31. OKTOBER 2023

M295 SHOP BACKEND PROJEKT

DRUCKDATUM 31.10.2023

Kontaktinformation:
E-Mail: julian.brecht@ict.csbe.ch

JULIAN BRECHT
BIS22 MODUL 295

Inhalt

Einleitung

Dies ist meine Dokumentation über das Schulprojekt zur Erstellung eines einfachen Backend für einen Onlineshop.

Das Projekt wird mit Spring Boot umgesetzt und ziel darauf ab, ein verwendbares Backend für einen Onlineshop zu erstellen.

Benutzer sollen mithilfe dieser Produkte, Kategorien und User erfassen können.
Diese sollen in einer MariaDB gespeichert, abgerufen und verändert werden können.
Die Benutzer sollen ausserdem über Zwei verschiedene Zugriffslevel verfügen (Admin und User), damit nicht jeder Benutzer Änderungen in der Datenbank machen kann.

Analyse

Zielgruppe

Die Zielgruppe dieses Projektes schliesst jeden ein der eine API für ein Shop-backend braucht.

Da diese Zielgruppe sehr breit gefächert ist, sollte die API so Benutzerfreundlich wie möglich erstellt werden.

Ausgangslage (Ist-Situation)

Aktuell gibt es einige APIs, die im Internet bereitstehen, um ein Shop-backend zu realisieren.

Da wir allerdings so unabhängig wie möglich von anderen Softwareprodukten sein wollen, haben wir uns entschieden ein eigenes API mit Spring Boot und einer MariaDB umzusetzen.

Unabhängigkeit von anderen APIs gibt uns die Freiheit unsere eigene nach unseren speziellen Wünschen zu gestalten und zu verwalten.

Da es ausserdem für viele bereits verfügbaren Shop-backend APIs keine Garantie für einen längeren Support und Entwicklung gibt, ist es eine zusätzliche Sicherheit für uns, eine eigene API zu haben.

Projektziel (Soll-Situation)

Am Ende des Projekts soll eine vollständige API für ein Shop-backend realisiert sein.

Diese soll die Registration und Autorisierung (Logins) mittels JWT verwalten können.
Die User, die sich Registriert haben sollen, dabei in unserer Datenbank mit einem verschlüsselten Passwort gespeichert werden.

Die API soll ausserdem in der Lage seine CRUD-Operationen in unserer Datenbank durchzuführen, damit wir mit Hilfe dieser unseren Shop verwalten können.

Anforderungen

Tabelle 1 Anforderungen

#	**Anforderung**
1	Neue Kunden sollen sich Registrieren können.
2	Kunden sollen sich einloggen können und dabei ein JWT erhalten um sich zu Authentifizieren.
3	Es soll eine Datenbank im Hintergrund sein, welche Produkte, Kategorien und Benutzer speichert
4	Es soll nur Administratoren gestattet neue Produkte und Kategorien zu erstellen.
5	Es soll nur Administratoren gestattet seine Produkte, Kategorien und User in der Datenbank zu verändern.
6	Es soll nur Administratoren gestattet Produkte, Kategorien und Benutzer zu löschen.
7	Es soll jedem möglich sein unsere Produkte und Kategorien anzuschauen und aufzulisten
8	Ein Administrator soll einen anderen Benutzer zum Administrator befördern können.

Planung

Die API soll mit Spring Boot und einer MariaDB umgesetzt werden.

In diesem Abschnitt geht es um die Planung der API.

Packages und Klassen

Tabelle 2 Products

Package	ch.csbe.productstore.products
Products	Definiert die Eigenschaften eines Produktes und wie dieses in der DB gespeichert wird. Hat dazugehörige Getter und Setter.
ProductsController	Verantwortlich für die Request-Mappings und die auszuführenden Funktionen
ProductsDto	Data Transfer Objekt welches zum Ausgeben von Produkten und Kategorien benötigt wird.
ProductsRepository	Interface welche Kommunikation mit der Datenbank ermöglicht.
ProductsService	Enthält alle Methoden, die ausgeführt werden, wenn ein Endpoint mit dem entsprechendem Request aufgerufen wird.

