Hochschule Esslingen
University of Applied Sciences

Projektdokumentation Datenbanken 2

im Studiengang

Wirtschaftsinformatik

vorgelegt von

Jannis Joos
Matr.-Nr. : 765377
am 5. Januar 2024
an der Hochschule Esslingen

Prüfer: Prof. Dr. Dirk Hesse

Kurzfassung

Die entwickelte TODO-Liste-Applikation ist eine Java-Anwendung, die die Organisation von Aufgaben ermöglicht. Sie basiert auf einer relationalen Datenbank und ermöglicht Benutzern das Erstellen, Aktualisieren und Löschen von Aufgabenkarten mit spezifischen Identifikatoren, Titeln und Erstellungszeiten. Die Anwendung bietet ein benutzerfreundliches Button-Interface um den Status von Aufgaben zwischen den Kategorien "Geplant", "In Bearbeitung" und "Erledigt" zu verwalten. Dabei integriert sie Funktionen zur Verhinderung von Duplikaten und zur Archivierung gelöschter Aufgaben. Die Implementierung erfolgte unter Verwendung von Java Swing für die Benutzeroberfläche und einer Microsoft SQL Server-Datenbank für die Datenverwaltung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Projektsteckbrief

Das Projekt zielt darauf ab, eine umfassende TODO-Liste-Applikation bereitzustellen, die Benutzern ermöglicht, Aufgaben zu erstellen, zu organisieren und zu verwalten. Die Anwendung verwendet eine relationale Datenbank, um den Fortschritt von Aufgaben zu speichern. Benutzer können Karten mit Titel, ID und Erstellungszeitpunkt erstellen, verschieben und löschen. Ein Button-Interface ermöglicht eine einfache Aktualisierung des Aufgabenstatus zwischen den Zuständen "Planned", "In Progress" und "Done". Der Projektumfang umfaßt die Implementierung von Swing GUI Elementen, die Interaktion mit einer SQL Server-Datenbank sowie die Integration von Funktionen zur Verhinderung von Duplikaten und zur Archivierung von gelöschten Aufgaben.

Projektbeschreibung

Wie in Abbildung 1 zu sehen beinhaltet das Programm drei Felder : Planned, InProgress und Done.

