JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll zur INSY-Übung „Fussballverein“

INSY
4AHITM 2015/16

Yunus Sari

Version 1.0

Note:

Betreuer: Michael Borko

Begonnen am 15. März 2016

Beendet am 06. Juni 2016

Inhaltsverzeichnis

1 Aufgabenstellung

1.1 Aufgabenstellung - E-Learning

Ein österreichischer Fußballverein braucht eine neue Datenbank, um zumindest die Personenverwaltung auf eine professionelle Basis zu stellen. In dieser Datenbank werden folgende Daten verwaltet:

Jede Person ist identifiziert durch eine eindeutige Personalnummer (kurz "persnr"). Außerdem werden zu jeder Person folgende Informationen verwaltet: Vorname ("vname"), Nachname ("nname"), Geschlecht ("geschlecht") und Geburtsdatum ("gebdat"). Für "geschlecht" sind einzig die Eingaben "W", "M" und "N" gültig.

Jede Person, die am Vereinsleben beteiligt ist, gehört zu genau einer der folgenden 4 Kategorien: Angestellter, Vereinsmitglied (kurz "Mitglied"), Spieler oder Trainer. Für all diese Kategorien von Personen müssen zusätzliche Informationen verwaltet werden:
Von jedem Angestellten werden das Gehalt ("gehalt"), die Überstunden ("ueberstunden") und die E-Mail-Adresse ("e_mail") vermerkt.
Für jedes Vereinsmitglied werden ausständige Beitrage ("beitrag") abgespeichert.
Jeder Spieler hat eine Position ("position") (z.B. Tormann, Stürmer, etc.). Außerdem sind Gehalt ("gehalt") sowie Beginn ("von") und Ende ("bis") der Vertragsdauer abzuspeichern. Jeder Spieler ist zumindest einer Mannschaft zugeordnet. Die Spieler innerhalb einer Mannschaft müssen jeweils eine eindeutige Trikot-Nummer ("nummer") zwischen 1 und 99 haben.
Jeder Trainer hat ein Gehalt ("gehalt") sowie Beginn ("von") und Ende ("bis") der Vertragsdauer. Jeder Trainer ist genau einer Mannschaft zugeordnet.

Der Verein hat mehrere Mannschaften. Für jede Mannschaft sind folgende Informationen zu verwalten: eine eindeutige Bezeichnung ("bezeichnung") (wie 1. Mannschaft, Junioren, A-Jugend, ...), die Klasse ("klasse") sowie das Datum des nächsten Spiels ("naechstes_spiel"). Jede Mannschaft hat mindestens 11 Spieler, genau einen Chef-Trainer und genau einen Trainer-Assistenten. Beide sind als Trainer des Vereins registriert. Außerdem hat jede Mannschaft einen Kapitän (der ein Spieler des Vereins ist).

Der Verein hat mehrere Standorte. Jeder Standort ist identifiziert durch eine eindeutige ID ("sid"). Außerdem sind zu jedem Standort folgende Informationen verfügbar: Land ("land"), Postleitzahl ("plz") und Ort ("ort").

An jedem Standort gibt es 1 oder mehrere Fan-Clubs. Jeder Fan-Club hat einen für den jeweiligen Standort eindeutigen Namen ("name"), ein Gründungsdatum ("gegruendet") und einen Obmann ("obmann"), der Vereinsmitglied sein muss. Ein Vereinsmitglied kann von höchstens einem Fan-Club der Obmann sein. Die Fan-Clubs werden von Angestellten des Vereins betreut: Und zwar kann jeder

Angestellte einen Fan-Club jeweils für einen gewissen Zeitraum betreuen. Dieser Zeitraum ist durch Anfangsdatum ("anfang") und Endedatum ("ende") bestimmt. Jeder Angestellte kann 0 oder mehere Fan-Clubs betreuen, und jeder Fanclub kann von einem oder mehreren Angestellten betreut werden.

Der Verein führt Aufzeichnungen über die absolvierten Spiele. Jedes Spiel ist gekennzeichnet durch die Bezeichnung der Mannschaft ("mannschaft") und das Datum ("datum"). Für jedes Spiel werden der Gegner ("gegner") und das Ergebnis ("ergebnis") eingetragen, das einen der Werte "Sieg", "Unentschieden" oder "Niederlage" haben kann. Außerdem werden zu jedem Spiel die beteiligten Spieler abgespeichert, wobei für jeden Spieler die Dauer Einsatzes ("dauer") als Wert zwischen 1 und 90 vermerkt wird.