Tabelle 3 Categories

Package	ch.csbe.productstore.categories
Categories	Definiert die Eigenschaften einer Kategorie und wie dieses in der DB gespeichert wird. Hat dazugehörige Getter und Setter.
CategoriesController	Verantwortlich für die Request-Mappings und die auszuführenden Funktionen
CategoriesDto	Data Transfer Objekt welches zum Ausgeben von Kategorien und Produkten benötigt wird.
CategoriesRepository	Interface welche Kommunikation mit der Datenbank ermöglicht.
CategoriesService	Enthält alle Methoden, die ausgeführt werden, wenn ein Endpoint mit dem entsprechendem Request aufgerufen wird.

Tabelle 4 User

Package	ch.csbe.productstore.user;
User	Definiert die Eigenschaften eines Users und wie dieses in der DB gespeichert wird. Hat dazugehörige Getter und Setter.
UserController	Verantwortlich für die Request-Mappings und die auszuführenden Funktionen
UserDto	Data Transfer Objekt welches zum Anpassen eines Users und Login benötigt wird.
UserRepository	Interface welche Kommunikation mit der Datenbank ermöglicht. Enthählt ausserdem die Funktion einen User mit dem Usernamen zu finden.
UserService	Enthält alle Methoden, die ausgeführt werden, wenn ein Endpoint mit dem entsprechendem Request aufgerufen wird.

Tabelle 5 Auth

Package	ch.csbe.productstore.auth
AuthConfig	Enthält die Konfiguration welche Endpoints für alle Freigeschalten und welche nur für Benutzer mit Administratoren Recht freigeschalten sind.
AuthService	Sorgt dafür, dass wenn ein User sich erfolgreich einloggt, ein gültiges JWT erhält.
JwtFilter	Filter, welcher HTTP Requests auf ein gültiges JWT prüft um User zu Authentifizieren
JwtService	Ist für die Erstellung des JWTs zuständig.

Namensschema

Tabelle 6 Namesschema

Element	**Schreibweise**	**Beispiel**
Package	lower case mit punkten getrennt	ch.csbe.packagename
Klasse	Pascal Case	Klasse, MeineKlasse
Variabel	Camel Case	variabel, meineVariabel
Methode	Camel Case	method, myMethod
Konstante	all Uppercase mit _ getrennt	KONSTANTE, MY_KONSTANTE
Endpoints	all lowercase mit - getrennt mit / weitergeführt	endpoint, my-endpoint, my-endpoint/{id}

Endpoints

Tabelle 7 Endpoints

Entität	URL	http-verb	Authentifiziert	Request Body	Response Body	Beschreibung
Products	/product	GET	Nein	-	Liste von Produkten	Gibt eine Liste aller vorhandenen Produkte zurück
Products	/product/{id}	GET	Nein	-	Einzelnes Produkt	Gibt ein einzelnes Produkt anhand der ID wieder
Products	/product/{id}	POST	Admin	Produkt	"Produkt erstellt"	Erstellt ein Produkt aus dem im Request Body mitgegebenem JSON und fügt es der Kategorie der mitgegebenen ID hinzu
Products	/product/{id}	PUT	Admin	Produkt	"Id {id} {Produktname} was updated"	Updated ein Produkt mit den Werten des im Request befindlichem JSON mit der entsprechenden ID
Products	/product/{id}	DELETE	Admin	-	-	Löscht das Produkt mit der angegeben ID
Categories	/category	GET	Nein	-	Liste von Kategorien	Gibt eine Liste aller vorhandenen Kategorien zurück
Categories	/category/{id}	GET	Nein	-	Einzelne Kategorie	Gibt eine einzelne Kategorie zurück
Categories	/category/{id}/products	GET	Nein	-	Alle Produkte einer Kategorie	Gibt eine Liste aus allen Produkten, die sich in einer Kategorie befinden zurück
Categories	/category	POST	Admin	Category	Category JSON	Erstellt eine neue Kategorie
Categories	/category/{id}	PUT	Admin	Category	Category JSON	Updated eine Kategorie mit den Werten des im Request befindlichem JSON mit der entsprechenden ID
Categories	/category/{id}	DELETE	Admin	-	-	Löscht die Kategorie mit der angegeben ID
User	/user	POST	Nein	User	"User Created"	Registriert einen neuen User
User	/user/login	POST	Nein	User	JWT wenn einloggen erfolgreich	Gibt das JWT des erfolgreich eingeloggten Users wieder
User	/make-admin/{id}	PUT	Admin	User	"user is made admin"	Stuft einen User zum Admin hoch

Umsetzung

Wie in der Planung beschrieben wurde die API mittels Spring Boot und einer MariaDB umgesetzt.

