BACKEND FÜR APPLIKATIONEN REALISIEREN

Bis 22, Modul_UK295

OCTOBER 28, 2023

DANIEL.BOGDANOV@ICT.CSBE.CH
Daniel Schmitz

Inhaltsverzeichnis

Einleitung

Eine Einleitung ist der erste Abschnitt oder Teil eines Textes, einer Rede, eines Aufsatzes oder einer anderen schriftlichen oder mündlichen Arbeit.

Herzlich willkommen zur Dokumentation des Moduls UK295, das unter der Leitung von Herrn Daniel Schmitz die Entwicklung des Backends für Anwendungen behandelt. In diesem Modul werden wir die Kommunikation zwischen einer Datenbank, unterstützt von MariaDB und MySQL Workbench, etablieren. Unser Ziel ist es, mithilfe unserer Programmierkenntnisse eine effiziente Kommunikation herzustellen, die über Postman gesteuert werden kann.

Die präzise Programmierung wird verschiedene Zugriffsrechte berücksichtigen, insbesondere für Administratoren und Benutzer. Dies bedeutet, dass bestimmte Rechte ausschliesslich dem Administrator vorbehalten sind. Wir werden uns jedoch auf die Implementierung und Umsetzung konzentrieren, um sicherzustellen, dass alle Endpunkte strukturiert und effizient gestaltet sind.

Wie funktioniert das Alles:

In Übereinstimmung mit festgelegten Richtlinien haben wir eine umfassende Kommunikationsschnittstelle für das Backend entwickelt. Ein nicht authentifizierter Benutzer hat die Möglichkeit, sich zu registrieren und erhält nach erfolgreicher Anmeldung ein JWT-Sicherheitstoken. Im Anschluss haben wir eine Logik implementiert, die unmittelbar anhand des Tokens feststellt, ob der Benutzer Standardberechtigungen besitzt oder Administratorrechte innehat. Ein Administrator verfügt über weitreichende Berechtigungen, einschliesslich der Manipulation von Daten durch Löschen, Aktualisieren und Bearbeiten. Ein normaler Benutzer hingegen ist auf eingeschränkte Aktionen beschränkt, nämlich das Herstellen einer Verbindung, das Anmelden und das Anzeigen von Benutzern. All dies wurde sorgfältig mit klar definierten Logiken und sauberem Code in jeder Schnittstelle umgesetzt.

Problemraum und Lösungsraum

Sind Konzepte, die in der Problemlösung verwendet werden, um den Rahmen für die Analyse und Lösung eines Problems zu definieren.

Problemraum

Der Problemraum basiert auf einem System, das die Verwaltung von Benutzern, Produkten und Produktkategorien ermöglicht. Es gibt mehrere Sicherheits und Funktionalitätsprobleme, die gelöst werden müssen:

Sicherheitslücken bei der Authentifizierung:

a. Es besteht die Gefahr, dass ein nicht authentifizierter Benutzer sich im System authentifizieren kann, was ein erhebliches Sicherheitsrisiko darstellt.

b. Es ist möglich, dass nicht autorisierte Benutzer Konten erstellen können, was die Integrität des Systems gefährdet.

Zugriffsrechte und Administration:

a. Ein authentifizierter Benutzer mit Administrationsrechten kann andere Benutzer zu Administratoren befördern, was missbräuchlich genutzt werden könnte. Dies stellt ein Risiko für die Verwaltung des Systems dar

Datenbankinitialisierung und Integrität:

a. Während der Initialisierung werden Daten in die Datenbank geladen, und die Qualität und Konsistenz dieser Daten sind nicht ausreichend sichergestellt.

Produkt und Produktkategorienverwaltung:

a. Ein authentifizierter Benutzer mit Administrationsrechten kann Produkte und Produktkategorien erstellen, bearbeiten und löschen, was zu Inkonsistenzen und Fehlern führen kann.

b. Authentifizierte und nicht authentifizierte Benutzer haben derzeit uneingeschränkten Zugriff auf Produkt und Produktkategorienlisten

Nichtfunktionale Anforderungen:

REST-Richtlinien: Es ist erforderlich, dass die Applikation die REST-Richtlinien erfüllt und Endpoints einheitlich und sinnvoll benannt sind

Datenbankanbindung: Die Datenbankanbindung erfolgt derzeit mittels Spring JDBC, aber es könnten Effizienz und Skalierbarkeitsfragen auftreten.

Authentifizierung mit JWT: Die Authentifizierung für geschützte Endpoints erfolgt mithilfe von JWT, jedoch müssen möglicherweise Sicherheitsverbesserungen implementiert werden.

Administratorenrechte: Es ist erforderlich, dass stets mindestens ein Benutzer mit Administratorenrechten im System vorhanden ist, um die Systemverwaltung sicherzustellen.

Passwortsicherheit: Die Passwörter werden derzeit mit einem starken Verschlüsselungsalgorithmus geschützt, was positiv ist, jedoch müssen sicherheitsrelevante Aspekte weiterhin überwacht werden.

Lösungsraum:

Um die oben genannten Probleme im Problemraum zu lösen, habe ich folgende Schritte in Betracht gezogen:

Sicherheitslücken bei der Authentifizierung:

Die Authentifizierung muss überarbeitet werden, um sicherzustellen, dass nicht authentifizierte Benutzer keinen Zugriff auf geschützte Bereiche erhalten und keine Konten erstellen können.

Zugriffsrechte und Administration:

Die Verwaltung von Benutzerrechten muss verbessert werden, um missbräuchliche Beförderungen zu verhindern.

Datenbankinitialisierung und Integrität:

Es ist notwendig, sorgfältige Validierungen und Datenintegritätsprüfungen während der Datenbankinitialisierung durchzuführen.

Produkt und Produktkategorienverwaltung:

Die Berechtigungen und Zugriffssteuerung für Produkt und Produktkategorienverwaltung sollten überarbeitet werden, um die Datenintegrität sicherzustellen.

Nichtfunktionale Anforderungen:

REST-Richtlinien: Die Endpoints sollten gemäss den REST-Richtlinien überarbeitet und sinnvoll benannt werden, um die Benutzerfreundlichkeit und Konsistenz zu verbessern.