Aufgabenstellung

1.) Schreiben Sie die nötigen CREATE-Befehle, um die vorgestellten Relationen mittels SQL zu realisieren. Dabei sind folgende Punkte zu beachten:
* Die Datenbank soll keine NULL-Werte enthalten.
* Realisieren Sie folgende Attribute mit fortlaufenden Nummern mit Hilfe von Sequences: "persnr" von Personen und "sid" von Standorten. Für das "persnr"-Attribut sollen nur gerade, 5-stellige Zahlen vergeben werden (d.h.: 10000, 10002, ..., 99998). Für das "sid"-Attribut sollen beliebige positive Zahlen (d.h. 1,2, ...) vergeben werden.
* Falls zwischen 2 Tabellen zyklische FOREIGN KEY Beziehungen herrschen, dann sind diese FOREIGN KEYs auf eine Weise zu definieren, dass es möglich ist, immer weitere Datensätze mittels INSERT in diese Tabellen einzufügen.

2.) Schreiben Sie INSERT-Befehle, um Testdaten für die kreierten Tabellen einzurichten. Jede Tabelle soll zumindest 100000 Zeilen enthalten. Sie dürfen die Wahl der Namen, Bezeichnungen, etc. so einfach wie möglich gestalten, d.h.: Sie müssen nicht "real existierende" Fußballer-Namen, Länder, Städte, etc. wählen. Stattdessen können Sie ruhig 'Spieler 1', 'Spieler 2', 'Land 1', 'Land 2', 'Stadt 1', 'Stadt 2', etc. verwenden. Sie können für die Erstellung der Testdaten auch entsprechende Generatoren verwenden!

3.) Schreiben Sie die nötigen DROP-Befehle, um alle kreierten Datenbankobjekte wieder zu löschen.

Lösen Sie die folgenden Probleme mittels SQL:

S1.) (Fan-Club Betreuung) Wählen Sie "per Hand" die Personalnummer eines Angestellten aus Ihren Testdaten aus. Schreiben Sie eine SQL-Anfrage, die jene Fan-Clubs ermittelt, die dieser Angestellte im Moment nicht betreut. Geben Sie

zu jedem derartigen Fan-Club die Standort-ID und den Namen des Fan-Clubs aus.
Bemerkung: Ein Fan-Club wird von einem Angestellten im Moment nicht betreut, wenn entweder der Angestellte diesen Fan-Club überhaupt nie betreut hat oder wenn das heutige Datum (= sysdate) außerhalb des Betreuungszeitraums liegt. Vergessen Sie nicht, jene Fan-Clubs zu berücksichtigen, die von überhaupt keinem Angestellten betreut werden (dieser Fall sollte zwar laut Datenmodell nicht vorkommen. Die Einhaltung dieser Bedingung wird aber vermutlich vom Datenbanksystem nicht überprüft)!

S2.) (Die eifrigsten Angestellten) Schreiben Sie eine SQL-Anfrage, die den Nachnamen und die Personalnummer jener Angestellten ausgibt, die im Moment sämtliche Fan-Clubs betreuen. Ordnen Sie die Nachnamen alphabetisch. Bemerkung: Passen Sie die Testdaten so an, dass diese Anfrage zumindest zwei Angestellte liefert.

S3.) (Spielereinsätze) Geben Sie für alle Spiele des Jahres 2015 jeweils alle Spieler und die Dauer ihres Einsatzes aus, d.h.: Gesucht sind alle Tupel (mannschaft, datum, vorname, nachname, dauer), mit folgender Eigenschaft:
"mannschaft" ist die Bezeichnung der Mannschaft, die gespielt hat.
"datum" ist das Datum, an dem das Spiel stattfand.
"vorname" und "nachname" beziehen sich auf einen Spieler, der bei diesem Spiel zum Einsatz kam.
"dauer" gibt die Dauer des Einsatzes (in Minuten) dieses Spielers bei diesem Spiel an.

S4.) (Spieler-Ranking) Geben Sie für jeden Spieler den Vornamen und Nachnamen sowie die Gesamtdauer ("gesamtdauer") der von ihm bei Spielen im Jahr 2015 geleisteten Einsätze aus. Vergessen Sie nicht, jene Spieler des Vereins zu berücksichtigen, die im Jahr 2015 bei keinem einzigen Spiel mitgespielt haben (d.h. gesamtdauer = 0). Ordnen Sie die Ausgabe in absteigender Gesamtdauer. Bei Gleichheit der Gesamtdauer sollen die Spieler in alphabetischer Reihenfolge (zuerst des Nachnamen, dann des Vornamen) sortiert werden.

S5.) (Der fleißigste Spieler) Geben Sie den Vornamen und Nachnamen jenes Spielers aus, von dem die unter b) berechnete Gesamtdauer am größten ist, d.h.: dieser Spieler ist bei Spielen im Jahr 2015 insgesamt am längsten im Einsatz gewesen. Falls sich mehrere Spieler den ersten Platz teilen (d.h. sie kommen auf die gleiche Gesamtdauer), dann sollen diese in alphabetischer Reihenfolge (zuerst des Nachnamen, dann des Vornamen) geordnet werden. Der Fall, dass im Jahr 2015 überhaupt kein Spiel stattfand, darf ignoriert werden. Bemerkung: Berücksichtigen Sie bei Ihren Testdaten die Situation, dass sich zumindest 2 Spieler den ersten Platz teilen.