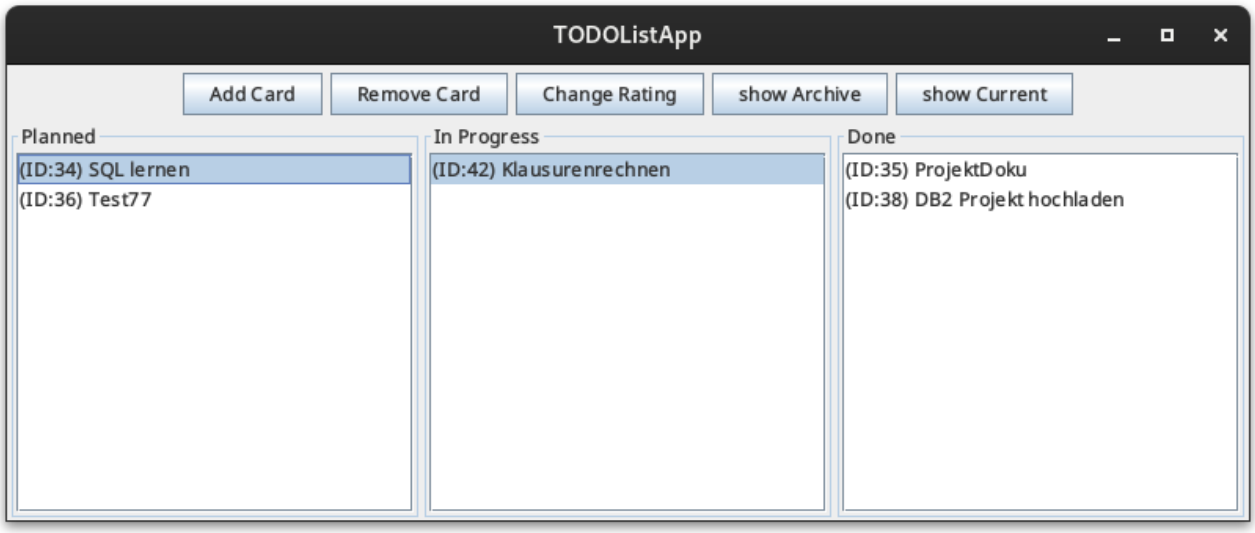

Abbildung 1 : GUI der TODOListApp

Bei dem Drücken des ,,AddCard“ Buttons erscheint ein Popup-Fenster das zur Aufforderung des Namens für die erste Karte signalisiert, welches in Abbildung 2 dargestellt ist. Die ID für die Karten werden bei der Übergabe des Wertes an die Datenbank automatisch erstellt. Bei dem Feld ID ist eine Identität wie folgend realisiert, welches dazu führt, dass immer bei dem hinzufügen einer neuen ID diese hochgezählt wird.

```
[ID] INT IDENTITY (1, 1) NOT NULL
```

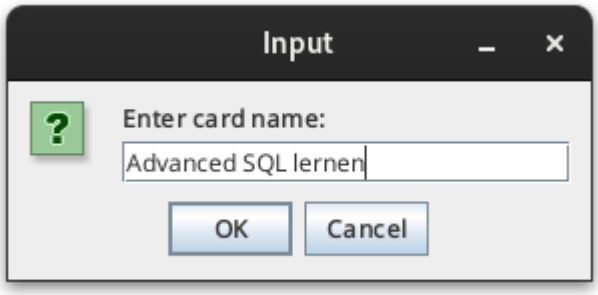

Abbildung 2 : Eingabefenster für den Kartennamen

Für das löschen einer Karte wird nach der ID gefiltert. Beim drücken auf den „Remove Card“ Button öffnet sich ein Äquivalentes Fenster welches Auffordert eine ID für das löschen der gewünschten Karte einzugeben. Nach dem Drücken auf OK verschwindet diese Karte von der Oberfläche und wird von der Datenbank in die Archiv-Tabelle übernommen.

Umgang mit Exeptions

Falls ein Feld leer gelassen wird, meldet das Programm sofort einen Fehler zurück, wie in Abbildung 3 gezeigt.

Abbildung 3 : Error bei der Eingabe

Bei doppelten Events ist ein Trigger integriert, der aktiviert wird, und somit das einfügen verhindert.

Archiv

Des weiteren beinhaltet die Datenbank eine Archiv Tabelle. Falls Karten gelöscht werden. Wird ein Trigger gestartet bei dem in die Archiv Tabelle ein Backup der Karte und das Datum der Löschung eingefügt wird. Beim Klicken auf den „show Archive“ Button öffnet sich ein Fenster, das zum eingeben eines Datums auffordert, wie in Abbildung 4 Dargestellt.

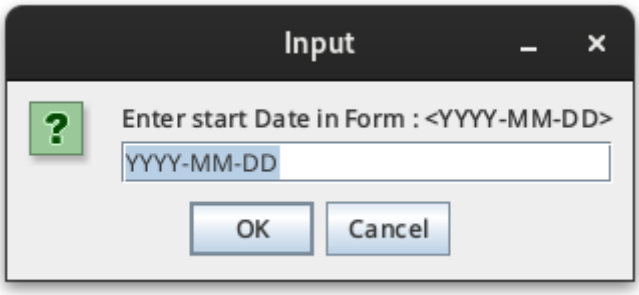

Abbildung 4 : Fenster zur Dateneingabe

Zu Demonstrationszwecken wurde das Datum 2024-01-01 eingegeben. Mit der Übergabe des Datums als SQL-Timestamp an die Stored Procedure „GetCardsFromArchive“ werden alle Karten aus der Archiv-Tabelle in die Oberfläche geladen. Zusätzlich wird auch die Zeit der Löschung angezeigt, was in Abbildung 5 zu sehen ist.