Datenbankanbindung: Die Datenbankanbindung sollte überprüft werden, um Effizienz und Skalierbarkeit sicherzustellen. Eine alternative Datenbanktechnologie könnte in Betracht gezogen werden.

Authentifizierung mit JWT: Die Sicherheit der JWT-Authentifizierung sollte überprüft und verbessert werden, um potenzielle Schwachstellen zu beheben.

Administratorenrechte: Es ist erforderlich, eine Lösung zu implementieren, um sicherzustellen, dass stets mindestens ein Benutzer mit Administratorenrechten im System vorhanden ist.

Passwortsicherheit: Die Passwortsicherheit sollte weiterhin überwacht und auf dem neuesten Stand gehalten werden, um sicherheitsrelevante Bedrohungen zu minimieren.

User Stories

Eine User Story ist in der Regel eine kurze, einfache Aussage, die den Zweck und den Nutzen einer Funktion oder eines Features beschreibt. '' Als [Rolle] möcht ich das [Anliegen] damit [Grund]

User Story 1: Sicherheitsverbesserte Authentifizierung

Als nicht authentifizierter Benutzer möchte ich sicher und einfach in der Anwendung authentifiziert werden, ohne die Gefahr von Sicherheitslücken. Ich erwarte, dass die Authentifizierung schnell und unkompliziert ist, und dass nur autorisierte Benutzer Zugriff auf geschützte Bereiche haben.

User Story 2: Administratorrechte verwalten

Als Administrator möchte ich sicherstellen, dass nur autorisierte Benutzer die Berechtigung haben, andere Benutzer zu Administratoren zu befördern. Ich erwarte, dass die Verwaltung von Benutzerrechten missbräuchliche Beförderungen verhindert und die Systemverwaltung sichert.

User Story 3: Datenbankinitialisierung und Integrität

Als Datenbankadministrator möchte ich während der Datenbankinitialisierung gründliche Validierungen und Datenintegritätsprüfungen durchführen, um die Qualität und Konsistenz der Daten sicherzustellen.

User Story 4: Zugriffskontrolle für Produkte und Kategorien

Als Produktmanager möchte ich die Zugriffssteuerung für die Produkt- und Produktkategorienverwaltung verbessern, um sicherzustellen, dass nur autorisierte Benutzer Produkte und Kategorien erstellen, bearbeiten und löschen können. Ich erwarte, dass die Datenintegrität gewährleistet ist.

User Story 5: REST-Richtlinien und Benutzerfreundlichkeit

Als Entwickler möchte ich die Endpoints der Anwendung gemäss den REST-Richtlinien überarbeiten und sinnvoll benennen, um die Benutzerfreundlichkeit und Konsistenz der API zu verbessern.

User Story 6: Datenbankeffizienz und Skalierbarkeit

Als Systemarchitekt möchte ich die Datenbankanbindung überprüfen und gegebenenfalls eine alternative Datenbanktechnologie in Betracht ziehen, um die Effizienz und Skalierbarkeit der Anwendung sicherzustellen.

Zielgruppenanalyse

Eine Zielgruppenanalyse ist eine System tische Untersuchung und Bewertung der Merkmale, Bedürfnisse, Vorlieben und Verhaltensweisen einer spezifischen Gruppe von Menschen oder Organisationen, die als potenzielle Benutzer oder Kunden für ein Produkt.

System zur Verwaltung von Benutzern, Produkten und Produktkategorien.

1. Als Administrator:

Beschreibung: Als Administrator bin ich die Schlüsselperson für die Sicherheit und Konfiguration des Systems. Ich habe umfassende Rechte zur Verwaltung von Benutzern und zur Kontrolle der Produktdaten.

Bedürfnisse: Mein Hauptanliegen ist die einfache Verwaltung von Benutzern, Rechten und die Gewährleistung der Sicherheit im System. Ich benötige Zugang zum System-Dashboard, um Sicherheitsrichtlinien zu implementieren und die Datenbank und die Produktinformationen zu überwachen.

2. Als authentifizierter Benutzer:

Beschreibung: Als authentifizierter Benutzer möchte ich mich sicher und einfach anmelden, um Produkte anzusehen, Käufe zu tätigen und meine Kontoeinstellungen zu verwalten.

Bedürfnisse: Mein Fokus liegt auf einer einfachen und sicheren Anmeldemöglichkeit, dem Zugriff auf Produktinformationen, dem Kauf von Produkten und der Möglichkeit, meine Kontoeinstellungen unkompliziert zu ändern.

3. Als nicht authentifizierter Benutzer:

Beschreibung: Als nicht authentifizierter Benutzer bin ich ein Besucher der Anwendung, der sich nicht angemeldet hat. Ich möchte auf öffentliche Produktinformationen und Produktlisten zugreifen können.

Bedürfnisse: Ich benötige eine klare Benutzeroberfläche, um Produkte und Produktkategorien zu durchsuchen, ohne auf vertrauliche Informationen zuzugreifen. Die Anmeldung sollte mir verwehrt bleiben.

4. Als Datenbankadministrator:

Beschreibung: Als Datenbankadministrator bin ich für die Datenbankinitialisierung und -integrität verantwortlich. Ich stelle sicher, dass die Datenqualität und -konsistenz gewahrt bleiben.

Bedürfnisse: Ich benötige Werkzeuge und Prozesse, um Datenbankinitialisierung und -integrität zu überwachen und sicherzustellen.

Planmässige Umsetzung

Eine planmässige Umsetzung bezieht sich auf die systematische Durchführung eines Projekts.

Schritt 1: Problemraum verstehen

Ich werde zunächst eine umfassende Analyse des Problemraums durchführen, um ein tiefes Verständnis für die bestehenden Herausforderungen im System zu entwickeln. Dazu gehören Sicherheitslücken, Zugriffsrechte, Datenbankinitialisierung und nichtfunktionale Anforderungen. Ich werde auch die aktuelle Systemarchitektur dokumentieren.

Schritt 2: Zielsetzung und Priorisierung

Ich werde klare Ziele für die Problemlösung definieren und diese nach Dringlichkeit und potenziellen Risiken priorisieren. Dies wird mir helfen, die Schwerpunkte für die Lösungen festzulegen.

