m29987

Programmieren 3 - Hausarbeit 2023

Dokumentation

Aufgabe:

Die Aufgabe war es einen REST Service für Informatik Fachliteratur in Java zu erstellen. Der Service soll JSON basiert sein.

Server: Soll mit Anfragen von mehreren Klienten umgehen können. Er besitzt eine Verbindung zu einer Datenbank die die Daten zur Fachliteratur enthält. Der Server nimmt Anfragen entgegen und kommuniziert entsprechend mit der Datenbank.

Klienten: Sollen mit JavaFX implementiert werden. Sie können Anfragen an den Server senden und die erhaltenden Daten in POJOs umwandeln und darstellen.

Es kann einen Hauptbenutzer geben, der zusätzlich Daten in der Datenbank erstellen / bearbeiten / löschen kann.

Starten des Servers sowie der Datenbank:

Um die Anwendung benutzen zu können muss zuvor mit einem Programm wie XAMPP eine Datenbank dem Programm zur Verfügung gestellt werden. Es wird das Vorgehen mit XAMPP beschrieben:

1. XAMPP Launcher starten, MySQL und Apache starten.
2. phpMyAdmin starten.
3. Unter „Datenbanken“ eine neue Datenbank mit Name: „informatik“ erstellen.
4. Auf Benutzerkonten wechseln und neuen Benutzer anlegen.
5. Name: „minf“ und Passwort „prog3“ eingeben, sowie alle Rechte verteilen.
6. Unter „Datenbank“ die Datenbank „informatik“ wählen und alle Rechte vergeben
7. phpMyAdmin kann nun geschlossen werden.

Damit die Datenbank verwendet werden kann, muss sie weitergehend auf Port 3306 durch XAMPP (MySQL) laufen gelassen werden.

Starten des Servers: Der Server kann über Gradle gestartet werden. Beim ersten Mal starten wird die Datenbank automatisch gefüllt.

Soll der Server nicht nur auf dem lokalen System laufen, kann unter „ServerResources.properties“ von „localhost“ auf eine IP Adresse gewechselt werden.

Starten des Klienten:

Die Klienten können über Gradle gestartet werden.

Der Klient startet immer im Login Bildschirm. Von dort kann er als normaler Benutzer oder als Hauptbenutzer das Programm betreten.

Der Benutzerstatus kann bei Bedarf auch gewechselt werden, ohne wichtige Benutzereinstellungen im Programm zu verlieren.

Wird der Klient ohne den Server gestartet, enthält er auch keine Daten. Außerdem kann man sich nicht als Hauptbenutzer anmelden.

Wenn der Server gestartet wurde, kann man sich als Hauptbenutzer einloggen. Nur ein Klient kann jeweils Hauptbenutzer werden. Dieser Klient muss sich erst wieder abmelden oder das Fenster schließen, bevor der Hauptbenutzer wieder freigegeben wird.

Die Voreinstellung des Host Namens im Klienten kann vor dem Login auch vom Benutzer verändert werden.

Normaler Benutzer:

Der Normale Benutzer kann die Bücher in der Datenbank anfragen und darstellen.

Er erhält eine Liste der Bücher und kann per Doppelklick mehr Informationen zu einem Buch in einer neuen Ansicht ansehen.

Die Listengröße (Seitengröße) kann vom Benutzer angepasst werden. So können mehr oder weniger Bücher auf einer Seite angezeigt werden. Der Benutzer kann durch die Seiten wechseln.

Der Benutzer kann die Sortier-Richtung einstellen. Es wird nach Titel sortiert.

Der Benutzer kann die Bücher nach Kategorien filtern. Ebenso kann er auch eine Zeichenfolge eingeben. Titel, die die Zeichenfolge enthalten werden markiert und gefiltert.

Filter werden erst nach Bestätigung durch den Filter-Knopf angewandt.

Bestehende Filter können durch den Zurücksetzen-Knopf zurückgesetzt werden.

Mit dem Return-Knopf kann der Benutzer zurück ins Login Menü.

Hauptbenutzer:

Der Hauptbenutzer kann alles, was der normale Benutzer kann.

Der Hauptbenutzer gibt beim Log Out den Hauptbenutzer für andere Klienten wieder frei.

Der Hauptbenutzer hat einen Edit und Delete Knopf neben jedem Buch in der Liste. Damit kann ein Buch entweder editiert oder aus der Datenbank gelöscht werden.