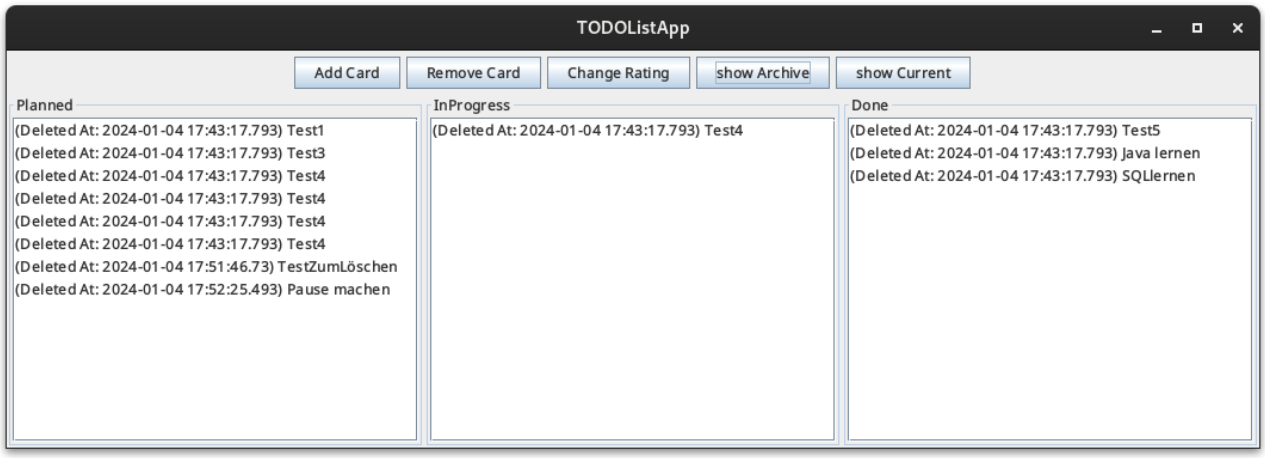

Abbildung 5 : Archiv der gelöschten Karten

Aufbau der Anwendung

Die Anwendung ist in zwei Teile Aufgebaut, dem Graphischen Benutzer Interface welches in Java realisiert ist und der Datenbank, für welche MSSQL in der Expressvariante zum Einsatz kommt. Die Java Anwendnug besteht aus zwei Klassen der Main-Klasse TODOListApp, in der die Java-Swing Komponenten und Logik in dem Frontend vorhanden ist, und der Datenbank-Klasse bei der die Verbindung zur Datenbank hergestellt wird und Methoden die mit dieser Interagieren.

Die Datenbank besteht aus zwei Tabellen, der timetable Tabelle, welche die aktuellen Karten verwaltet und anzeigt, und der Archiv-Tabelle, welche für Archivierte Karten bereitgestellt ist.

Datenbankverbindung in Java

Bei der Verbindung mit der Datenbank wurde sich an das Example.java aus dem Vorlesungsskript gehalten, wie in Abbildung 6 dargestellt.

Zuerst wird eine Methode in der Database.java Klasse erstellt, die zum Verbinden mit der MSSQL-Server Datenbank dient. Diese wird mit einer ConnectionUrl in der folgenden Form erstellt.

"jdbc:sqlserver://
<server>:<port>;encrypt=true;databaseName=AdventureWorks;user=<user>;passwo
rd=<password>"

In er ConnectionUrl wird als Host eine lokale MSSQL-Server Variante verwendet, welche für gewöhnlich auf den Port 1433 hört. Der Name der Datenbank ist DB2Projekt. Der Standard Administrator ist in SQL-Server als „sa“ bezeichnet.

```
public void MSSQLconnect() {
    String dbHost = "localhost";
    String dbPort = "1433";
    String dbName = "DB2Projekt";
    String dbUser = "sa";
    String dbPass = [redacted]

    String connectionUrl = "jdbc:sqlserver://" +
            dbHost + ":" +
            dbPort + ";" +
            "databaseName=" + dbName + ";" +
            "user=" + dbUser + ";" +
            "password=" + dbPass + ";" +
            "encrypt=" + "true" + ";" +
            "trustServerCertificate=" + "true" + ";";
    try {
        Class.forName( className: "com.microsoft.sqlserver.jdbc.SQLServerDriver");
        con = DriverManager.getConnection(connectionUrl);
        sm = con.createStatement();
        System.err.println("Connection Successfull!");
    } catch(Exception e) {
        System.err.println("Fehler beim Verbinden: " + e);
    }
}
```

Abbildung 6: Verbindung zur Datenbank

Des weiteren war es nötig den mssql-JDBC Driver in das Projekt zu implementieren, welcher auf der Offiziellen Seite von Microsoft heruntergeladen werden kann.

https://learn.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver16

Wichtig ist es hier die .jar Datei für den Treiber als externe „.jar“ dem Projekt hinzuzufügen, sonst kann es zu Problemen kommen.

