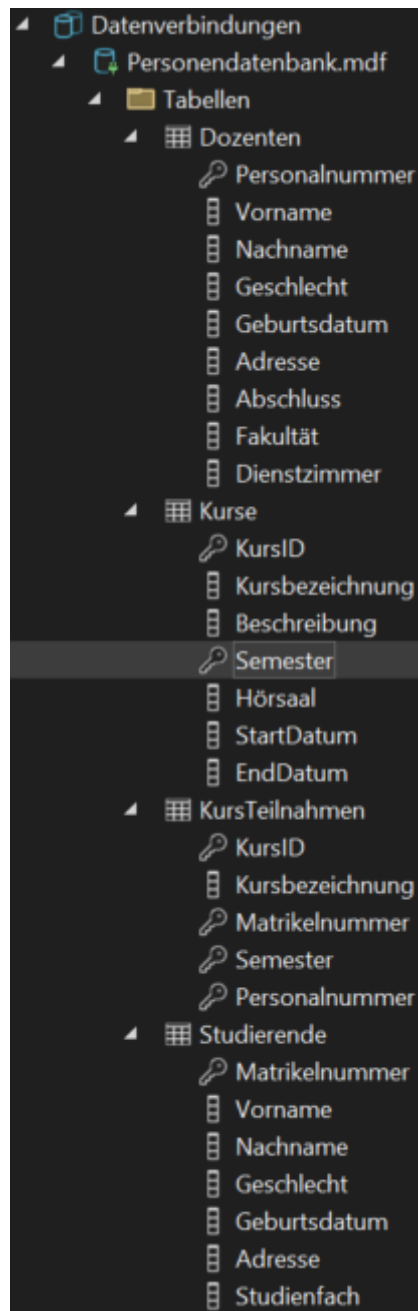


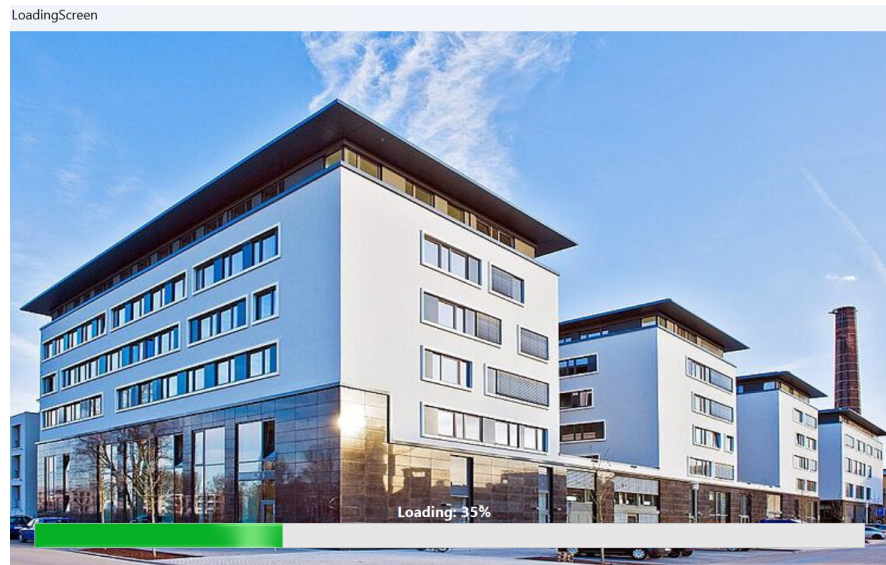
C# Studienleistung

1. Allgemeines

Das folgende Projekt einer Hochschulverwaltung wurde mit der IDE Visual Studio 2022 und dem .NET Core Framework Runtime v8.0 entwickelt. Außerdem gibt es ein User Interface, das mit Hilfe von Windows Forms aufgebaut wurde. Dieses soll den Benutzern die Interaktion mit den Daten in der Datenbank erleichtern. Die Daten, wie z. B. Dozenten, Studenten und Kurse sowie Kursteilnahmen wurden dabei in einer Personendatenbank als .mdf - Datei lokal gespeichert (siehe Projektordner). Es gibt insgesamt 4 Datenbanktabellen, wie im folgenden Bild dargestellt:




Weiterhin im Projekt enthalten ist die Klasse LoadingScreen, die beim Start des Programms das Laden simuliert:




Eine Klasse Hochschulverwaltung übernimmt nach dem Laden des Programms als Hauptseite des Projekts die Aufgabe, dass Studenten und Dozenten zu Kursen hinzugefügt, bearbeitet oder gelöscht werden können:

Hochschulverwaltung (Hauptmenü)


CRUD-Operations für Studenten, Dozenten und Kurse:



Studenten



Dozenten



Kurse

Zuordnung von Studenten und Dozenten zu Kursen:

Kursbezeichnung:

KursID:

Matrikel.-Nr.:

Semester:

Pers.-Nr.:

KursID	Kursbezeichnung	Matrikelnummer	Semester	Personalnummer
1009	Informatik I	10007	3	4
1009	Informatik I	11008	1	1008
1013	Informatik II	11008	2	9
2013	Datenstrukturen &	11008	3	1008
2013	Datenstrukturen &	11008	3	1009
2014	Grundlagen Digitaltechnik	10007	1	4
2014	Grundlagen Digitaltechnik	10007	3	9

Diese werden als sog. CRUD-Operationen bezeichnet. Ein Eintrag kann durch Auswahl (anklicken) der jeweiligen Zeile im DataGridView selektiert werden. Dieses befindet sich unter den Steuerelementen (siehe Bild). Aufgrund der Datenbankmodellierung (siehe Extra Blatt), können nur passende Einträge generiert werden. Passend meint hier, konsistente Dateneinträge, in denen die Fremdschlüsselbeziehungen übereinstimmen.