In der unteren Reihe gibt es 5 Knöpfe für den Hauptbenutzer.

Der grün markierte Knopf zeigt die aktuelle Ansicht an.

Es können sich Bücher, Kategorien, Autoren und Verlage angesehen werden.

Die Ansicht kann durch das Klicken auf die blauen Knöpfe geändert werden.

Nur in der Buch Ansicht kann die Auswahl „Kategorie“ als Filter verwendet werden, alle anderen besitzen nur die Texteingabe.

Wird in der aktuellen Ansicht der grüne Knopf gedrückt, betritt der Benutzer den jeweiligen Kreieren-Modus.

Es besteht ebenfalls die Möglichkeit durch den roten Knopf die Datenbank in ihren Ursprungszustand zurückzubringen.

Filter, Seitenansicht (Größe, aktuelle Seite) und Ordnung bleiben in den einzelnen Modi für sich erhalten, auch wenn die Ansicht gewechselt wird.

Edit Modus:

Mit dem Editieren Knopf neben einem Eintrag kommt der Hauptbenutzer in eine neue Ansicht, in der er veränderbare Werte editieren kann.

Editieren von Werten: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog den neuen Wert eingeben.

Editieren vom Verlag: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog einen neuen Verlag aus einer Auswahl auswählen.

Editieren von Autoren und Kategorien: Durch das Klicken auf den Editieren Knopf kann der Benutzer in einem Dialog Elemente zwischen den ausgewählten Elementen (linke Liste) und allen nicht ausgewählten Elementen (rechte Liste) wechseln. Bei Bestätigung werden die ausgewählten Elemente in der linken Liste übernommen.

Die Veränderungen können nicht Bestätigt werden, solange ungültige Werte vorhanden sind, ansonsten werden sie durch ein Klicken auf den Submit-Button in die Datenbank übernommen.

Im Editieren Modus kann sich ebenfalls entschieden werden den Eintrag zu löschen.

Verlassen werden kann der Editieren Modus durch den Exit Button oben links. Nicht gespeicherte Änderungen gehen dabei verloren.

Create Modus:

Wenn der Benutzer in den Kreieren-Modus wechselt, erhält er ein leeres Objekt, von dem er alle Werte einstellen kann.

Jedes Feld muss in der Regel einen gültigen Wert vom Benutzer erhalten.

Wenn ein Buch erstellt wird muss mindestens ein Autor und eine Kategorie verteilt werden.

Die initiale Bewertung kann per Klick auf die Sterne eingestellt werden.

Beim Erstellen eines Autors kann der Alias freigelassen werden, dann wird dieser aus Vor und Nachname erstellt.

Vorgehen / Design-Entscheidungen:

Projektstruktur:

Das Projekt besteht aus den Java Modulen „Client“, „Server“ und „Common“. Hauptsächlich wurde das aus Softwaretechnik bekannte und dort bereits in einer Übung angewandte MVC Modell angewandt.

Dies bedeutet, dass alle Teile des Programms (vor allem im Klient) in Model, View und Controller aufgeteilt wurde.

Datenbank:

Die Datenbank wird beim ersten Start des Servers initialisiert.

Es gibt Bücher mit allen wichtigen Attributen.

Zusätzlich zu den Büchern gibt es noch Autoren, Verlage und Kategorien, mit ihren jeweiligen Attributen.

Jedes Buch wird von nur einem Verlag veröffentlicht (Publisher_id wird im jeweiligen Buch als Foreign Key gespeichert)

Jedes Buch kann mehrere Kategorien und Autoren haben (Tabellen book_authors und book_categories).

Primary Key Buch: ISBN

Primary Key Autor, Publisher, Category: id (auto_increment für einfachen Insert von weiteren Einträgen)

Die Datenbank wird im Programm über einen MariaDB Treiber geladen. Sollte eine Verbindung nicht zustande kommen, so läuft der Server trotzdem weiter, liefert aber keine Daten. Sollte eine Verbindung abbrechen, so muss der Server neu gestartet werden (die Klienten müssen nicht neu gestartet werden).

Die Datenbank ist als Singleton implementiert, um konsistenten Zugriff auf die Datenbank zu gewährleisten. Dies wurde deshalb so implementiert, weil es in Softwaretechnik einen Vergleichbaren Übungsaufbau gab, in dem dies so verwendet wurde.

