LB2 Module 295

Thema: Module 295 Description LB2

Dokumentinformationen

Dateiname:	LB2 295-SQL
Speicherdatum:	01.04.2021
Version:	0.1.x

Autoreninformationen

Autor:	Andre Gaillard
E-Mail:	andre@tie-international.com


TIE
INTERNATIONAL

Änderungsverlauf

#	Datum	Version	Geänderter Inhalt	Autor
	21.12.2021	v100	Formatted Document	Andre Gaillard
	07.07.2021	v110	Removed some functionalities	André Gaillard
	16.02.2023	V120	Update with new specifications	Patrick Engelhardt

1 Preface

1.1 Purpose

Dieses Dokument beschreibt die Aufgaben im LB2 des Moduls 295: Implementierung des Back-Ends für eine Anwendung.

1.2 Voraussetzungen

Eine funktionierende Installation von Nodejs. Eine Vorlage wird zur Verfügung gestellt. Ein Editor sollte bereits installiert sein. Wir empfehlen Visual Studio Code. Docker sollte installiert sein.

1.3 LB2 Details

Hilfsmittel:	Open Book
Gewicht:	70%
Gruppengröße:	Keine Gruppen
Dauer:	1-2h Installation am ersten Kurstag, 13 Stunden Programmierarbeit
Punkte:	70
Bewertung:	Linear

2 Das Projekt: Backend mit SQL für Krypto-Registrierungsformular

2.1 General Description

In diesem Projekt müssen Sie ein Backend für das Registrierungsformular auf einem Frontend einrichten, mit dem sich ein Benutzer für einen Kryptomarkt registrieren kann. Wenn ein Benutzer versucht, sich zu registrieren, müssen Sie ihn/sie zur Datenbank hinzufügen, aber nur, wenn sein/ihr Benutzername oder seine/ihre E-Mail noch nicht in unserer Datenbank existiert.
Wenn der Benutzer erfolgreich registriert werden konnte, wird ein Erfolgscode an das Frontend gesendet. Andernfalls muss ein Fehlercode mit einer vordefinierten Fehlermeldung als Antwort gesendet werden.
Wir werden eine Mischung aus Datenbank (Postgres) und Dateisystem (für die Speicherung der ID-Bilder) für die Speicherung unserer Daten verwenden. Das Basis-Setup für das Projekt wird von Ihrem Lehrer in Form eines Git-Repositorys zur Verfügung gestellt.

2.2 Technology Stack

Wie im Frontend verwenden wir NodeJS mit Typescript und ESLint. Wir empfehlen dringend, den Server mit Express und die Dateiverarbeitung mit Multer zu implementieren. Für die Protokollierung können Sie Winston verwenden.

2.3 Die Einrichtung

Richten Sie das Projekt ein, indem Sie der README-Datei im Repository folgen.

2.4 Die Datenbank

Wir müssen unsere Datenbank entsprechend der Spezifikation definieren. Wir müssen alles, was aus dem Formular des Frontends kommt, korrekt in der Datenbank speichern. Sie können die Spezifikation im Swagger-Ordner des Repos finden. Der Einfachheit halber listen wir sie hier auf. Über die Typen, die Sie den Einträgen geben wollen, müssen Sie selbst nachdenken:

- name
- address
- city
- phoneNumber
- country
- username
- postcode
- password
- date of birth
- id confirmation (this should be a file name. Note that there can be multiple files!)
- email

Damit ist noch nicht die gesamte Datenbank definiert. Sie müssen auch einen Primärschlüssel definieren, wobei wir eine automatisch inkrementierende ID empfehlen. Für den Rest der Spezifikation werden mehrere andere Felder benötigt, die im nächsten Abschnitt beschrieben werden:

- password salt

- registration time
- isEmailConfirmed (for email confirmation)

Wir empfehlen, die Datei init.sql zu verwenden, um die Datenbank gleich zu Beginn zu definieren. Der Inhalt der init.sql-Datei könnte wie folgt aussehen:

CREATE TABLE public.users (
id serial NOT NULL PRIMARY KEY,
other fields you want to add
)

Sie wird bei der Erstellung eines Containers kopiert, wodurch die relationale Datenbank mit der angegebenen Relation effektiv initialisiert wird. Danach erstellen Sie das Datenbank-Image wie in der README.md beschrieben.

Wann immer Sie das Gefühl haben, dass Sie etwas zur Datenbank hinzufügen oder aus ihr entfernen müssen, können Sie zwei Möglichkeiten nutzen:

1. Delete the docker containers and the image. Schreiben Sie init.sql neu und erstellen Sie das Docker-Image und den Container wie in der README beschrieben.
2. Verwenden Sie die Postgres-GUI oder die Befehlszeilenschnittstelle, um Spalten in der Tabelle hinzuzufügen oder zu entfernen (https://alvinalexander.com/blog/post/postgresql/log-in-postgresql-database/)

2.5 Der Sever

Wir werden einen einfachen Nodejs-Server schreiben, der einen eingehenden Post-Aufruf mit vielseitigen Daten aus dem Registrierungs-Frontend verarbeiten kann. Wir empfehlen die Verwendung von ExpressJS. Das Backend muss die folgenden Anforderungen erfüllen:

- Restful-API-Endpunkt, an den das Frontend Daten senden kann. Verwenden Sie dazu einen POST-Endpunkt mit dem Namen /login. Dieser Endpunkt sollte von dem Ihnen zur Verfügung gestellten Frontend angesteuert werden können
- Akzeptieren und Extrahieren der Multiform-Daten aus der Abfrage
- Extrahieren der Dateien (z.B. .pdf, .jpeg, .jpg, .png) aus dem Multiform
- Verschlüsseln Sie das Passwort mit einer geeigneten Bibliothek und einem Salt
- Prüfen Sie, ob der Benutzername oder die E-Mail bereits vergeben sind.
- Wenn nein, füge den Benutzer zur Datenbank hinzu
- Wenn ja, antworten Sie mit dem angegebenen HTTP-Code (405)
- Speichern Sie die Daten im Dateisystem (nur wenn die Registrierung abgeschlossen wurde)

Für den letzten Punkt empfehlen wir die Verwendung von Multer.