Schritt 3: Definition der Clare und Pretzien Klassen

In diesem Schritt werde ich das Fundament für die Pretize-Softwareanwendung in meinem IntelliJ-Projekt aufbauen. Dieser Schritt ist entscheidend, um in meiner Entwicklungsarbeit ein hohes Mass an Verständlichkeit und Effizienz zu gewährleisten. Ich werde alle Dateien und Ressourcen in passenden Ordnern organisieren, um in IntelliJ leicht navigieren zu können und sicherzustellen, dass meine Code Basis gut strukturiert ist.

Durch diese Vorgehensweise behalte ich immer den Überblick über meine Arbeit, optimiere die Organisation meines Codes und sorge dafür, dass mein Projekt eine saubere und übersichtliche Struktur hat.

Schritt 4: Definition klarer Logik und benutzerfreundlicher Endpunkt

In diesem Schritt ist es von höchster Wichtigkeit, dass ich alle meine Endpunkte klar und logisch definiere. Dies ermöglicht es mir, Postman effizient zu nutzen, insbesondere wenn es um das Erstellen, Löschen oder Aktualisieren von Ressourcen geht.

Die präzise Ausarbeitung meiner Endpunkte gewährleistet, dass ich schnell und zielsicher agieren kann, um die gewünschten Aufgaben in meiner Softwareanwendung zu erfüllen.

Schritt 5: Sorgfältige und präzise Überlegungen zur Rollenverteilung

In diesem Schritt ist es von höchster Wichtigkeit, dass ich äusserst aufmerksam bei der Verteilung von Rollen und Berechtigungen vorgehe. Ich habe klare Anforderungen, wer in der Lage sein sollte, welche Aktionen auszuführen. Es liegt an mir, sicherzustellen, dass sämtliche Endpunkte entsprechend den jeweiligen Rechten konfiguriert sind und dass ein reibungsloser Ablauf der Anwendung gewährleistet ist.

Die gründliche Überlegung und präzise Verwaltung der Rollen und Berechtigungen sind entscheidend, um sicherzustellen, dass die Anforderungen an die Zugriffskontrolle erfüllt werden und dass die Anwendung reibungslos und sicher betrieben werden kann.

Arbeitsjournal

ist ein schriftliches Dokument, das dazu dient, die täglichen und Aktivitäten mit Ereignissen festzuhalten.

Abbildung 1 Arbeitsjournal

Leitthema:	**Backend**
Arbeitstitel:	Backend für Applikationen
Name:	Daniel Bogdanov
Fachlehrperson:	Daniel Schmitz

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
16.10.2023	Montag	8H	6H und 30 Min

Alle Informationen sind auch auf GitHub: https://github.com/Dexxba/Modul_295

Tätigkeiten:

In meiner Planung für die Architektur von Intelidjei habe ich zunächst die Grundstruktur festgelegt, die es mir ermöglichen sollte, effizient innerhalb der Anwendung zu navigieren. Zu diesem Zweck habe ich drei Hauptklassen erstellt: Product, Category und User, jeweils mit zugehörigen Controllern, Services und Repositories.

Nachdem diese Klassen erstellt waren, war es notwendig, eine Logik zu implementieren, die die Beziehungen zwischen den Tabellen klar definierte. Dies ermöglichte es mir, schnell herauszufinden, welches Produkt sich in welcher Kategorie befindet, und wie diese Informationen in der Antwort auf GET-Anfragen an die API präsentiert werden sollen. Dies war entscheidend für die effiziente Navigation und die Darstellung der Daten in der Benutzeroberfläche

1. Erstellen von Produkt mit der passenden Logik – Gebrauchte Zeit: 2 H
2. Erstellen von Category mit der passenden Logik – Gebrauchte Zeit: 2 H
3. Erstellen von User mit der passenden Logik – Gebrauchte Zeit: 2 H
4. Erstellen der Methode um Produkt Liste zu Bekommen - Gebrauchte Zeit: 30 Min

Reflektion:

Mit Hilfe von Intelidjei habe ich die grundlegende Struktur der Anwendung definiert, indem ich die Klassen Product, Category und User erstellt habe, jeweils mit zugehörigen Controllern, Services und Repositories. Die klare Definition der Datenbeziehungen ermöglichte eine effiziente Navigation und die Darstellung von Daten in der Benutzeroberfläche, was die Benutzerfreundlichkeit verbesserte, und die Erfüllung der Nutzererwartungen unterstützte.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
17.10.2023	Dienstag	6H	3H 30 Min

Tätigkeiten:

Herstellung UserDto (Benutzerdatenübertragungsobjekts) und die Implementation seiner Logik. Ebenso werde ich auf die Einrichtung von POST-, PUT- und DELETE-Operationen für Benutzer eingehen. Des Weiteren werde ich die Verwendung der Annotation '@JsonProperty' zur Verschleierung des Passworts im UserDto behandeln, um sicherzustellen, dass das Frontend keinerlei Zugriff auf das Passwort hat

1. Erstellen von UserDto mit der passenden Logik – Gebrauchte Zeit: 1 H
2. Erstellen von Post, Put, Delete von User mit der passenden Logik – Gebrauchte Zeit: 2 H
3. Erstellen von @JsonPropery mit der passenden Logik – Gebrauchte Zeit: 30 Min

Reflektion:

Die Erstellung des UserDto und die Implementierung seiner Logik fühle ich mich sehr zufrieden mit den erzielten Ergebnissen. Dieses Projekt war eine Gelegenheit, mein Verständnis für die Konzepte der Datenübertragung und der Benutzerverwaltung zu vertiefen. Die Erklärungen zur Einrichtung der POST-, PUT- und DELETE-Operationen für Benutzer sowie zur Verwendung der '@JsonProperty'-Annotation für die Sicherheit des Passworts sind meiner Meinung nach klar und präzise.