Informationen/ Parameter zu der Datenbank sind im „resources“ Ordner des Server Moduls zu finden und einstellbar.

Common:

Das Modul Common enthält die Schnittstelle für die Kommunikation von Klient und Server.

Unter „resources“ sind in „ServerResources.properties“ alle Informationen über den Server vorhanden. Sie enthält die Adresse und Port des Hosts, sowie die einzelnen Ressourcenpfade des Servers. Bei einer Umstrukturierung der Ressourcen können also hier die Pfade geändert werden.

Ebenso kann durch das ändern von „Host.Adress“ der Server für unterschiedliche Rechner im Netzwerk erreichbar gemacht werden.

Im „main“ Ordner sind alle Klassen (POJOs) vorhanden, die als Schnittstelle für den Transfer von Daten via JSON notwendig sind.

Der Server nutzt sie, um konkrete Objekte aus den Daten des Servers zu machen und sie in korrektes JSON umzuwandeln.

Der Klient nutzt sie, um eingehende JSON Strings wieder in korrekte Objekte umzuwandeln und deren Inhalte darzustellen.

Die Klassen beinhalten alle übertragbaren Informationen: Bücher, Kategorien, Autoren, Verlage.

Jedes Objekt kann mithilfe von JSONGettern und JSONSettern geschrieben und gelesen werden. Die Attribute entsprechen nur denen, die von den Daten in der Datenbank auch verwendet werden sollen.

DBMeta Klasse:

Es gibt eine Metadaten-Klasse „DBMeta“, die dazu verwendet wird, um spezifische Anfragen nach Listen oder Listengrößen zu bewerkstelligen. Sie werden hauptsächlich dafür verwendet, um Objektlisten zu übertragen. Das Attribut „max“ wird dazu verwendet, bei partiellen Ergebnislisten Überblick über die Maximalanzahl eines Ergebnissatzes zu erhalten (Beispiel: es wird eine Teilliste aller Bücher von 1 -10 angefragt. Max speichert die Maximalanazahl von Büchern, damit festgestellt werden kann, ob das Ende der gesamten Ergebnisliste schon erreicht ist)

Server:

Der Server ist in die Packages „program“, „rest“, „parsing“ und „database“ aufgeteilt.

Unter „program“ finden sich die Klassen zum Starten des REST Servers.

Unter „rest“ finden sich alle REST Ressourcen die zur Verfügung stehen.

Unter „database“ befinden sich alle Klassen zur Datenbankanbindung.

Unter „parsing“ finden sich alle Klassen, die Operationen auf der Datenbank ausführen und Ergebnisse auslesen/ umwandeln.

Im Order „resources“ finden sich alle Parameter für die Verbindung mit der Datenbank, sowie die initialen Datenbankinhalte als SQL Datei.

Der Server wird auf der in Common spezifizierten Server Adresse gestartet.

Der Server verarbeitet alle Anfragen in unterschiedlichen Ressourcen-Klassen (xxxResource).

Diese Klassen können Manager-Klassen (xxxManager) verwenden, um Anfragen an der Datenbank zu bearbeiten.

Der Server arbeitet mit Responses und JSON Strings.

xxxResource Klassen:

UserResource: verarbeitet generelle Anfragen, die den Benutzer beim Login betreffen, oder den Server direkt betreffen sollen (Datenbank reset). Hier wird auch gespeichert, ob es bereits schon einen Hauptbenutzer gibt.

Alle weiteren Ressourcen enthalten die GET, PUT, POST und DELETE Funktionen, die Verwendet werden können, um jeweils Daten von der Datenbank zu holen oder zu verändern.

Es existiert jeweils getAll() und getBySelection(), die eine komplette Liste oder eine partielle Liste als DBMeta Objekt zurückgeben.

Die Ressourcen verwenden einen generellen ResourceManager, der die spezifischen Manager enthält.

ResourceManager:

Der ResourceManager enthält generelle Methoden, z.B. zum Umwandeln von JSON nach Objekten und umgekehrt.

Ebenso enthält er alle weiteren Manager, die von den einzelnen Ressourcen verwendet werden können, um mit Anfragen an die Datenbank umgehen zu können.

Der ResourceManager muss für alle Ressourcen verfügbar sein. Außerdem müssen die spezifischen Manager miteinander kommunizieren können. Deshalb wurde die Klasse als Singleton entworfen, sodass alle darin enthaltenen Manager nicht immer wieder neu erstellt werden müssen und diese Zugriff auf die Methoden haben.

xxxManager:

Jede Resource verwendet ihre eigene Manager Klasse.