Dependencies

Das Backend hat die folgenden Dependencies:

Tabelle 8 Dependencies

Dependency	ArtiofactId	Funktion
MariaDB.jdbc	mariadb-java-client	Treiber für die MariaDB
spring-boot-starter-data-jpa	spring-boot-starter-data-jpa	Zuständig für die Übersetzung von Java zu Datenbankoperationen.
jsonwebtoken	Jjwt	Zuständig für die Erstellung und Verwendung von JWT
Springdoc	springdoc-openapi-starter-webmvc-ui	Zuständig für die Swagger Dokumentation der App
flywaydb	Flyway-mysql / Flyway-core	Wird gebraucht, um Initiale Daten in die DB zu laden
Spring Security	spring-boot-starter-security	Wird gebraucht, um Berechtigungen und Authentifizierung zu setzen und zu verwalten

Authentifizierung

Die Authentifizierung erfolgt wie in den Anforderungen genannt über Jason Web Token.

Wenn sich ein Registrierter Benutzer mit dem richtigem Benutzernamen und Passwort anmeldet, wir ihm ein entsprechendes JWT generiert, welches er für die Authentifizierung nutzen kann.

Das System prüft ausserdem anhand dieses JWTs ob der Benutzer ein Administrator ist, indem es einen Abgleich in der Datenbank mit der Spalte „Admin“ macht.

Endpoints, die wie in der Planung nur mit einer Administrator Authentifizierung aufgerufen werden sollen können, werden so vor zugriffen von normalen Benutzern und nicht Authentifizierten geschützt, da diese ohne dieses JWT nicht aufrufbar sind.

User

Registration

Personen, die noch keinen Account in unserem Backend haben, können sich wie in den Vorgaben verlangt über den Endpoint „/user“ registrieren.

Bei der Registration muss im Requestbody der Username und das Passwort im JSON Format mitgegeben werden.

Der User wird dann in der Datenbank mit dem entsprechenden Usernamen und gehashtem Passwort abgespeichert. Dabei ist zu beachten, dass er automatisch bei „Admin“ aus Sicherheitsgründen ein false eingetragen bekommt.

Login

Ein bereits Registrierter Benutzer kann sich über den Endpoint „/user/login“ in das System einloggen.

Hierzu müssen im Requestbody Benutzername und Passwort im JSON-Format angegeben werden. Wenn der Benutzername mit dem Passwort übereinstimmen, erhält der Benutzer ein JWT mit welchem er sich Authentifizieren kann.

Products

Produkte können, wie in den Anforderungen gefragt, nur von einem Administrator über die vorher geplanten Endpoints erstellt, verändert oder gelöscht werden.

Produkte können von jedem Benutzer, egal ob authentifiziert oder nicht, über die entsprechend vorher geplanten Endpoints eingesehen oder aufgelistet werden.

Categories

Kategorien können, wie in den Anforderungen gefragt, nur von einem Administrator über die vorher geplanten Endpoints erstellt, verändert oder gelöscht werden.

Kategorien können von jedem Benutzer, egal ob authentifiziert oder nicht, über die entsprechend vorher geplanten Endpoints eingesehen oder aufgelistet werden.

Es ist über den Endpoint „/category/{id}/products“ ausserdem auch möglich, alle Produkte einer Kategorie aufzulisten, egal ob authentifiziert oder nicht.