S6.) Schreiben Sie CREATE und DROP Befehle für eine View, die alle Informationen über Trainer aus der Personen- und Trainer-Tabelle

zusammenfügt, d.h.: sowohl die allgemeinen Personendaten (Personalnummer, Vorname, Nachname, Geschlecht und Geburtsdatum) als auch die Trainer-spezifischen Informationen (Gehalt sowie Beginn und Ende der Vertragsdauer). In Summe ist also folgende View erforderlich:
Trainer_view (persnr, vname, nname, geschlecht, gebdat, gehalt, von, bis).

Datenbankclient (Java/C++) und DB-Connector (JDBC/libpqxx):

Schreiben Sie einen Client, der eine Datenbank-Verbindung herstellt. Realisieren Sie eine GUI (JavaFX/Qt), die das einfache Ändern (CRUD) der Spieler des Vereins erlaubt. Verwenden Sie dabei auf jeden Fall eine Tabelle (TableView, QTableView), die auch eine grafische Veränderung der Datensätze erlauben soll.

Ermöglichen Sie die gleichzeitige Verbindung von mehreren Clients auf die Datenbasis. Implementieren Sie dabei eine transaktionelle, gesicherte Erstellung und Änderung von Spielen. Beachten Sie dabei, dass der Spielstand und die Spielzeit der einzelnen Spieler laufend und von mehreren Clients gleichzeitig aktualisiert werden könnte. Stellen Sie für die Eingabe der Spielerzeit und Spielstand eine einfache grafische Möglichkeit zur Verfügung. Verwenden Sie dabei Transaktionen bzw. Locks und entsprechende programmtechnische Mittel um Inkonsistenzen zu vermeiden. Definieren Sie dabei für die einzelnen Informationen (Spielerzeit, Spielstand) eigene Threads.
Informationssysteme und Content Management:
Versuchen Sie in einer kurzen schriftlichen Ausführung anzuführen, welche Informationssysteme im betrieblichen Umfeld genutzt werden und wie diese für die vorliegende Aufgabenstellung eingesetzt werden könnten. Im Bereich der Informationssysteme wurde im Unterricht im Speziellen auf die Thematik der Content Management Systeme eingegangen. Versuchen Sie auch hier kurz zu erläutern welche der kennengelernten CMS für die vorliegende Aufgabenstellung besonders gut eingesetzt werden könnten und warum.
Ausgehend von der vorangegangenen Analyse wählt ein entsprechendes Content Management aus, Implementiert ein passendes Template zu unserem Fußballverein und legt entsprechende Benutzer an die in eurem System erstellen, bearbeiten oder auch nur lesen können sollen. Holen Sie hierbei mittels angelegten Benutzer aus dem Content Management System Datensätze aus der Aufgabe Fußballverein (z.b. Spieler usw.) und zeigt diese entsprechend in eurem CMS an um das Design verdeutlichen sollen.
Setzen Sie das entwickelte Rahmenwerk "Fußballverein", das im Zuge der vorliegenden Arbeit generiert wurde, in Form eines Plug-Ins für das von Ihnen gewählte CMS um.

Abgabe:

Die Abgabe ist bis zum 18. April 2016 um 12:00 auf moodle zu tätigen (fixe Deadline). Es werden ein Protokoll (Metaregeln), die SQL-Files sowie ein

ausführbares JAR-Archiv oder eine plattformunabhängige Build-Umgebung (Make, CMake) erwartet. Sollten nur entsprechende Kompetenzen unter Beweis gestellte werden, ist das Protokoll und die Abgabe auf diese zu beschränken. Die Abgabe wird mit einem Prüfungsgespräch validiert, wobei auf eine eigenständige Lösung geachtet wird. Das Beispiel soll für eine Postgresql 9.5 Umgebung implementiert werden.
Kompetenzzuordnung:
https://docs.google.com/spreadsheets/d/1t2--OpDfwGsXnTskpMCoPz4kD_99L6l4LGPRB7XhgGQ/edit?usp=sharing

Links:
http://www.postgresql.org/docs/current/static/external-interfaces.html
http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784
http://doc.qt.io/qt-5/qtableview.html
https://www.cri.ensmp.fr/people/coelho/datafiller.html#download
(c) dbai Institut; adaptiert bei Michael Borko, Dominik Dolezal und Christoph Brein

1.2 Negative Kompetenzen

- **Eigenschaften und Architekturen von Datenbanksystemen**

können den Begriff "Transaktion" erklären und die Voraussetzungen für eine korrekte Abarbeitung nennen sowie die Problematiken bei parallel auftretenden Transaktionen aufzeigen und diese in Fehlerklassen kategorisieren

- **Datenbankanwendungen**

können Anwendungen mit Datenanbindung entwickeln

- **Anwendungsentwicklung**

können Anwendungssysteme unter Verwendung von Nebenläufigkeit entwickeln

1.3 Geplante Arbeit

Es wird eine Verbindung mit der Datenbank erstellt. Die Datenbank soll auf einer graphischen Oberfläche angezeigt und die Möglichkeit haben diese Datenbank zu modifizieren. Weiter soll die Verwendung von Threads stattfinden.