Dort werden entweder die Objekte in die Datenbank übertragen, oder entsprechende Anfragen getätigt, um Daten von der Datenbank zu erhalten und in Objekte umzuwandeln.

Jeder Manager besitzt eine Methode, die ResultSets in eine Liste von Objekten umwandelt. Diese kann sowohl für Listenanfragen, als auch für Einzelanfragen benutzt werden.

Es gibt die Möglichkeit, Listen der jeweiligen Objekte anzufragen und als DBMeta Objekt zurückzugeben, sowie Anzahlen von Objekten in der Datenbank zu ermitteln.

Ebenso gibt es die Möglichkeit, Objekte nach ihren Primary Keys auszuwählen und zu löschen.

Ebenso können Objekte als Update (Veränderung) oder Insert (Erstellung) in die Datenbank übertragen werden.

CategoryManager und AuthorManager erlauben es ebenso, Einträge in die Verbundtabellen book_authors und book_categories zu tätigen oder daraus zu löschen.

Die SQL Befehle sind auf die in der database_setup.sql Datei festgelegte Implementierung festgelegt, da Änderungen an der Grundstruktur der Datenbank im Rahmen dieser Aufgabe nicht erwartet werden. Die Aufteilung von Resource und Manager erfolgte, um eben diese SQL Implementationen von anderer Logik zu trennen.

Klient:

Der Klient ist in Controller, Model und View aufgeteilt. Ebenso gibt es Interfaces die die Interaktion der Controller mit Model und View festlegen (Einsatz der Interfaces aus Softwaretechnik bekannt).

Unter „program“ befinden sich die Klassen zum Starten der JavaFX Anwendung.

Unter „view“ finden sich alle Klassen die die Benutzeransicht definieren.

Unter „controller“ finden sich alle Controller, die die Anfragen des Benutzers annehmen und verarbeiten.

Unter „model“ befinden sich alle Klassen, die Anfragen an den Server stellen können und erhaltende Daten speichern und verarbeiten.

Im Ordner „resources“ finden sich alle Elemente der Benutzeroberfläche. Dazu gehören alle verwendeten Icons und die FXML Szenendateien.

Der Klient ist JavaFX basiert und verwendet FXML Dateien, um die Benutzeroberfläche darzustellen.

View:

Der View stellt eine Stage bereit, die als View verwendet wird. Ebenso wird hier das Model erstellt.

Um Code nicht zu duplizieren, wurde das Laden aller FXML Szenen selbst auf den SceneController ausgelagert.

FXML Szenen:

Im „resources“ Ordner sind alle FXML Dateien zu finden.

„login.fxml“: Enthält das Login Fenster in dem Benutzerstatus und Verbindungsadresse gewählt werden können.

„main.fxml“: Enthält ein generelles Main Fenster, mit Filtern, Return/LogOut Button, einer Liste, Order und Page Buttons, sowie 5 zuschaltbare Buttons. Der jeweilige Controller stellt beim Laden die entsprechenden Elemente auf die passende Ansicht um.

„book.fxml“, „category.fxml“, „author.fxml” und “publisher.fxml”: Enthalten die spezifische Ansicht für alle entsprechenden Werte und einen Return Button, sowie zuschaltbare Submit und Delete Buttons.

SceneController:

Der SceneController stellt szenenübergreifende Funktionen bereit.

Dazu gehört vor allem die aktuelle Szene und dessen Controller zu wechseln. Der SceneController enthält alle Controller, damit darin enthaltende Einstellungen über einen Szenenwechsel hinaus behalten werden. Ebenso können szenenübergreifend Dialoge benutzt werden.

HINWEIS: Aus unbekannten Gründen können zu schnelle / häufige Szenenwechsel das Programm crashen. Der Fehler tritt an einem nicht erkennbaren Punkt in „com.sun.prism.d3d.D3DTextureData.getContext()“ auf.

Login Ansicht:

In der Login Ansicht kann der Benutzer sich entscheiden, in welchem Modus er die Anwendung betreten will.

Der Benutzer kann sich als normaler Benutzer oder als Hauptbenutzer anmelden.

