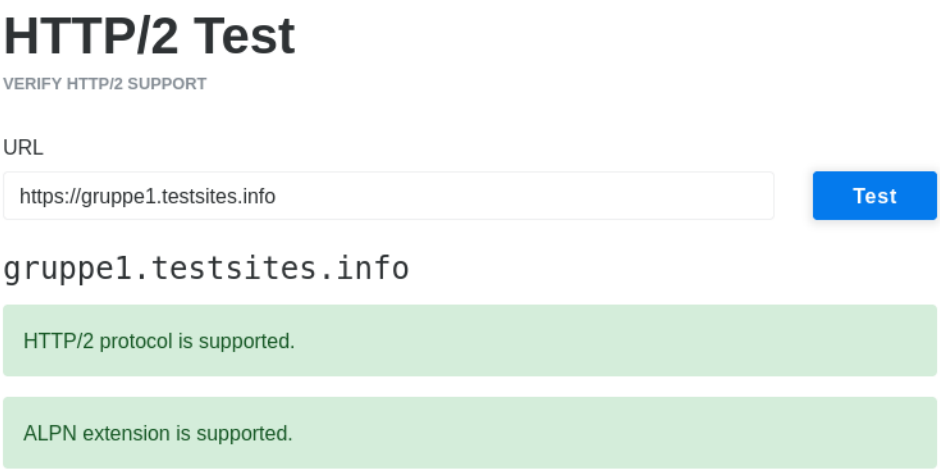
6. Umsetzung der Anforderungen

1. Das Frontend unsere Applikation wurde mit dem Web-Framework Angular realisiert.
2. Das Backend wurde mithilfe von APIFlask (https://github.com/apiflask/apiflask) und SQLAlchemy sowie einer MariaDB-Datenbank realisiert.
3. Angular berauht auf dem Single Page Application Prinzip, zusätzlich verwenden wir Bootstrap um ein interaktives Design zu gewährleisten.
4. Die Registrierung funktioniert über die API mithilfe von E-Mail, Username und Passwort. Das Passwort wird in der Datenbank in einer gehashten Form gespeichert, mithilfe von bcrypt sowie einem Salt.
   Die Login Page nimmt im Frontend die Userdaten auf und sendet eine HTTP Request an das Backend. Bei einer erfolgreichen Rückmeldung kann der User auf die Anwendung zugreifen.
   #1: Diese optionale Anforderung wurde nicht umgesetzt
5. Wie in 2. bereits erwähnt, wurde die API mithilfe von APIFlask verwirklicht.
   #1: Sobald der User sich über die API angemeldet hat, bekommt er als Antwort ein JSON-Objekt, das einen JWT-Token enthält. Das Postman Projekt lässt sich im “documentation”-Ordner finden
   #2: Das Postman Projekt lässt sich im “documentation”-Ordner finden
6. Als Reverse Proxy wird Traefik 2.6 (latest) verwendet. Der Reverse Proxy sowie die gesamte Anwendung laufen als Docker Container.

#1:



#2:



7. Für das Projekt wird Docker verwendet. Als Container werden Traefik (Reversae Proxy), MariaDB (Datenbank), phpMyAdmin (Datenbank-Verwaltung), Portainer (Docker-Verwaltung), Watchtower (Container-Aktualisierungen), Goaccess + Nginx (Analyse der Traefik-Logs) sowie die Container des Projektes (Frontend, API und Bot) verwendet.
8. Angular unterstützt Minification und Bundling bereits, so dass die Applikation schneller geladen werden kann.

#1:

9. Präsentation wird am 12.05.2022 fertiggestellt und hochgeladen, präsentiert wird am 17.05.2022
10. Wir verwenden GitHub für unser Projekt.
11. Der gesamte Code umfasst Kommentare und Funktionsbeschreibungen im Python Code des Projekts
12. Kurzdokumentation ist ebenfalls in vollem Umfang bereitgestellt (siehe dieses Dokument)

7. Selbsteinschätzung

Da wir wie oben erklärt alle Anforderungen, inklusive fast aller Erweiterungen (WebAuth/oAuth fehlt), vollständig erfüllt haben, viele Zusatztools genutzt und gute Dokumentation geschrieben haben und das Projekt, wie geplant in vollem Umfang umgesetzt wurde, schätzen wir in Anbetracht des hohen Zeit- und Energie Aufwandes unsere Projektnote auf Sehr gut (1,0 bis 1,3).