Graphical User Interface

Das GUI (Graphical User Interface) wurde in Java mit Swing aufgebaut. Die gewünschten Formate für die GUI und wie diese Umgesetzt werden ist in der Dokumentation auf :

```
https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/package-summary.html
```

Zu finden.

Aufbau der Datenbank

Die Datenbank ist zu Testzwecken auf dem Lokalen Betriebssystem gehostet welche auf den Port 1433 hört. Eine Verbindung mit dem Azure Data Studio sowie mit dem MSSQL-Management-Studio war erfolgreich. Der Name der Datenbank ist „DB2Projekt“. In der Datenbank ist eine Tabelle definiert „timetable“, welche alle Informationen zu den Karten erhält. Erstellt wurde sie wie in Abbildung 5 Dargestellt :

```
CREATE TABLE [dbo].[timetable] (
[ID] INT IDENTITY (1, 1) NOT NULL,
[Time created] DATETIME NULL,
[Time due] DATETIME NULL,
[Event] NVARCHAR (50) NULL,
[Rating] NVARCHAR (50) NULL,
CONSTRAINT [PK_timetable] PRIMARY KEY CLUSTERED ([ID] ASC)
);
```

Abbildung 5 : Erstellen der timetable Tabelle

In der ersten Spalte ist die ID welche als Primary-Key fungiert. Gefolgt von einer Spalte mit dem Zeitpunkt an dem die Karte der Datenbank hinzugefügt wird.

SQL Funktionen

Insert Funktion

Für das Einfügen von Daten zur Datenbank ist in der Database Klasse eine Methode addEvent() mit einem String Parameter eventName definiert. Falls diese Methode aufgerufen wird, und ein entsprechender Wert übergeben wurde, wird in die Tabelle „timetable“, das lokale Datum der Erstellung, das Event als eventName übergebenen Parameter und als Rating, Planned in die dafür vorgesehenen Spalten eingefügt. Hier wird nochmal darauf hingewiesen, dass sich die ID Automatisch generiert, da für diesen Wert in der Spalte eine Identity(1,1) eingetragen wurde.

Update Funktion

Um das Rating, also den Zustand der Karte von z.B. „Planned“ in „in Progress“ zu ändern ist eine Update Funktion in der Database Klasse bereitgestellt. Übergeben werden die Parameter „ID“ und „Rating“ als String. Beim aufrufen und übergeben der richtigen Parameter wird in der Tabelle „timetable“ ein UPDATE ausgeführt und das Rating für die jeweilige ID neu gesetzt.

Delete Funktion

Wenn Karten gelöscht werden müssen wird das mit klicken auf ,,Remove Card“ ausgeführt. Wieder ist eine Funktion in der Datenbank klasse dafür notwendig. Die Funktion ,,deleteCard“ erhält als Parameter die ,,ID“ der Karte und bereitet so das SQL Statement auf. Gelöscht wird die Zeile in der die zugehörige ID steht.

Trigger-Funktionen

PreventDuplicateEvent

Der Erste Trigger vermeidet das Einfügen von doppelten Kartennamen. Diese Logik ist direkt in die Datenbank implementiert. Falls ein Kartenname der schon bereits existiert versucht wird hinzuzufügen greift der Trigger ein und verhindert das Einfügen. Die Fehlermeldung ist dann in der Konsole sichtbar.

Wichtig ist hier, dass auch verglichen wird, dass das Ereignis nicht NULL ist, um sicherzustellen, dass NULL-Werte nicht für Duplikat Überprüfungen verwendet werden. Was mit ,,WHERE i.ID <> t.ID“ Sichergestellt wird.