Während des Schreibprozesses habe ich darauf geachtet, eine klare und verständliche Sprache zu verwenden, um sicherzustellen, dass meine Zielgruppe, andere Entwickler, die Informationen leicht nachvollziehen können.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
18.10.2023	Mittwoch	8H	4H 30 Min

Tätigkeiten:

Für die Implementierung der Passwortaktualisierungsfunktion habe ich eine umfassende Logik entwickelt, um sicherzustellen, dass Benutzer ihre Passwörter in Übereinstimmung mit den geltenden Sicherheitsrichtlinien und Datenschutzbestimmungen ändern können. Diese Logik gewährleistet die Integrität des Systems und schützt die vertraulichen Benutzerdaten. Darüber hinaus habe ich die Klassen CategoryDto und ProductDto erstellt, die sämtliche erforderlichen Methoden und Logik beinhalten, um die nahtlose Verwaltung von Produktkategorien und Produkten zu ermöglichen. Diese Klassen sind essenziell für die effiziente Handhabung und Speicherung von Daten. Um sicherzustellen, dass alle relevanten Aktivitäten im System verfolgt werden können, habe ich eine Textnachrichten-Funktion implementiert. Wenn beispielsweise eine neue Kategorie erstellt wird, wird automatisch eine Meldung wie 'Kategorie hinzugefügt' generiert. Dies ermöglicht eine transparente Überwachung von Systemänderungen und stellt sicher, dass wichtige Informationen erfasst werden.

1. Erstellen von CategoryDto mit der passenden Logik – Gebrauchte Zeit: 2 H
2. Erstellen von ProductDto mit der passenden Logik – Gebrauchte Zeit: 2 H
3. Erstellen von Implementierung der Passwortaktualisierungsfunktion – Gebrauchte Zeit: 30 Min

Reflektion:

Die Entwicklung der Passwortaktualisierungsfunktion, der Klassen CategoryDto und ProductDto sowie der Textnachrichten-Funktion war eine herausfordernde, aber äusserst lohnende Aufgabe. Die Implementierung dieser Funktionen hat nicht nur die Sicherheit und Integrität des Systems verbessert, sondern auch die Effizienz und Benutzerfreundlichkeit gesteigert.

Die Passwortaktualisierungsfunktion stellt sicher, dass Benutzer ihre Passwörter gemäss den Sicherheitsrichtlinien ändern können, was die Datensicherheit gewährleistet.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
19.10.2023	Donnerstag	2H	1H

Tätigkeiten:

Implementierung der Open API Swagger Dokumentation, auf: http://localhost:8080/swagger-ui/index.html#/

1. Implementierung der Open API Swagger Dokumentation - – Gebrauchte Zeit: 1 H

Reflektion:

Dank der Swagger Dokumentation habe ich gelernt, wie ich mein Code, Dokumentieren kann. Und Welche Punkte sichtbar sein müssen, damit man einen klaren überblick haben kann in Felder die nicht lehr sein dürfen.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
21.10.2023	Samstag	8H	8H

Tätigkeiten:

Implementierung zweier wichtiger Funktionen in meiner Anwendung. Zum einen habe ich den Crypt Password Encoder eingeführt, um die Passwortsicherheit zu erhöhen. Zum anderen habe ich die Authentifizierung konfiguriert, um den Zugriff auf autorisierte Benutzer zu beschränken. Ausserdem erkläre ich die Sicherheitseinstellungen und wie ich das LoginDto geändert habe, um den Admin-Status von Benutzern mit einem boolean-Wert anstelle eines Strings anzuzeigen. Diese Anpassungen tragen zur Sicherheit und Funktionalität meiner Anwendung bei.

1. Erstellen von Crypt Password Encoder – Gebrauchte Zeit: 2 H
2. Erstellen von AutchConfig – Gebrauchte Zeit: 1 H
3. Erstellen von update in Autch von der Security – Gebrauchte Zeit: 2 H
4. Erstellen von LoginDto und allen Autch Classen – Gebrauchte Zeit: 2 H
5. Erstellen von Admin von String auf boolean und alle Änderungen in code– Gebrauchte Zeit: 1H

Reflektion:

Der Crypt Password Encoder verbessert die Passwortsicherheit erheblich, während die Authentifizierungskonfiguration den Zugriff auf autorisierte Benutzer beschränkt. Zudem habe ich die Sicherheitseinstellungen dargestellt und erläutert, wie die Umstellung des Admin-Status auf einen boolean-Wert die Effizienz und Sicherheit meiner Anwendung steigert. Diese Anpassungen tragen zur Gesamtsicherheit und Funktionalität meiner Anwendung bei.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
22.10.2023	Sonntag	6H	6H

Tätigkeiten:

Ich habe die JWT-Token-Implementierung abgeschlossen, um sicherzustellen, dass meine Anwendung die Authentizität der Benutzer überprüft. Ein JWT besteht aus einem Header, Payload und einer Signatur. Das Token wird für den Zugriff auf geschützte Endpunkte benötigt, wobei die Endpunkte je nach Benutzerrolle unterschiedliche Berechtigungen haben.

Mit der JWT-Implementierung sind bestimmte Endpunkte geschützt und erfordern ein JWT-Token, um auf sie zuzugreifen. Dies gilt insbesondere für Ressourcen, die nur für authentifizierte Benutzer zugänglich sind.

Nach der JWT-Implementierung haben wir festgelegt, welche Benutzer auf welche Endpunkte zugreifen dürfen. Normale Benutzer haben eingeschränkten Zugriff, während Administratoren erweiterte Berechtigungen haben.

Ich habe die Swagger-Dokumentation aktualisiert, um klarzustellen, welche Endpunkte ohne die Notwendigkeit eines JWT-Tokens zugänglich sind. Dies erleichtert es Benutzern und Entwicklern, die API zu verstehen und die Funktionen zu erkennen, die keine Authentifizierung erfordern.

1. Erstellen von JWT-Token-Implementierung – Gebrauchte Zeit: 2 H
2. Erstellen von bestimmte Endpunkte – Gebrauchte Zeit: 2 H
3. Erstellen von Benutzer auf welche Endpunkte zugreifen dürfen – Gebrauchte Zeit: 1 H
4. Erstellen von Swagger-Dokumentation aktualisiert – Gebrauchte Zeit: 1 H

Reflektion:

Die Implementierung von JWT-Token und Autorisierungslogik ist für die Sicherheit und den Zugriff in meiner Anwendung von entscheidender Bedeutung. Es ermöglicht mir, Benutzer sicher zu authentifizieren und ihre Berechtigungen zu verwalten, um den Schutz unserer Daten sicherzustellen. Ich habe die Dokumentation aktualisiert.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
23.10.2023	Montag	8H	10H

Tätigkeiten:

In diesem Stadium des Projekts, das sich der Übergabe nähert, ist es von entscheidender Bedeutung, den Code angemessen zu kommentieren und sicherzustellen, dass andere Entwickler ihn leicht verstehen können. Dies ist die erste Aufgabe des heutigen Tages.