Dadurch sollen die erste, die zweite und die dritte Normalform eingehalten werden. Insgesamt wäre eine Implementierung über “Code to DB” wahrscheinlich einfacher gewesen. Dies würde im nächsten Projekt bevorzugt werden. Über die oberen Schaltflächen können Studenten, Dozenten und/oder Kurse jeweils bearbeitet werden (siehe Bild oben). “Hinter” den jeweils genannten Schaltflächen (sog. Buttons) verbirgt sich jeweils eine eigene Klasse und damit ein separater Screen, wie nachfolgend gezeigt:

Studentendatenbank

Vorname

Nachname

Geschlecht ☒ männlich ☐ weiblich

Geb.-Datum

Adresse

Studienfach

Kursdatenbank

Kursbezeichnung

Kursbeschreibung

Semester

Hörsaal

Kurs Start-Datum

Kurs End-Datum

Dozentendatenbank

Vorname Fakultät

Nachname Dienstzimmer

Geschlecht ☒ männlich ☐ weiblich

Geb.-Datum

Adresse

Abschluss

Die Formelemente sind ähnlich aufgebaut und beinhalten ähnliche Funktionalitäten. Labels beschreiben jeweils Eingabefelder wie z. B. Combo-Boxen oder Textfelder. Über diese Elemente können Daten für Kurse, Studenten und Dozenten eingegeben werden. Unterhalb dieser Elemente befindet sich in jedem Screen ein DataGridView. Dadurch werden die Daten aus der Datenbank zu jeder Zeit visualisiert. Sobald eine Eingabe erfolgt ist, wird dieser aktualisiert.

Eine weitere Klasse DB (die sog. Datenbank) übernimmt die Verbindung zur DB über einen sog. Connection-String und führt die jeweiligen SQL-Statements aus. Im vorliegenden Projekt wurde nicht mittels Linq gearbeitet. Stattdessen wurden die SQL Statements mittels eines formatierten Strings definiert. Künftig würde auch hier eine Eingabe mittels Linq (gerade für große Datenmengen) sinnvoll erscheinen.

2. Probleme/Schwierigkeiten/Beachten

- Oftmals wird beim Selektieren der Zeilen im DataGridView, diese nicht erkannt. Deshalb muss öfter auf die jeweilige Zeile gedrückt werden. Zudem gab es während der Entwicklung in der IDE ständig Darstellungsprobleme von Formelementen. Stattdessen werde ich künftig WPF "Windows Presentation Foundation" verwenden. Weitere Recherchen ergaben, dass ASP.NET 6.0 heute oft für Webentwicklung angewandt wird und oftmals mit Node.JS konkurriert. D. h. dieses Framework (Werkzeug) wäre ggf. eine Alternative zu Windows Forms.
- In der Klasse DB in der Methode DataBaseConnection() muss der jeweilige ConnectionString mit dem richtigen Speicherort angepasst werden, sobald die Datenbank lokal gespeichert wurde (.mdf.-Datei).
- In der Klasse Hochschulverwaltung wurde das Textfeld KursID auf enabled=false gesetzt. D. h. eine Auswahl des jeweiligen Kurses wird über die Kursbezeichnung gemacht. Dies ist für den Benutzer verständlicher. Die KursID wird entsprechend automatisch angepasst.
- In der Klasse Hochschuldatenbank und in der Klasse DB wird aufgezeigt, wie eine gute Dokumentation aussehen könnte. Hier werden Funktionen und Parameter beschrieben und inline Kommentare zwischen Codezeilen verwendet.
- In einem realen Projekt existieren ein Lasten- und ein Pflichtenheft. Demnach werden die Anforderungen (Funktionalität, Ergonomie, Wartbarkeit, Qualität etc.) überprüft. Eine geeignete

Bedienungsanleitung wird dem Unternehmen bzw. dem Benutzer mit ausgeliefert. Hierauf wurde ebenfalls verzichtet.

- In der Klasse Hochschulverwaltung bin ich mit der Methode FillComboData() nicht zufrieden. Hier ist der Code bzw. die Struktur unsauber bzw. nicht gut genug überdacht.
- Die Tabellen wurden in der Datenstruktur Liste abgelegt, damit die SQL-Abfrage dynamisch wird. Allerdings werden die Strings später wieder verwendet, sodass die Liste wenig Sinn ergibt. Demnach hätten diese auch manuell eingetippt werden können. Sobald es neue Tabellen in der DB gibt, muss die Datenstruktur sowieso angepasst werden. Zuvor hatte ich überlegt, ob die Tabellennamen und Spaltennamen gemeinsam als Key-Value-Pair in einem Dictionary abgelegt werden können. Allerdings entstand hierbei ein Problem. Werden zwei Primärschlüssel (KursID und Semester) benötigt, um die Eingabefelder zu befüllen, muss eine weitere Schleife programmiert werden. Folglich erhöht sich die Laufzeit auf $O(n) = O^3$, was die Leistung (Performanz) erheblich reduziert. Auf weitere Aspekte soll an folgender Stelle nicht weiter eingegangen werden. Deshalb beantworte ich gerne Rückfragen persönlich.

3. Ansatz zur ER-Datenmodellierung

siehe extra Blatt "ERM_SL_WS2324.pdf"

4. GitHub Repo:

<https://github.com/PatrickSchubert-Munich/Hochschuldatenbank.git>