swp15-lib

Entwurfsbeschreibung der Softwarestudie

Projektleiter: Christian Blecha

Christian Blecha
07.04.2015

Inhaltsverzeichnis

1 Allgemeines

Ziel der Softwarestudie war es, einen ersten Prototypen zu implementieren, der sich schon im System anmelden und einige rudimentäre Testdaten aus der MySQL-Datenbank auslesen kann. Desweiteren diente die Softwarestudie auch dem Einarbeiten in C++ sowie dem Arbeiten mit Qt. Sie sollte auch aufzeigen, ob und wie die Gruppe sich auf die einzelnen Aufgaben verteilen kann und wie dann die Kommunikation und Koordinierung vonstattengehen kann. Ein weiteres Bestreben der Softwarestudie war es, die Datenbank komplett zu entwerfen und zu implementieren. Die erfolgreiche Verbindung zwischen App und Datenbank zum Abschluss der Softwarestudie setzt somit breits jetzt einen Großteil der Implementierung des Gesamtprojektes um.

2 Produktübersicht

Die Softwarestudie besteht aus einem einfachen Prototypen der App, auf den man direkt ohne größere Probleme aufsetzen kann, der MySQL-Datenbank und der Verbindung der App mit der DB. Der Prototyp ist dabei noch sehr einfach gehalten. Er besitzt nur ein paar Textfelder zum Anzeigen rudimentärer Testdaten. Diese Testdaten werden direkt aus der DB, mit Hilfe der kürzlich implementierten Verbindung der App zur DB, ausgelesen. Die DB wurde komplett entworfen und auf einem Server erstellt. Da unsere Anforderungen nicht mit den Gegebenheiten des uns zur Verfügung stehenden und anfangs geplanten pcai-Servers vereinbar waren, wurde ein eigener Server dafür aufgesetzt und nach unseren Anforderungen eingerichtet. Trotzdem kann die Softwarestudie auf jedem anderen Server mit installiertem MySQL sowie einem Webserver mit PHP-Unterstützung aufgesetzt werden, man muss bloß die entsprechenden Rechte besitzen (speziell in MySQL: root-Rechte).

3 Grundsätzliche Struktur- und Entwurfsprinzipien

Die essentielle Aufgabe in der Softwarestudie bestand darin, eine MySQL-Datenbank anzulegen und eine Verbindung zu dieser erfolgreich aufzubauen. Die Softwarestudie teilt sich dabei, wie in der Produktübersicht schon beschrieben, in 3 Teile: die MySQL-Datenbank, die Verbindung des Smartphones mit dieser und die App an sich.
Die DB wurde mit Hilfe des Datenbankmanagementsystem MySQL implementiert, da dies kostenlos und frei verfügbar ist. Außerdem wurde nicht wie geplant der pcai-Server, mit dessen MySQL-DB genutzt, sondern ein eigener. Diese Überlegung aus dem Projektangebot musste verändert werden, da uns auf dem pcai-Server und dessen MySQL-DB die für unser Projekt erforderlichen Rechte fehlen. Dies tritt dadurch ein, weil wir für jeden neuen Verteiler einen neuen Account in der DB anlegen lassen, so dass jeder Verteiler seinen eigenen Zugang und sein eigenes Passwort hat. Speziell dafür wurde eine „Stored Procedure“ im DBMS angelegt, die es dem Admin erleichtert, einen neuen Verteiler anzulegen (nähere Informationen: siehe Kapitel 4.1 auf Seite 5). Es wurden auch zwei weitere „Stored Procedures“ für das Anlegen eines neuen Admins und das Löschen