Datenbank

Die Datenbank wurde wie verlangt mit MariaDB umgesetzt

Datenbank Schema

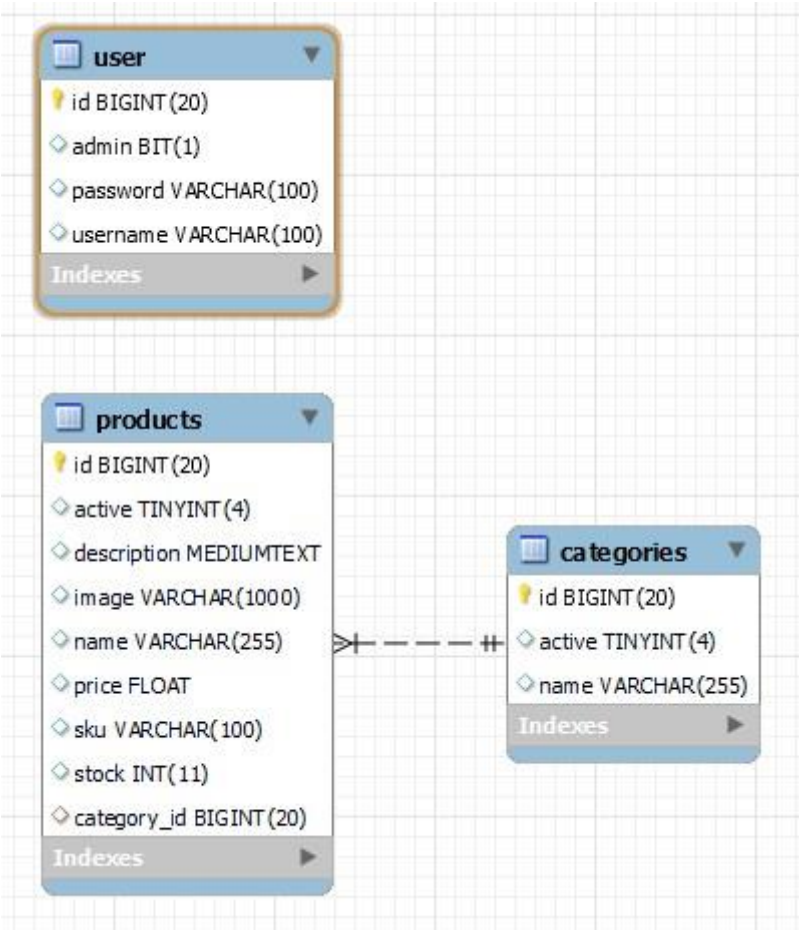

Abbildung 1 Datenbank Schema

Datenbank Seed

Die Datenbank wird über die dependency „flyway-mysql“ erstellt und mit initialen Daten versorgt.

Der Administrator Benutzer hat den Benutzernamen: „Admin“ und das Passwort „123456789“ welches gehasht abgespeichert wird.

Arbeitsjournal

Tabelle 9 Arbeitsjournal Tag1

Tag 1	
Tätigkeit	**Arbeitszeit (h)**
Planung des Projekts	6
Erstellung der Dokumentation	0,5
Erstellung des Projektes und der Klassen	2
Erstellung von Test Requests	1

Reflexion

Die Planung des Projektes hat mehr Zeit in Anspruch genommen als Anfangs gedacht.
Da ich dieses Framework zum ersten Mal Benutze wusste ich nicht gleich wie die Klassen miteinander interagieren. Nach einigen Tests ist es mir allerdings klar geworden und ich konnte das Projekt erfolgreich planen.

Da ich nun schon mehrere Dokumentationen in der Schulzeit verfasst habe, ist es ein leichtes gewesen die Struktur der Dokumentation zu erstellen.

Die Erstellung des Projekts, Packages, Klassen ging ohne grössere Probleme von statten. Die Seite https://start.spring.io/ hat die Erstellung des Projektes stark vereinfacht und ich konnte dort bereits die meisten Dependencies direkt im Projekt mit einbinden die ich benötige.

Nach dem Erstellen der ersten Controller und Services habe ich Test Requests geschrieben, um die Funktion dieser sicherzustellen. Auch dies hat ohne Probleme funktioniert.

Tabelle 10 Arbeitsjournal Tag2

Tag 2	
Tätigkeit	**Arbeitszeit (h)**
Schreiben von weiteren Service Klassen	1
Erstellung von DTOs	0,5
Erstellung von Funktion alle Produkte einer Kategorie auszugeben	1
Schreiben von weiteren Test Requests	0,5

Reflexion

Das Schreiben des User Services und des Categorie Services ging gut voran. Es gab kleine Stolpler welche nach kurzem Debugging beseitigt werden konnten

Das Erstellen der DTOs war einfach. Es gab während dieser Tätigkeit keine Probleme.

Die Funktion alle Produkte einer gewählten Kategorie auszugeben war anfangs etwas holprig. Nach einigen Versuchen und Tests hat aber auch dies geklappt.