Wenn sich der Benutzer als normaler Benutzer anmeldet, werden keine weiteren Einstellungen vorgenommen und er kann ohne weiteres das Programm betreten.

Wenn der Benutzer sich als Hauptbenutzer anmelden will, kann er das nur, wenn er erfolgreich beim Server angemeldet wird. Kann er nicht angemeldet werden, wird ihm der Fehlschlag angezeigt und er verbleibt im Login Menü. Wenn er erfolgreich angemeldet ist, wird sein Benutzerstatus für alle weiteren Ansichten übernommen. Der Hauptbenutzer verbleibt so lange Hauptbenutzer, bis er sich wieder erfolgreich abmeldet.

Der Benutzer kann in der Login Ansicht ebenfalls von der Host Adresse „localhost“ auf eine andere IP Adresse wechseln. Dazu muss er im entsprechenden Feld eine neue Adresse eintragen und den „Connect“ Button drücken.

Die neue Adresse wird im Rahmen dieser Aufgabe nicht auf Korrektheit geprüft.

Die Login Ansicht besitzt nur einen LoginController der alle beschriebenen Interaktionsmöglichkeiten bereitstellt und auf Model und View ausführt.

Hauptansicht:

In der Hauptansicht wird dem Benutzer nur das angezeigt, was er nach seinem Benutzerstatus auch sehen sollte.

Generell wird eine Liste mit von der Datenbank erhaltenden Elementen angezeigt. Die Liste entspricht den Suchvorgaben, die in der Ansicht vom Benutzer auch verändert werden können. Elemente in den Listen können durch Doppelklick in einer neuen Ansicht betrachtet werden.

Der Benutzer kann die Sortierrichtung ändern, den Listenausschnitt verändern und Textfilter (oder Kategorie Filter) setzten. Ebenso kann der Benutzer die Ansicht durch einen Button wieder verlassen. Wenn der Hauptbenutzer das Programm verlassen will, muss er davor erst wieder erfolgreich beim Server abgemeldet werden.

Der normale Benutzer sieht nur die Bücher-Ansicht.

Der Hauptbenutzer erhält 5 weitere Buttons, mit denen er entweder die Datenbank zurücksetzen kann, oder die aktuelle Ansicht wechseln kann.

Es stehen zusätzlich zu den Büchern noch Ansichten für Kategorien, Autoren und Verlage bereit. Diese enthalten aber keinen „Kategorie“-Filter.

Durch das erneute Klicken eines Buttons kann der Benutzer auch neue Einträge dieser Art erstellen. Dem Hauptbenutzer werden ebenso Edit und Delete Buttons für jedes Listenelement zur Verfügung gestellt, um spezifische Einträge zu bearbeiten oder zu verwerfen.

Da es 4 verschiedene Ansichten gibt, die aber alle im selben View laufen, erben die entsprechenden spezifischen Controller von einer abstrakten „MainController“ Klasse. Diese implementiert alle Funktionen, die unabhängig von der konkreten Ansicht immer ausgeführt werden können.

Die spezifischen Ansichts-Controller implementieren dann die jeweils notwendigen Aktionen, um mit entsprechenden Anfragen an den Server oder an die View umgehen zu können. Durch Vererbung stehen den spezifischen Klassen immer die generellen Funktionen der Oberklasse zur Verfügung, was für diese Anwendung sinnvoll erschien.

Konstruktoren verhindern aus unbekannten Gründen den Start der FXML Datei, deshalb kann ein initialisierter MainController durch eine „setup“ Methode erstellt werden.

Spezifische Ansicht:

Wenn der Benutzer spezifische Elemente ansehen, bearbeiten oder erstellen will, wird eine neue Ansicht zur Darstellung aller vorhandenen Werte dieses Elementes benutzt.

Jede Art von Element hat dabei eine eigene FXML View (Buch, Kategorie, Autor, Verlag).

Der Hauptbenutzer erhält ebenfalls Zugriff auf Edit Buttons, sowie die Funktion die aktuell eingetragenen Werte in die Datenbank zu übernehmen oder das aktuelle Objekt aus der Datenbank zu löschen.

Der Benutzer kann den Modus jederzeit wieder verlassen und gelangt wieder in die entsprechende Hauptansicht zurück.

Alle verwendeten Controller können auch hier von einer abstrakten Klasse „ViewController“ abgeleitet werden. Diese fasst alle Funktionen zusammen, die die spezifischen Ansichten gemeinhin besitzen sollen.