eines Verteilers erstellt.
Bei der Verwendung von Qt sind Treiber und Bibliotheken vorhanden, welche eine Verbindung zur MySQL-DB realisierbar machen sollten. Dadurch konnten wir eine Verbindung zu einer lokalen MySQL-DB von der für Desktop kompilierten App herstellen, um einen ersten Test der Funktionalität auszuführen. Leider gelang es uns nicht (nach zahlreichen Arbeitsstunden), diese Verbindung bei der für Android kompilierten App aufzubauen. Grund dafür war eine fehlerhafte Kompilierung der MySQL-Lib für die ARM-Architektur der Android-Geräte. Die Ursachen für diesen Fehler waren u.a. falsche Toolchains, Kompiler, etc. Letztendlich entschieden wir uns gegen die Direktverbindung zwischen der App und MySQL-DB über Qt, da trotz enormen Arbeitsaufwand keine Lösung zustande kam. Unser Ausweichplan lässt nun ein PHP-Skript zwischen App und MySQL-DB vermitteln. Dieses baut die Verbindung zur DB auf, leitet das SQL-Statement von der App an die DB weiter und die Ergebnisse wieder an die App zurück.
Das Design der Softwarestudie besteht im Grunde nur aus dem im Abb. 1 auf Seite 4 sichtbaren Screen. Dieser kann in 2 Teile unterteilt werden: der obere beinhaltet ein paar Buttons die zu ein paar rudimentären weiteren Screens führen. Diese sind die Grundlage für das Gesamtprojekt und sind im Moment rein aus Testgründen über diese Buttons erreichbar. Außerdem existiert ein „Exit“-Button zum Schließen der App. Der zweite Teil beinhaltet im Grunde nur die Ausgabe von zwei Strings, welche die aus der DB ausgelesenen Testdatensätze wiedergeben.

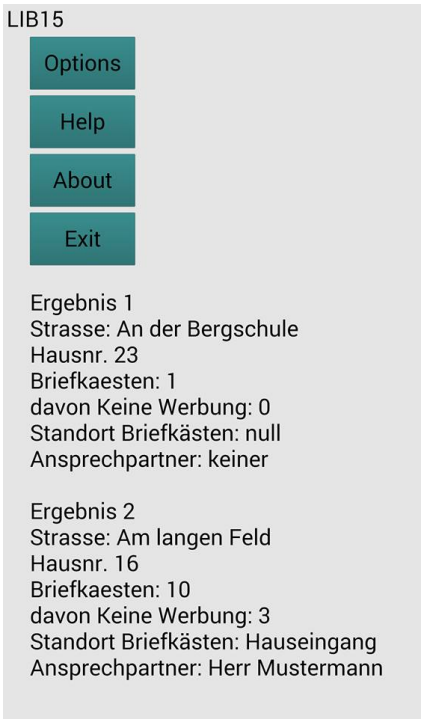

Abbildung 1: Der Screen der Softwarestudie.

4 Struktur- und Entwurfsprinzipien einzelner Pakete

4.1 Die Datenbank

- Das Attribut Materialien & die Tabelle Haeuser_auf_Route:
  Ein Haus kann grundsätzlich auf mehreren Routen mit unterschiedlichen Materialien liegen. Deshalb wird zu jeder Route in der Tabelle „Routenverteilung“ auch gleichzeitig das dazugehörige Material, welches ausgeliefert werden soll, eingetragen. Man muss darauf achten, dass die Materialbezeichnungen immer gleich geschrieben werden & dass die gleiche Wortwahl für die Bezeichnung eines Materials verwendet wird. Ein Haus darf nicht auf zwei Routen mit dem selben Material liegen. Ein Trigger ist hier schwer realisierbar, da es vom DBMS nicht realisierbar ist, verschiedene Schreibweisen für ein Material bzw. verschiedene Wortwahlen in der Bezeichnung für das selbe Material, zu vergleichen. Deshalb muss man selbst bei der Erstellung bzw. dem Import der Routendaten in die Datenbank darauf achten.

- Der Start & das Ende einer Route sind GPS-Referenz und keine Routenpunkt-Referenz:
  Wäre dies nicht gegeben, würde ein Referenz-Zyklus entstehen, welcher die Implementierung der Datenbank auf einem MySQL-Server nicht zulässt, denn das DBMS lässt keine Relationen zu, die sich gegenseitig als Fremdschlüsselbeziehungen besitzen. Trotzdem sollten diese Punkte als besondere Punkte einer Route abgespeichert werden. Der Start & das Ende können auch Routenpunkte auf einer Route sein (siehe Abb. 2 auf Seite 6). Weiterhin können Start und Ende einer Route ebenfalls gleich sein (siehe Abb. 3 auf Seite 6).