Die weiteren Test Requests dienen dazu die Produktauflistungsfunktion zu Testen und haben nach einigen Umschreiben der Methode schliesslich erfolgreich funktioniert.

Tabelle 11 Arbeitsjournal Tag3

Tag 3	
Tätigkeit	**Arbeitszeit (h)**
Einbindung Spring Security und Erstellung der Auth Config	1
Erstellung des JWT-Filters	2
Erstellung von Funktion zum Einloggen und Registrieren	0,5

Reflexion

Das Einbinden von Spring Security war einfach, da man nur die Dependency im Pom File hinterlegen musste. Die Konfiguration für diese war dafür schwerer. Ich hatte erst Probleme diese zu Verstehen. Nach nochmaligem anschauen des Unterrichts zu diesem Thema wurde mir diese allerdings klar.

Auch der JWT-Filter hatte mir Anfangs grosse Probleme bereitet. Ich habe erst gar nicht verstanden, wie dieser Funktionieren sollte. Auch hier habe ich mir den Unterricht zu diesem Therma noch einmal anschauen müssen, um dieses umsetzen zu können.

Die Funktion zum Einloggen und um sich zu Registrieren ging dafür dann wieder einfacher, da ich zu diesem Zeitpunkt den Unterricht bereits ein zweites Mal angeschaut habe.

Tabelle 12 Arbeitsjournal Tag4

Tag 4	
Tätigkeit	**Arbeitszeit (h)**
Erstellung der Methode, um User zum Admin zu promoten	1
Anpassung des JWT-Filters und der Auth Config	2
Test der Endpoints mit Autorisation	1

Reflexion

Die Erstellung der Methode, um User zum Admin zu promoten, wenn im Request eine Admin Authorization mitgegeben wird, war anfangs etwas schwierig. Ich habe diese Methode des Öfteren neu schreiben müssen, da sie nie so funktioniert hat, wie ich dies wollte. Nach vielen Tests und Versuchen hat es dann aber funktioniert.

Die Anpassung des JWT-Filters darauf das er Anfragen auf gewisse Endpoints nicht Filtern soll, war anfangs schwierig. Es hat auch hier länger als gewünscht gedauert, bis dies Funktioniert hat. Das Anpassen der Auth Config auf die Endpoints, die eine Admin Authorization benötigen und die die keine brauchen war eine kleine Herausforderung, die im Gegensatz zum JWT-Filter aber einfach zu überwinden war.

Die Tests der Endpoints und wie diese auf die Authorization reagieren ging gut von Statten. Es gab hier und da ein paar kleine Fehler, die aber schnell ausgemerzt werden konnten. In dem Excel File „Authorization Test“ kann man sich die Endergebnisse anschauen.

Tabelle 13 Arbeitsjournal Tag5

Tag 5	
Tätigkeit	**Arbeitszeit (h)**
Hinzufügen von Flyway	0,5
Ausprobieren von Flyway	4
Formattierung von Code	2

Reflexion

Das Hinzufügen von Flyway war einfach. Auch dort konnte man die Dependency einfach im Pom File hinterlegen.

Flyway hat eine spezielle Schreibweise für Filenamen. Diese war anfangs nicht klar nachzuvollziehen, was nach einiger Zeit dann aber doch gut funktioniert hat. Da Flyway einfach SQL-Befehle, die in diesen Files hinterlegt werden, ausführt, war es nicht schwer mit diesen zu experimentieren.

Der Code wurde von mir neu formatiert, da diese r viele leere Zeilen, falsche Einrückungen enthielt.

Tabelle 14 Arbeitsjournal Tag6

Tag 6	
Tätigkeit	**Arbeitszeit (h)**
Hinzufügen von Code Kommentaren	2,5
DDL der Datenbank auf Flyway geschrieben	2
Hinzufügen von Swagger	1

Reflexion

Das Hinzufügen von Code Kommentaren war zwar einfach aber eine Fleissarbeit. Ich werde dies beim nächsten Projekt direkt beim Schreiben machen, damit ich nicht wieder in jede Klasse zurückschauen muss und Kommentare im Nachhinein verfassen muss.