Um sicherzustellen, dass der Code klar und verständlich ist, habe ich umfassende Kommentare hinzugefügt. Diese Kommentare erklären den Zweck und die Funktionalität verschiedener Teile des Codes, um anderen Entwicklern die Arbeit zu erleichtern und die Wartbarkeit des Codes sicherzustellen.

Im Anschluss an die Codekommentierung habe ich mich der Überarbeitung meiner Dokumentation gewidmet. Während des Projekts hatte ich umfangreiche Notizen erstellt, die jetzt in einen standardisierten Zustand gebracht wurden. Dies stellt sicher, dass die Dokumentation umfassend, konsistent und leicht verständlich ist.

Die Sorgfalt, die auf die Kommentierung des Codes und die Überarbeitung der Dokumentation gelegt wurde, ist von entscheidender Bedeutung, um sicherzustellen, dass das Projekt für die Übergabe bereit ist und von anderen Entwicklern problemlos genutzt und gewartet werden kann. Dies sind wichtige Schritte, um die langfristige Nachhaltigkeit des Projekts zu gewährleisten.

1. Den Code angemessen zu kommentieren – Gebrauchte Zeit: 2 H
2. Erstellen von Dokumentation – Gebrauchte Zeit: 8 H

Reflektion:

Die heutigen Aufgaben bestanden darin, meinen Code zu kommentieren und die Projektdokumentation zu überarbeiten. Die Codekommentierung war entscheidend, um sicherzustellen, dass andere Entwickler den Code ohne Schwierigkeiten verstehen und warten können. Durch die Dokumentationsüberarbeitung konnten meine umfangreichen Notizen in eine benutzerfreundliche und konsistente Form gebracht werden.

Diese Schritte sind von entscheidender Bedeutung, da sie die Lesbarkeit und die langfristige Wartbarkeit des Codes sicherstellen. Die Übergabe des Projekts wird somit reibungsloser verlaufen, und andere Entwickler können effizient damit arbeiten. Dies reflektiert die professionelle Herangehensweise an die Softwareentwicklung und die Verantwortung für die Qualität unserer Arbeit.

Datum:	Tag:	Geplante Zeit:	Gebrauchte Zeit:
28.10.2023	Samstag	4H	6H

Tätigkeiten:

Ein zentraler Aspekt, der für die Fertigstellung meines Projekts von entscheidender Bedeutung war, ist die Einrichtung einer Datenbank in IntelliJ, die automatisch beim Starten der Anwendung erstellt wird und voreingestellte Werte einführt. Heute ist der Tag, an dem ich mein Projekt abschliesse und alle erforderlichen Schritte für die Einreichung vorbereite. Neben einer Aktualisierung der Dokumentation und einer abschliessenden Überarbeitung benötige ich noch einen letzten Feinschliff, um sicherzustellen, dass alles den Anforderungen meines Dozenten entspricht.

Die Datenbank, die ich implementiert habe, erstellt automatisch einen Administratorbenutzer, der die erforderlichen Berechtigungen besitzt, um die Daten unmittelbar zu verwalten. Diese Implementierung wurde mithilfe des Frameworks 'FlayWay' durchgeführt, wodurch das Projekt nun vollständig für die Einreichung vorbereitet ist."

1. Implementierung wurde mithilfe des Frameworks 'FlayWay' – Gebrauchte Zeit: 3 H
2. Aktualisierung der Dokumentation – Gebrauchte Zeit: 3 H

Reflektion:

In der finalen Phase meines Projekts fühle ich, dass ich auf eine intensive und lehrreiche Reise zurückblicken kann. Die Herausforderungen, die auf dem Weg lagen, haben mich nicht nur fachlich, sondern auch persönlich wachsen lassen. Die Implementierung der automatischen Datenbankerstellung in IntelliJ war ein entscheidender Schritt, der es ermöglichte, die Anwendung reibungslos zu starten und vordefinierte Datenwerte hinzuzufügen.

Die Aktualisierung der Dokumentation und die finale Überarbeitung des Projekts sind wichtige Schritte, um sicherzustellen, dass alle Anforderungen meines Dozenten erfüllt sind. Diese Phase erfordert sorgfältige Aufmerksamkeit, um sicherzustellen, dass jedes Detail und jeder Aspekt des Projekts auf höchstem Niveau ist.

Die Implementierung der Datenbank mit einem Administratorbenutzer, der die Berechtigung zur Datenverwaltung hat, war ein Schlüsselerfolg, und dies wurde durch die Verwendung des Frameworks 'FlayWay' effizient erreicht. Ich bin stolz auf die Fortschritte, die ich in diesem Projekt gemacht habe, und freue mich darauf, es erfolgreich abzuschliessen und meinem Dozenten präsentieren zu können.

Obwohl mein Dozent grosszügigerweise zusätzliche Zeit zur Fertigstellung des Projekts angeboten hat, habe ich mich entschieden, mich strikt an die ursprünglich vereinbarte Deadline zu halten. Ich glaube fest daran, dass die Einhaltung der ursprünglichen Vereinbarung ein wichtiger Schritt in der Professionalisierung und Disziplin im Projektmanagement ist.

Durch diese Entscheidung konnte ich das Projekt innerhalb des gesetzten Zeitrahmens fertigstellen und bin stolz darauf, dieses Engagement für die vereinbarte Frist einzuhalten. Diese Erfahrung hat mich darin bestärkt, wie wichtig es ist, sich an selbst gesetzte Ziele zu halten und die eigenen Fähigkeiten zur effizienten Zeiteinteilung und Aufgabenbewältigung zu verbessern.

Namensschema für Daten

Um ein Namensschema für deine Daten in IntelliJ zu erstellen, benötigen wir die Namen für die Entitäten.

Modul295Application:

Dies ist die Hauptklasse für deine Modul295-Anwendung.

users:

Entity-Klasse (Users): Repräsentiert Benutzerdaten in der Datenbank.

Data Transfer Object (DTO) (UserDto): Wird verwendet, um Benutzerdaten zwischen Schichten der Anwendung zu übertragen.

Controller (UsersController): Nimmt HTTP-Anfragen für Benutzerentitäten entgegen und leitet sie an den Service weiter.