- Die Verteilertabelle:
  Diese dient nur noch der Organisation & der Haltung der Passwörter für die Administratoren. Sie ermöglicht zusätzlich einen „PW vergessen“-Support & organisiert die max. Anzahl an Routen die ein Verteiler bearbeiten kann. Diese Anzahl wird durch einen speziellen Trigger, welcher die Anzahl aller Routen für einen Verteiler bei jeder neuen Zuteilung eines Verteilers zu einer Route, überprüft.

- Die Routenverteilungstabelle:
  Da auch hier der Fakt eintritt, dass das DBMS die verschiedenen Schreibweisen einer Materialbezeichnung nicht erkennen kann, ist es notwendig, dass beim Einfügen einer neuen Route selbst darauf geachtet wird, dass eine Route mit einem Material nur einem Verteiler zugeordnet wird.

- Die Briefkastenanzahl & die Anzahl der „Keine Werbung“-Aufkleber bei Häusern:
  Das Neuzählen aller vorhandenen Briefkästen ist in den meisten Fällen nicht notwendig. Man muss nur die Anzahl der „Keine Werbung“-Aufkleber updaten. Dies führt zu einer Reduzierung der Fehler, da die Gesamtanzahl aller Briefkästen & da die Anzahl der „Keine Werbung“-Aufkleber jeweils leichter zählbar ist. Außerdem können so die tatsächlich benötigte Materialanzahlen bestimmt werden.

- Kein SSH:
  Da hierauf vom Productowner nicht extra Wert gelegt wurde, reicht für unsere Zwecke die Zugangsberechtigung über das DBMS.

- Stored Procedure „NewCreateUser“:
  Diese Methode erleichtert, wie bereits erwähnt, das Anlegen eines neuen Verteilers. Man übergibt ihr den Benutzernamen des Verteilers, sein Passwort und die Anzahl an Routen, die er max. zugeteilt bekommen möchte. Die „Stored Procedure“ legt einen neuen Nutzer im DBMS an, speichert die Werte in der Tabelle Verteiler, legt entsprechende Views für den Verteiler an und gibt ihm die entsprechenden Rechte.

- Stored Procedure „DeleteHelp“:
  Diese Methode erleichtert einem das Löschen eines Verteilers, in dem sie den Verteiler und alle zu ihm gehörigen Views und Einträge in der Tabelle Verteiler löscht.

- Stored Procedure „NewCreateAdmin“:
  Diese Methode legt einfach einen neuen Benutzer an und übergibt ihm im selben Zuge die gleichen Rechte wie jeder Admin.

- Die Attribute Karte_schematisch & Karte_Satellit der Tabelle Routenverteilung:
  Da wir uns dafür entschieden haben zwei Varianten der Karte (schematische Darstellung & Satellitenbild) anzubieten, müssen diese auch seperat in der DB abgelegt werden.