Ich habe nach dem Unterricht am 28.10.2023 meine Flyway Files so angepasst, dass sie die Erstellung der DB übernehmen. Dies hilft meiner Meinung nach die Datenbank einfacher replizieren zu können. Ich finde es zudem genauer Beziehungen und Datentypen selbst in SQL definieren zu können. Das einfügen Initialer Daten wurde dadurch ebenfalls einfacher.
An dieser Stelle noch einmal an grosses Danke an Stefano das er uns dies gezeigt hat.

Das hinzufügen von Swagger war auch hier wieder einfach. Es ging wieder einfach über das Pom File. Die ersten Swagger Annotationen waren auch einfach zu schreiben. Es gab allerdings Probleme mit dem Aufrufen der Swagger UI Seite. Diese wurde durch Spring Security geblockt. Nachdem ich aber die Präsentation über Spring Security nochmals studierte, habe ich den Punkt gefunden, wie man diese Freischalten konnte.

Tabelle 15 Arbeitsjournal Tag7

Tag 7	
Tätigkeit	**Arbeitszeit (h)**
Hinzufügen von weiteren Swagger Annotationen	2,5
Erstellung von Methode um Produkte ohne Kategorie hinzuzufügen	1
Felder welche ausgefüllt werden müssen auf Mandatory gestellt	0,5

Reflexion

Das Hinzufügen von weiteren Swagger Annotationen insbesondere der Parameter Beschreibungen und der Beschreibung von Methoden an sich, ging gut voran. Es hat zwar etwas herumprobieren gebraucht, aber nachdem dies für die ersten Requests gepasst hat, war es ein leichtes dies für die anderen auch anzuwenden.

Die Erstellung der Methode zum hinzufügen von Produkten ohne Kategorie war nicht schwer. Das richtige Mapping und die Richtige Anwendung dieser hat allerdings doch wieder einige Zeit gebraucht, da ich nicht ganz sicher war, wie ich die mappen sollte. Hat aber auch wieder nach einiger Zeit geklappt.

Die Mandatory Fields Anpassung war einfach. Dies war eine kleine Anpassung, um die Swagger Dokumentation verständlicher zu machen und damit die Endpoints besser funktionieren

Tabelle 16 Arbeitsjournal Tag8

Tag 8	
Tätigkeit	**Arbeitszeit (h)**
Abschluss der Dokumentation (Rechtschreibprüfung, Schönheitsarbeit und Export)	4

Reflexion

Der Abschluss der Dokumentation ging gut voran, es waren hier und da noch einige kleine Fehler, die aber schnell ausgebessert werden konnten.

Das Projekt ist nun abgeschlossen und wird abgegeben.

Alles in allem hat mir die Arbeit am Projekt sehr viel Spass bereitet.
Es war vom Umfang her herausfordernd aber grade diese Herausforderung hat mich angespornt mein Bestes zu geben.
Zudem wird man nur in etwas besser, wenn man sich Herausforderungen stellt.

Ich konnte durch dieses Projekt ausserdem das Framework Spring Boot kennenlernen und dadurch das dieses Projekt einen direkten Bezug zur Realität hatte, hat das Lernen doppelt so viel Spass gemacht.

Ich bedanke mich an dieser Stelle noch einmal bei Daniel Schmitz und Stefano Mavilio für den fabelhaften Unterricht und freue mich auch in Zukunft wieder Unterricht mit Ihnen zu haben.

Quellenverzeichnis

Internet

Bilder:

Spring Boot logo
https://www.inovex.de/wp-content/uploads/2021/04/training-spring-boot.png

Websites:

https://start.spring.io/

https://spring.io/projects/spring-boot

https://spring.io/projects/spring-security

https://swagger.io/

https://www.openapis.org/

https://howtodoinjava.com/spring-mvc/controller-getmapping-postmapping/

Videos:

Titel: Spring Boot Tutorial | Full Course [2023] [NEW]
Autor: Amigoscode
https://www.youtube.com/watch?v=9SGDpanrc8U
11.10.2023 16:15

Titel: Spring Boot 3 + Spring Security 6 - JWT Authentication and Authorisation [NEW] [2023]
Autor: Amigoscode
https://www.youtube.com/watch?v=KxqlJblhzfI
21.10.2023 19:12

Mündliche Überlieferungen:

Unterricht von Daniel Schmitz

Unterricht von Stefano Mavilio

Abbildungsverzeichnis

Tabellenverzeichnis