Die spezifischen ViewController legen dann konkrete Implementationen für ihre Funktionsweisen fest.

Ebenso werden in den konkreten Klassen alle Edit-Methoden festgelegt, die beim Klicken von den Edit Buttons aufgerufen werden.

Im Gegensatz zu den Buttons in der Liste der Hauptansicht sind die Edit Buttons der Werte in der spezifischen Ansicht in der FXML Datei festgelegt und werden nicht dynamisch hinzugefügt, um einen zu komplexen Aufbau in der FXML Datei zu verhindern.

Main Model:

Das Main Model fasst den Zugriff zu allen gespeicherten Daten und Möglichkeiten zum Stellen von Anfragen an den Server zusammen.

Das Model selbst ist auf fünf spezifische xxxModel Klassen ausgelagert.

Jede Model Klasse speichert alle benötigten Daten einer Art zusammen.

Die Controller können die „MainModel“ Klasse verwenden, um generelle Benutzereinstellungen zu tätigen, wie Benutzerstatus und die Einstellung von Edit / Create Modus. Diese Einstellungen sollen über alle Controller konsistent sein, deshalb werden sie im Model verarbeitet.

Request Klasse:

Die „Request“ Klasse wird von den einzelnen Model Klassen verwendet, um Anfragen an den Server zu stellen, Antworten zu erhalten und diese auch zu bearbeiten.

Sie enthält Debugging Methoden, Methoden zum Auslesen von Serverantworten und der Umwandlung von JSON-Strings auf POJOs.

Ebenso enthält sie eine Methode um Anfragen an den Server zu stellen (GET, PUT, POST, DELETE).

xxxModel:

UserModel: Enthält alle Methoden zum Verarbeiten von User-Einstellungen (Login/Logout), sowie Datenbank-Reset und Hostname Veränderung.

xxxModel: Jede Model Klasse enthält die entsprechenden Methoden zum erhalten, bearbeiten, erstellen und löschen von entsprechenden Ressourcen.

Jede Model-Klasse speichert eine Liste an Objekten und ein konkretes Objekt. Die Listen können primär zum Darstellen in der Hauptansicht verwendet werden, oder zum Beispiel für Listen in der Filter-Auswahl (Kategorie-Filter). Das konkrete Objekt speichert ein Ansichts-Objekt, welches alle Informationen für ein Objekt enthält, welches dann in einer entsprechenden Ansicht dargestellt wird.

Quellen:

Die FXML Dateien wurden mithilfe von SceneBuilder erstellt.

Die Auswahl bestimmter Tabelleninhalte (Inserts) für die database_startup.sql erfolgte testweise mit ChatGPT (openai.com), erforderte jedoch deutliche Korrekturarbeit.

Verwendete Grafiken für die Benutzeroberfläche:

https://cdn-icons-png.flaticon.com/512/33/33281.png

https://www.freeiconspng.com/uploads/writer-icon-png-25.png

https://upload.wikimedia.org/wikipedia/commons/thumb/3/31/VisualEditor_-_Icon_-_Book.svg/2048px-VisualEditor_-_Icon_-_Book.svg.png

https://icons.veryicon.com/png/o/commerce-shopping/icon-of-lvshan-valley-mobile-terminal/home-category.png

https://upload.wikimedia.org/wikipedia/commons/thumb/a/ae/Factory_icon.svg/2198px-Factory_icon.svg.png

https://cdn-icons-png.flaticon.com/512/31/31863.png

https://cdn-icons-png.flaticon.com/512/3405/3405244.png

https://upload.wikimedia.org/wikipedia/commons/thumb/8/8a/OOjs_UI_icon_edit-ltr.svg/2048px-OOjs_UI_icon_edit-ltr.svg.png

https://static.thenounproject.com/png/3201006-200.png

https://static.thenounproject.com/png/2331585-200.png

https://upload.wikimedia.org/wikipedia/commons/thumb/b/b1/Back_Arrow.svg/2048px-Back_Arrow.svg.png

https://icons-for-free.com/download-icon-restore+rotate+undo+icon-132018318493093365 5_512.png

https://icons-for-free.com/iconfiles/png/512/circle+close+cross+delete+exit+remove+icon-1320085939591374353.png

Eidesstattliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit bisher bei keiner anderen Prüfungsbehörde eingereicht, sie selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Wernigerode, 08.02.2023