Service (UsersService): Enthält die Geschäftslogik und Verarbeitung von Benutzerdaten.

Repository (UsersRepository): Stellt Methoden bereit, um auf die Benutzerdatenbank zuzugreifen.

products:

Entity-Klasse (Product): Repräsentiert Produktinformationen in der Datenbank.

Data Transfer Object (DTO) (ProductDto): Wird verwendet, um Produktinformationen zwischen Schichten der Anwendung zu übertragen.

Controller (ProductController): Nimmt HTTP-Anfragen für Produktentitäten entgegen und leitet sie an den Service weiter.

Service (ProductService): Enthält die Geschäftslogik und Verarbeitung von Produktinformationen.

Repository (ProductRepository): Stellt Methoden bereit, um auf die Produkt-Datenbank zuzugreifen.

category:

Entity-Klasse (Category): Repräsentiert Kategoriedaten in der Datenbank.

Data Transfer Object (DTO) (CategoryDto): Wird verwendet, um Kategoriedaten zwischen Schichten der Anwendung zu übertragen.

Controller (CategoryController): Nimmt HTTP-Anfragen für Kategorieentitäten entgegen und leitet sie an den Service weiter.

Service (CategoryService): Enthält die Geschäftslogik und Verarbeitung von Kategoriedaten.

Repository (CategoryRepository): Stellt Methoden bereit, um auf die Kategoriedatenbank zuzugreifen.

login:

Data Transfer Object (DTO) (LoginDto): Wird verwendet, um Anmeldeinformationen (Benutzername und Passwort) zwischen Schichten der Anwendung zu übertragen.

jwt (JSON Web Token):

Filter (JwtFilter): Verarbeitet und überprüft JWTs in eingehenden HTTP-Anfragen, um die Authentifizierung und Autorisierung zu ermöglichen.

Service (JwtService): Bietet Funktionen zum Erstellen und Verifizieren von JWTs.

auth (Authentifizierung und Autorisierung):

Konfiguration (AuthConfig): Enthält die Konfigurationseinstellungen für die Authentifizierung und Autorisierung der Anwendung.

Controller (AuthController): Nimmt HTTP-Anfragen für Authentifizierung und Autorisierung entgegen und leitet sie an den Service weiter.

Service (AuthService): Enthält die Geschäftslogik für die Authentifizierung und Autorisierung von Benutzern in der Anwendung.

Beschreibung der Endpoints

Hier sind die Endpunkte mit Pfad, Parametern, Zweck und HTTP-Anfragemethode.

UsersController:

Endpunkt: Neuen Benutzer erstellen

Pfad: POST /http://localhost:8080/users

Parameter:

userDto - Benutzerdaten als UserDto-Objekt im Anfragekörper (Request Body)

Zweck: Erstellt einen neuen Benutzer.

HTTP-Anfragemethode: POST

Antwort: ResponseEntity mit einer Erfolgs- oder Fehlermeldung. Bei Erfolg wird der Status CREATED zurückgegeben, andernfalls BAD REQUEST.

Endpunkt: Vorhandenen Benutzer aktualisieren

Pfad: PUT /http://localhost:8080/users/{id}

Parameter:

id - Die ID des zu aktualisierenden Benutzers im Pfad

userDto - Aktualisierte Benutzerdaten als UserDto-Objekt im Anfragekörper (Request Body)

Zweck: Aktualisiert einen vorhandenen Benutzer anhand seiner ID.

HTTP-Anfragemethode: PUT

Antwort: ResponseEntity mit einer Erfolgs- oder Fehlermeldung. Bei Erfolg wird der Status OK zurückgegeben, andernfalls NOT FOUND.

Endpunkt: Benutzer anhand der ID löschen

Pfad: DELETE /http://localhost:8080/users/{id}

Parameter:

id - Die ID des zu löschenden Benutzers im Pfad

Zweck: Löscht einen Benutzer anhand seiner ID.

HTTP-Anfragemethode: DELETE

Antwort: ResponseEntity mit einer Erfolgsmeldung und dem Status OK.

Endpunkt: Benutzerinformationen anhand der ID abrufen

Pfad: GET /http://localhost:8080/users/{id}

Parameter:

id - Die ID des abzurufenden Benutzers im Pfad

Zweck: Ruft Benutzerinformationen anhand der ID ab.

HTTP-Anfragemethode: GET

Antwort: ResponseEntity, dass entweder die Benutzerinformationen oder eine Fehlermeldung enthält. Bei Erfolg werden die Benutzerinformationen zurückgegeben, andernfalls NOT FOUND mit einer entsprechenden Fehlermeldung.

ProductController:

Endpunkt: Neues Produkt erstellen

Pfad: POST http://localhost:8080/product

Parameter:

productDto - Produktdaten als ProductDto-Objekt im Anfragekörper (Request Body)

Zweck: Erstellt ein neues Produkt.

HTTP-Anfragemethode: POST

Antwort: ResponseEntity mit einer Erfolgs- oder Fehlermeldung. Bei Erfolg wird der Status CREATED zurückgegeben, andernfalls BAD REQUEST.

Endpunkt: Vorhandenes Produkt aktualisieren

Pfad: PUT : http:/localhost:8080/product/{id}

Parameter:

id - Die ID des zu aktualisierenden Produkts im Pfad

productDto - Aktualisierte Produktdaten als ProductDto-Objekt im Anfragekörper (Request Body)

Zweck: Aktualisiert ein vorhandenes Produkt anhand seiner ID.

HTTP-Anfragemethode: PUT

Antwort: ResponseEntity mit dem aktualisierten Produkt oder einer Fehlermeldung. Bei Erfolg wird der Status OK zurückgegeben, andernfalls NOT FOUND.

Endpunkt: Produkt anhand der ID löschen

Pfad: DELETE http:/localhost:8080/product/{id}

Parameter:

id - Die ID des zu löschenden Produkts im Pfad

Zweck: Löscht ein Produkt anhand seiner ID.

HTTP-Anfragemethode: DELETE

Antwort: ResponseEntity mit einer Erfolgsmeldung und dem Status OK.

Endpunkt: Produktinformationen anhand der ID abrufen

Pfad: GET http:/localhost:8080/product/{id}

Parameter:

id - Die ID des abzurufenden Produkts im Pfad

Zweck: Ruft Produktinformationen anhand der ID ab.