ArchiveCardOnDelete

Der zweite Trigger ist für das Archivieren der Karten zuständig. Falls Der Benutzer eine Karte löscht wird diese in die Tabelle ,,ArchiveTimetable“ geschrieben. An der Stelle von der erstellten Zeit steht jetzt die Zeit der Löschung der Karte drin.

Stored Procedure

Das Stored Procedure ,,GetCardsFromArchive“ ist wie der Name schon andeutet dafür zuständig Karten von dem ArchiveTimetable Table zurückzugeben die Nach dem StartDate, welches der Prozedur übergeben wird, erstellt wurden. Somit ist es möglich nach Abfrage eines Datums, alle gelöschten Karten auszugeben. Mit einer WHERE Bedingung und einem größer gleich (>=) Operator wird geprüft welche Karten ab diesem Datum in der Vergangenheit liegen.

Schwierigkeiten

Lange hat es gedauert, den JDBC Driver zu installieren, auch in Foren wie Stackoverflow bin ich nicht weitergekommen. Ich konnte mich über das Azure Data Studio und über das MSSQL-Management Studio (MSSMS) erfolgreich mit der Datenbank verbinden.

Gelöst hat sich das Problem dann als die Jar-Datei direkt als Externe „.jar“ zum Projekt hinzugefügt wurde.

Bei dem Versuch ein ,,Drag and Drop“ Menü zu bauen kam ich an meine Grenzen, was vielleicht Leicht aussieht, dauert aber eine menge Zeit, weshalb aus Zeit und Komplexitäts- Gründen davon abgesehen wurde.

Um die Karten zu ändern und hinzuzufügen wie auch zu löschen sind hier Knöpfe benutzt worden. Wie in der Projektbeschreibung aufgeführt.

Bei dem Trigger der doppelte Events prüfen soll, kam das Problem auf, dass danach gar keine Events mehr hinzugefügt werden konnten. Die Datenbank hat bei allen Inserts reagiert und einen Fehler geworfen. Behoben wurde das Problem mit einem Vergleich der ID mit „WHERE i.ID <> t.ID“.

Weitere Schwierigkeiten gab es beim zugreifen auf einzelne Elemente in der Tabelle. Es hat eine weile gedauert, bis rausgefunden wurde das für Datenzeilen auch ein dementsprechender Variablentyp verwendet werden muss, da es mit String nicht möglich war.

Die korrekte Lösung für das Problem ist nachfolgend dargestellt.

```
Timestamp Date = rs.getTimestamp("TimeDeleted");
```

Auch bei einem weiteren Problem war dann sofort klar, wer mit Daten (Datums) arbeitet braucht auch die dementsprechende Variable. Z.B. bietet sich aus der Java SQL Bibliothek hervorragend die sql.Date Variable an, wie nachfolgend gezeigt.

```
java.sql.Date sqlDate = java.sql.Date.valueOf(startDate);
List<String> archive = db.getCardsFromArchive(sqlDate);
```

Fazit

Das Datenbankprojekt war eine Wertvolle und Interessante Erfahrung. Vom Installieren der Datenbank bis zum erstellen von gespeicherten Prozeduren. Ich würde so eine Art von Projekt auf jeden Fall weiterempfehlen, da es das Verständnis, wie eine Datenbank aufgebaut ist und wie man mit dieser Interagiert deutlich verbessert.

Besonders Interessant fand ich die Möglichkeiten, in der Datenbank zu Programmieren. Davor dachte ich immer Logik muss im Frontend gegeben sein, aber mit der Macht von Triggern oder auch Prozeduren ist das anderes.

Lessons Learned

Ich habe gelernt wie man den MSSQL-Server unter Linux installiert und mit einer Applikation in Java verbindet, wie man eine Datenbank aufbaut und den JDBC-Treiber erfolgreich installiert. Des weiteren habe ich meine SQL-Kenntnisse und auch Kenntnisse in Java deutlich gestärkt und somit wieder aufgefrischt.

In diesem Projekt habe ich Java Swing benutzt was sich ausgezeichnet zum Programmieren von Oberflächen eignet, und damit neue Elemente Kennengelernt.