Außerdem benötigen wir eine saubere Implementierung eines Loggers, der die wichtigsten Ereignisse sinnvoll protokolliert. Beachten Sie, dass Sie die Ausgabe vorerst nur auf die Konsole transportieren können und erst wenn Sie am Ende Zeit übrig haben, sollten Sie mit der Konfiguration des Loggers beginnen.

2.6 Zusätzlich erforderliche Funktionalität

Darüber hinaus haben wir die folgenden Anforderungen an das Backend:

- Das Passwort muss zusammen mit seinem Salt als Hashwert gespeichert werden.
- Man braucht ein Flag in der Datenbank, das angibt, ob ein Benutzer seine E-Mail bestätigt hat
- Man muss in der Lage sein, den Dateinamen eines gespeicherten Dokuments aus der Datenbank abzufragen, daher braucht man einen Eintrag dafür
- Der Server muss HTTPS sein (d.h. ein selbst generiertes Zertifikat haben)
- Wenn ein Benutzer nicht akzeptiert wird, weil der Benutzername oder die E-Mail bereits vergeben ist, müssen Sie die Dateien aus dem Dateisystem entfernen.

2.7 Das Front-End

Sie erhalten den Build des Front-Ends für den zu implementierenden Server. Um es zu verwenden, müssen Sie den Server instanziieren und dann den Build-Ordner statisch bedienen. In express können Sie dazu die folgende Zeile am Anfang verwenden:

```
app.use(express.static(path.resolve(__dirname, '../client/build')));
```

IMPORTANT: Der Server muss unter localhost:3002 laufen.

Dann können Sie auf das Frontend zugreifen, indem Sie localhost:3002 in den Browser eingeben.

Experimentieren Sie damit, wie das Frontend mit dem Server interagiert. Es ist wichtig, dass die folgenden HTTPS-Statuscodes korrekt für die Interaktion mit dem Frontend verwendet werden:

- 200: Alles hat funktioniert und der Benutzer ist in der Datenbank gespeichert
- 400: Problem mit dem Format der Formulardaten
    - Fehlende Datei oder falsches Dateiformat
    - Fehlende Felder im Formular
- 405: Entweder der Benutzer oder die E-Mail-Adresse oder beide sind bereits in der Datenbank. Das Frontend unterscheidet zwischen den folgenden Meldungen und reagiert unterschiedlich:
    - „Doppelter Benutzername und doppelte E-Mail-Adresse": Das Frontend zeigt sowohl den Benutzernamen als auch die E-Mail rot an
    - „Benutzername bereits vergeben": Das Frontend zeigt den Benutzernamen rot an
    - „E-Mail bereits vergeben: Das Frontend zeigt die E-Mail an rot

2.8 Optionale Funktionalitäten

Für die Fehlersuche kann es hilfreich sein, eine Funktion zu implementieren, die die Datenbank leert, damit Sie keine veralteten oder Testdaten herumliegen haben.

2.9 Hinweis

Um Ihre Implementierung zu testen, ohne in das Frontend eingreifen zu müssen, empfehlen wir die Verwendung von Postman. Dort können Sie einfache HTTP-Anfragen definieren. Dies ist beispielsweise nützlich, um zu prüfen, was der Server tut, wenn Sie eine Anfrage mit einem falschen Format des Anfragebodys oder ohne Dateien senden.

3 Evaluation

Wir werden Sie in einer Reihe von Punkten testen, die unterschiedlich gewichtet sind. Wir geben Ihnen einen kurzen Überblick über die wichtigsten Punkte.

- Funktionalität (60%)
    - Haben Sie die Datenbank korrekt implementiert?
    - Behandeln Sie doppelte Benutzernamen korrekt?
    - Werden Bilder korrekt verarbeitet und extrahiert?
    - Ist der Code verständlich und vernünftig dokumentiert?
    - Funktioniert die Kommunikation mit dem Frontend?
- Sicherheit (~17%)
    - Haben Sie das Passwort richtig gehasht und gespeichert?
    - Haben Sie ein Zertifikat generiert und einen HTTPS-Server implementiert?
- Protokollierung (~23%)
    - Haben Sie einen konfigurierbaren Logger implementiert/verwendet?
    - Verwenden Sie für die Protokollierung verschiedene Grade von Meldungen (Warnung, Fehler usw.)
    - Gibt Ihr Logger verständliche und aussagekräftige Meldungen aus?

Sie werden durch eine Kombination aus Code-Inspektion und Postman-Anfragen bewertet. Der Ausbilder wird prüfen, ob Sie bei falschen Anfragen korrekt protokollieren. Außerdem ist es von größter Bedeutung, dass keine ungültigen Benutzer in der Datenbank landen. Die Speicherung in der Datenbank ist teuer und es darf keine Sicherheitslücken geben.

4 Submission

Pushen Sie den Code in das angegebene Repo, wie von Ihrem Ausbilder bereitgestellt oder als Zip-Datei.