HTTP-Anfragemethode: GET

Antwort: ResponseEntity, dass entweder die Produktinformationen oder eine Fehlermeldung enthält. Bei Erfolg werden die Produktinformationen zurückgegeben, andernfalls NOT FOUND mit einer entsprechenden Fehlermeldung.

Endpunkt: Liste aller Produkte abrufen

Pfad: GET http:/localhost:8080/product/list

Parameter: Keine

Zweck: Ruft eine Liste aller Produkte ab.

HTTP-Anfragemethode: GET

Antwort: Eine Liste von Produktobjekten.

Endpunkt: Neue Produktkategorie erstellen

Pfad: POST http://localhost:8080/category

Parameter:

categoryDto - Kategoriedaten als CategoryDto-Objekt im Anfragekörper (Request Body)

Zweck: Erstellt eine neue Produktkategorie.

HTTP-Anfragemethode: POST

Antwort: ResponseEntity mit einer Erfolgs- oder Fehlermeldung. Bei Erfolg wird der Status CREATED zurückgegeben, andernfalls BAD REQUEST.

Endpunkt: Vorhandene Produktkategorie aktualisieren

Pfad: PUT http://localhost:8080/category /{id}

Parameter:

id - Die eindeutige Kennung (ID) der zu aktualisierenden Kategorie im Pfad

categoryDto - Aktualisierte Kategoriedaten als CategoryDto-Objekt im Anfragekörper (Request Body)

Zweck: Aktualisiert eine vorhandene Produktkategorie anhand ihrer ID.

HTTP-Anfragemethode: PUT

Antwort: ResponseEntity mit den aktualisierten Kategoriedaten oder einer Fehlermeldung. Bei Erfolg wird der Status OK zurückgegeben, andernfalls NOT FOUND.

Endpunkt: Produktkategorie anhand der ID löschen

Pfad: DELETE http://localhost:8080/category /{id}

Parameter:

id - Die eindeutige Kennung (ID) der zu löschenden Kategorie im Pfad

Zweck: Löscht eine Produktkategorie anhand ihrer ID.

HTTP-Anfragemethode: DELETE

Antwort: ResponseEntity mit einer Erfolgsmeldung und dem Status OK.

Endpunkt: Produktkategorieinformationen anhand der ID abrufen

Pfad: GET http://localhost:8080/category /{id}

Parameter:

id - Die eindeutige Kennung (ID) der abzurufenden Kategorie im Pfad

Zweck: Ruft Informationen zur Produktkategorie anhand ihrer ID ab.

HTTP-Anfragemethode: GET

Antwort: ResponseEntity, das entweder die Kategoriedaten oder eine Fehlermeldung enthält. Bei Erfolg werden die Kategoriedaten zurückgegeben, andernfalls NOT FOUND mit einer entsprechenden Fehlermeldung.

Endpunkt: Benutzeranmeldung

Pfad: POST http://localhost:8080/auth/login

Parameter:

loginDto - Login-Informationen des Benutzers als LoginDto-Objekt im Anfragekörper (Request Body)

Zweck: Behandelt HTTP POST-Anfragen für die Benutzeranmeldung und stellt einen JWT-Token zur Verfügung, wenn die Anmeldung erfolgreich ist.

HTTP-Anfragemethode: POST

Antwort: ResponseEntity mit dem HTTP-Status 200 OK und dem JWT-Token im Antwortkörper. Dieses Token kann verwendet werden, um den authentifizierten Benutzer zu identifizieren.

Die shouldNotFilter-Methode in Ihrem JwtFilter ist dafür verantwortlich, zu entscheiden, ob der Filter auf eine bestimmte Anfrage angewendet werden sollte oder nicht. In diesem Fall gibt die Methode true zurück, wenn der Filter nicht auf die Anfrage angewendet werden sollte, andernfalls gibt sie false zurück.

Hier sind die URI-Muster, für die der Filter nicht angewendet wird:

/users: Dies ist der Endpunkt für die Benutzerregistrierung. Der Filter wird nicht auf Anfragen an diesen Endpunkt angewendet, da die Benutzerregistrierung authentifiziert erfolgt.

/auth/login: Dies ist der Endpunkt für die Benutzerauthentifizierung. Der Filter wird auch nicht auf Anfragen an diesen Endpunkt angewendet, da die Authentifizierung authentifiziert erfolgt.

/swagger-ui: Dieser Pfad ist mit der Swagger-Benutzeroberfläche für die API-Dokumentation verknüpft. Der Filter wird nicht auf Anfragen an die Swagger-Benutzeroberfläche angewendet, um den Zugriff auf die Dokumentation zu ermöglichen, ohne eine Authentifizierung zu erfordern.

/v3: Dies ist eine Version der API, und der Filter wird nicht auf Anfragen an diese Version angewendet.

Für alle anderen Anfragen, die nicht zu den oben genannten URI-Mustern gehören, wird der Filter angewendet, um die Authentifizierung mithilfe von JWT-Token durchzuführen. Dies stellt sicher, dass nur für bestimmte Endpunkte (wie die Benutzerregistrierung und die Authentifizierung) keine Authentifizierung erforderlich ist, während für andere geschützte Ressourcen eine Authentifizierung und Autorisierung erforderlich ist.

Der SecurityFilterChain in Code ist für die Konfiguration der Sicherheitsfilterkette in meiner Anwendung verantwortlich. Er definiert, welche Anfragen authentifiziert werden müssen und welche ohne Authentifizierung zugelassen sind.

addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class): Hier wird der jwtFilter vor dem UsernamePasswordAuthenticationFilter hinzugefügt, um sicherzustellen, dass JWT-Authentifizierung vor der Benutzername-Passwort-Authentifizierung erfolgt.

authorizeHttpRequests(authorizationManagerRequestMatcherRegistry -> {...}: Hier erfolgt die Konfiguration der Zugriffsberechtigungen für verschiedene Endpunkte und HTTP-Methoden.

Mit permitAll() wird der Zugriff ohne Authentifizierung erlaubt. Dies wird für Endpunkte wie die Benutzerregistrierung, die Benutzerauthentifizierung und bestimmte Swagger-Endpunkte festgelegt.

hasAuthority("admin"): Hier wird festgelegt, dass für bestimmte HTTP-Methoden und Endpunkte die Autorisierung (in diesem Fall die Rolle "admin") erforderlich ist. Dies bedeutet, dass nur Benutzer mit der "admin"-Rolle auf diese Endpunkte zugreifen können.

