MLZ KREDITKARTENPRUEFUNG DB2

Für die Zugangsprüfung einer Kreditkartenzahlung soll eine kleine Windows Anwendung mit Visual Studio und C# entwickelt werden. In der Anwendung soll die Zugangsprüfung und die Erfassung der Zugangs- und Kontodaten realisiert werden. Sämtliche Daten müssen in einer MS-SQL Datenbank gespeichert werden. Einfachheitshalber kann auf die Verschlüsselung verzichtet werden.

EINLEITUNG

Für diese Arbeit war ein gutes Relationsmodell sowie durchdachte Planung sehr wertvoll. Ich konnte extrem viel Zeit dadurch ersparen, indem ich alles sehr gut geplant habe und weniger an Schwierigkeiten gestossen bin, welche auf eine schlechte Planung heraus entstanden. Mein RM Modell welches ich erstellt habe konnte ich immer wieder anschauen und das Resultat bildlich vorstellen.

REALISIERUNG / VORGEHEN

Das Vorgehen ist von Person zu Person unterschiedlich. Oft empfiehlt man bei einer Realisierung die Logik zuerst zu implementieren und danach das GUI, zumindest haben wir so gelernt. Jedoch mein Vorgehen war bei dieser Arbeit etwas unterschiedlich.

Mein Vorgehen:

1. Planung
2. Erstellung RM Modell
3. Erstellung Datenbank und entsprechende Tabellen
4. Test der Datenbank durch diverse Insert/Delete/Update
5. GUI Design
6. Logik auf Datenbankebene
7. GUI mit Logik verbinden
8. Testen
9. Dokumentieren

Author: Parthipan Nagulanandan
Datum: 05.05.2017
Version: 1.3

Das GUI

Abbildung 1:: Hauptseite

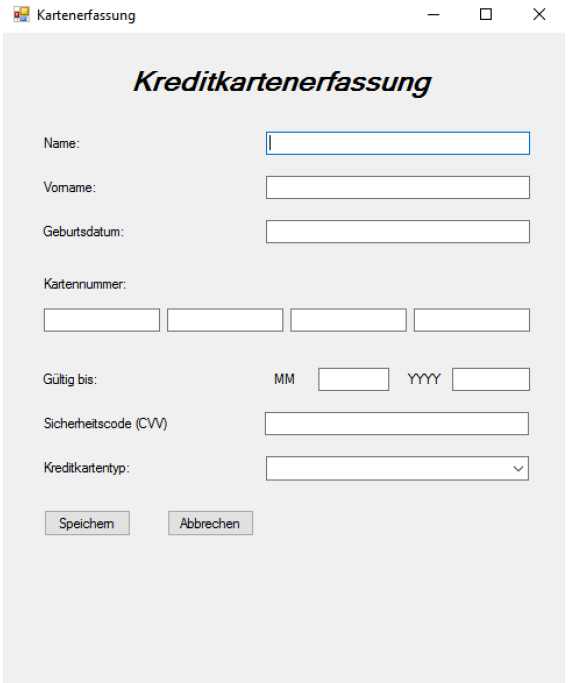

Abbildung 2: Karte erfassen

Die GUI Seiten sehen seht ähnlich und sind einfach gehalten. Das GUI ist auch Kompakt.

Beschreibung Karte erfassen

Um eine Karte im System zu erfassen gelangt man über das Hauptmenu indem man auf der Hauptseite auf erfassen klickt. Dies wurde anhand der Anforderung so implementiert. Klar, in der Realität erfasst man die Karte nicht selber im System. Laut Anforderung jedoch sollte man die Karte erfassen können. Wenn man nun auf ‚Erfassen' klickt und die Informationen eintippt wird beim Speichern validiert. Ursprünglich wollte ich die Validierung auf C# Ebene implementieren, da dies viel benutzerfreundlich sowie einfacher ist. Laut Anforderung muss die Logik auf Datenbank Ebene sein. So wurde die Validierung mit einer gespeicherten Prozedur gelöst. Bei Meldungen gibt es zwei Arten. Entweder erhält man Validierungsmeldungen oder normale Meldungen. Wenn Felder nicht befüllt wurden oder das Format nicht korrekt ist, wird eine Validierungsmeldung angezeigt. Wenn jedoch sonstige Fehler oder Meldungen gibt wie z.b. wenn Karte bereits vorhanden ist, dann ist die normale Meldung aktiv.

Beschreibung Sicherstellung Zugangsberechtigung

Die Sicherstellung der Zugangsberechtigung erfolgt durch eine gespeicherte Prozedur, welches zuerst die Eingabe wie beim Erfassen auch validiert und anschliessend Jeder Eintrag im Datenbank gesucht und sichergestellt. Die Sicherstellung kann nur erfolgen, wenn die Karte existiert und keine Validierungsfehler vorhanden sind. Aus Debugging Gründen und detailliertere Log sowie GUI Meldung wurde jeder falsche Eintrag auch erkannt und ausgegeben. In der Realität dürfte dies aus meiner Sicht nicht sichtbar sein was man falsch eingetippt hat.

Beschreibung Sperrung der Karte

Die Kartensperrung sowie der Logik mit 3 Versuchen wurde über einen BIT/Boolean und Integer realisiert.

Die Logik sieht nun so aus:

- Wenn Karte noch nicht blockiert und der Zähler noch unter 3 ist hat, kann man nochmals versuchen
- Wenn Karte nicht blockiert und Angaben richtig gemacht wurden wird BIT auf false gesetzt und Zähler auf 0.
- Wenn Karte blockiert BIT auf true, wird der Zähler weiter addiert, jedoch ohne Zugriff.

Beschreibung Protokollierung

Jede Meldung welches auf GUI sichtbar ist, wird auch gleichzeitig protokolliert.

Folgende Angaben sind pro Eintrag in Logmeldung vorhanden:

- Id Log
- Zeit Log
- Aktuelle Benutzer
- Beschreibung
- Name
- Vorname
- Geburtsdatum
- Kartennummer
- Gültigkeit Monat
- Gültigkeit Jahr
- Sicherheitscode (nur sichtbar, weil eine Übung)
- Kartentyp

FASZIT

Die Arbeit war interessant um es zu bearbeiten. Jedoch habe ich die Arbeit zeitlich sehr unterschätzt. Ich habe für die Arbeit eine Woche eingeplant (abends), jedoch habe ich die geschätzten 16h Arbeit vom Dozent sehr schnell erreicht und mir war zu wenig Zeit vorhanden um mich um die Details zu kümmern. Trotzdem konnte ich in dieser Zeit die Syntax von Datenbank besser kennenlernen und intensiver anwenden, was mir sicherlich viel gebracht hat. Was mir extrem aufgefallen ist, ist die Validierung. Die Anforderung war die Logik auf Datenbankebene zu implementieren. Ob damit die Validierung auch gemeint war, kann ich leider nicht beurteilen. Aber in meiner Sicht gehört die Validierung auf GUI Ebene. So ist es viel benutzerfreundlich und einfacher. Meine ursprüngliche Idee war mit Check Constraints die Validierung abzudecken und implementieren, dass zu grossen Probleme führte und ich lange auf Lösungssuche war. Danach habe ich die ganze Logik selber mit gespeicherte Prozedur implementiert.