Abbildung 2: Routenbeispiel 1

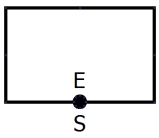

Abbildung 3: Routenbeispiel 2

5 Das Datenmodell

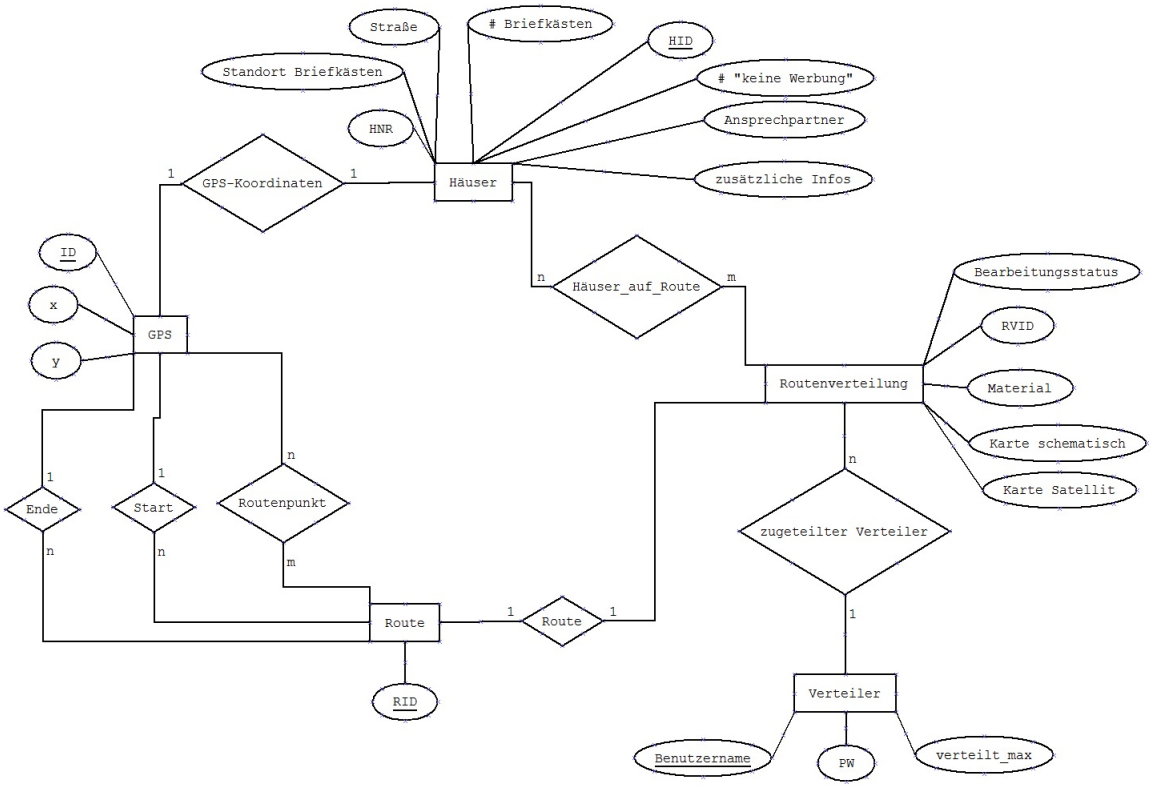

Abbildung 4: Das ER-Modell der zu Grunde liegenden Datenbank.

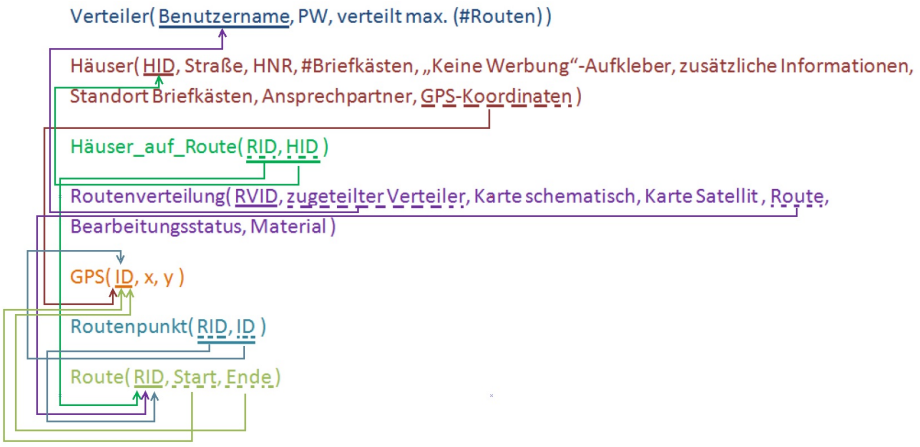

Abbildung 5: Das Relationenmodell der zu Grunde liegenden Datenbank.

6 Das Teskonzept

Für die Softwarestudie wurde die ausgewählte Testumgebung noch nicht eingesetzt. Es wurde nur die Verbindung des Smartphones mit der MySQL-Datenbank rudimentär mit ein paar selbst generierten (bzw. gegebenen) Datensätzen getestet. Diese wurden dann ohne weitere Veränderung oder Bearbeitung auf dem Smartphone angezeigt. Die App wurde bisher nur auf Android basierten Smartphones getestet. Andere Betriebssysteme, wie z.B. Windows Phone, werden dann im Gesamtprojekt mit in die Tests mit einbezogen.

7 Glossar

7.1 Abkürzungsverzeichnis

DB: Datenbank

DBMS: Datenbankmanagementsystem

7.2 Begriffserklärung

Weil schon sehr ausführliche Glossare mit dem 2. und 3. Arbeitsblatt erstellt worden, wird an dieser Stelle auf diese verwiesen.