Die Konfiguration stellt sicher, dass nicht authentifizierte Benutzer auf einige öffentliche Endpunkte zugreifen können, während für andere Endpunkte (z. B. Endpunkte zur Verwaltung von Kategorien und Produkten) eine "admin"-Rolle erforderlich ist.

Dieser SecurityFilterChain stellt sicher, dass die Sicherheit in meiner Anwendung gut konfiguriert ist und bestimmte Ressourcen vor unautorisiertem Zugriff schützt, während gleichzeitig öffentliche Endpunkte verfügbar bleiben.

```
authorizationManagerRequestMatcherRegistry
                // Permit access without authentication for these endpoints
                .requestMatchers(HttpMethod.POST, "/users").permitAll()
                .requestMatchers(HttpMethod.POST, "/auth/login").permitAll()
                .requestMatchers("/v3/api-docs").permitAll()
                .requestMatchers("/swagger-ui/**").permitAll()
                .requestMatchers("/v3/api-docs/swagger-config").permitAll();
        authorizationManagerRequestMatcherRegistry
                // Define required authorities for specific HTTP methods and endpoints
                .requestMatchers(HttpMethod.DELETE, "/users/*").hasAuthority("admin")
                .requestMatchers(HttpMethod.POST, "/category").hasAuthority("admin")
                .requestMatchers(HttpMethod.PUT, "/category/*").hasAuthority("admin")
                .requestMatchers(HttpMethod.DELETE, "/category/*").hasAuthority("admin")
                .requestMatchers(HttpMethod.POST, "/product").hasAuthority("admin")
                .requestMatchers(HttpMethod.PUT, "/product/*").hasAuthority("admin")
                .requestMatchers(HttpMethod.DELETE, "/product/*").hasAuthority("admin");
```

Fazit

Das letzte Wort Über das Modul

Das Modul UK295 wurde erfolgreich abgeschlossen, und es bereitete mir ausserordentliches Vergnügen. Die Programmierung, die Implementierung von Logik und die Herstellung der Kommunikation waren äusserst faszinierend und haben mich in vollem Masse begeistert. Ich habe mit grosser Hingabe an diesem Projekt gearbeitet und dabei eine beträchtliche Menge an Wissen erworben.

Während der Projektarbeit konnte ich meine Programmierfähigkeiten erheblich steigern. Jeder Aspekt dieses Projekts war äusserst ansprechend, und es gab keinen einzigen Bereich, in dem ich nicht vollkommen aufgegangen bin. Wir sahen uns enormen Herausforderungen gegenüber, die mitunter äusserst anspruchsvoll waren. Dennoch haben sie meine Fähigkeiten erheblich erweitert. Dieses Projekt schärfte meine logische Denkweise und half mir dabei, meine eigenen Grenzen weiter zu verschieben.

Nicht nur das, es erweiterte auch meinen Horizont in Bezug auf Wissen. Ich erhielt wertvolle Einblicke in die neuesten Technologien und Trends in unserem Fachbereich, was meinen Blick auf die IT-Landschaft wesentlich erweiterte. Diese Erfahrung half mir nicht nur in meiner aktuellen Aufgabe, sondern wird sich auch langfristig positiv auf meine berufliche Entwicklung auswirken.

Zusammenfassend kann ich sagen, dass die Arbeit an Modul UK295 nicht nur äusserst spannend und befriedigend war, sondern auch eine bedeutende Stufe in meiner beruflichen Entwicklung markiert. Ich freue mich darauf, das erworbene Wissen und die erworbenen Fähigkeiten in zukünftigen Projekten einzusetzen und weiterhin in meiner Karriere voranzukommen.

Obwohl mein Dozent grosszügigerweise zusätzliche Zeit zur Fertigstellung des Projekts angeboten hat, habe ich mich entschieden, mich strikt an die ursprünglich vereinbarte Deadline zu halten. Ich glaube fest daran, dass die Einhaltung der ursprünglichen Vereinbarung ein wichtiger Schritt in der Professionalisierung und Disziplin im Projektmanagement ist.

Durch diese Entscheidung konnte ich das Projekt innerhalb des gesetzten Zeitrahmens fertigstellen und bin stolz darauf, dieses Engagement für die vereinbarte Frist einzuhalten. Diese Erfahrung hat mich darin bestärkt, wie wichtig es ist, sich an selbst gesetzte Ziele zu halten und die eigenen Fähigkeiten zur effizienten Zeiteinteilung und Aufgabenbewältigung zu verbessern!

Sehr geehrter Herr Stefano Mavilio,

Ich möchte mich herzlich bei Ihnen für die äusserst lehrreiche Zeit, Ihren herausragenden Unterricht und die ausserordentlich fruchtbare Zusammenarbeit bedanken. Die Erfahrungen, die ich während unserer Zusammenarbeit gesammelt habe, waren von unschätzbarem Wert und haben meine Kenntnisse erheblich erweitert. Ich bin zutiefst dankbar für Ihr Engagement und Ihre Unterstützung.

Ich hoffe aufrichtig, dass sich in der Zukunft erneut Gelegenheiten ergeben, bei denen wir zusammenarbeiten können. Mit freundlichen Grüssen,

Bogdanov Daniel

Abbildung der Tabellen

Quellen

GitHub:

https://github.com/erostefano/csbe-demo/commits/main

GitHub:

https://github.com/Jl115/spring-backend-uek295/commits/master

Translator:

https://www.deepl.com/translator

Base64 Decode and Encode:

https://www.base64encode.org

JSON Web Tokens:

https://jwt.io

HTTP response status codes:

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful_responses

Getting Spring Security

https://docs.spring.io/spring-security/reference/getting-spring-security.html

Documenting a Spring REST:

https://springdoc.org

Eine Starke Ai:

https://chat.openai.com/?model=text-davinci-002-render-sha

OpenAPI definition:

http://localhost:8080/swagger-ui/index.html#/

Database Migrations with Flyway:

https://www.baeldung.com/database-migrations-with-flyway

Set-up Flyway with Spring Boot:

https://medium.com/hprog99/set-up-flyway-with-spring-boot-1b24b8abe56e