2 Aufwandschätzung

Arbeitspaket	Geschätzter Aufwand
Installation und Konfiguration der Umgebung	1 h
Erstellung der Grafischen Oberfläche	2 h 30 min
Erstellung der Funktionen	4 h 30 min
Protokoll	3 h
Summe	10 h

3 Vorbereitung/Installation

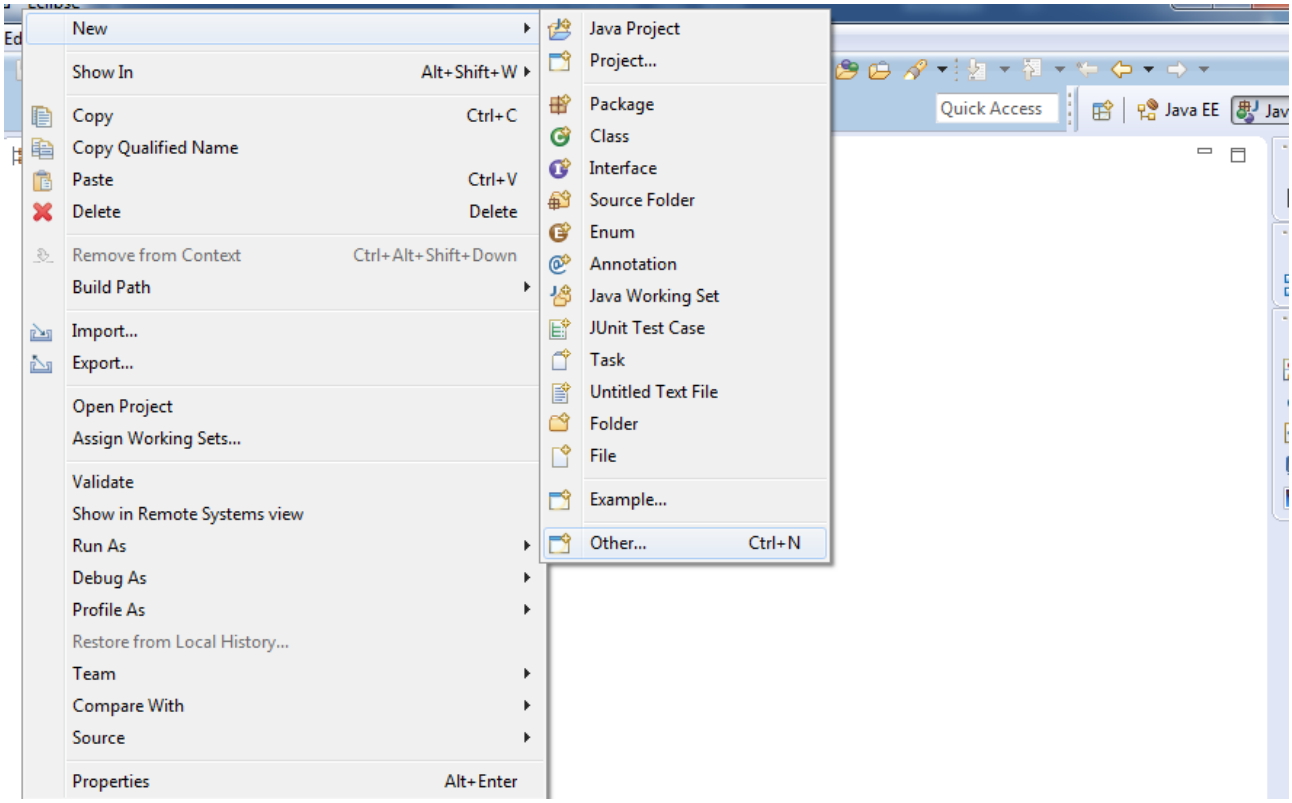
3.1 Voraussetzung

- Ein Server mit der Datenbank Postgresql 9.5
  https://www.postgresql.org/docs/9.5/static/tutorial-install.html
- Java SDK Version 1.8
- Eine Entwicklungsoberfläche mit der jeweiligen Software (In unserem Fall e(fx)clipse).

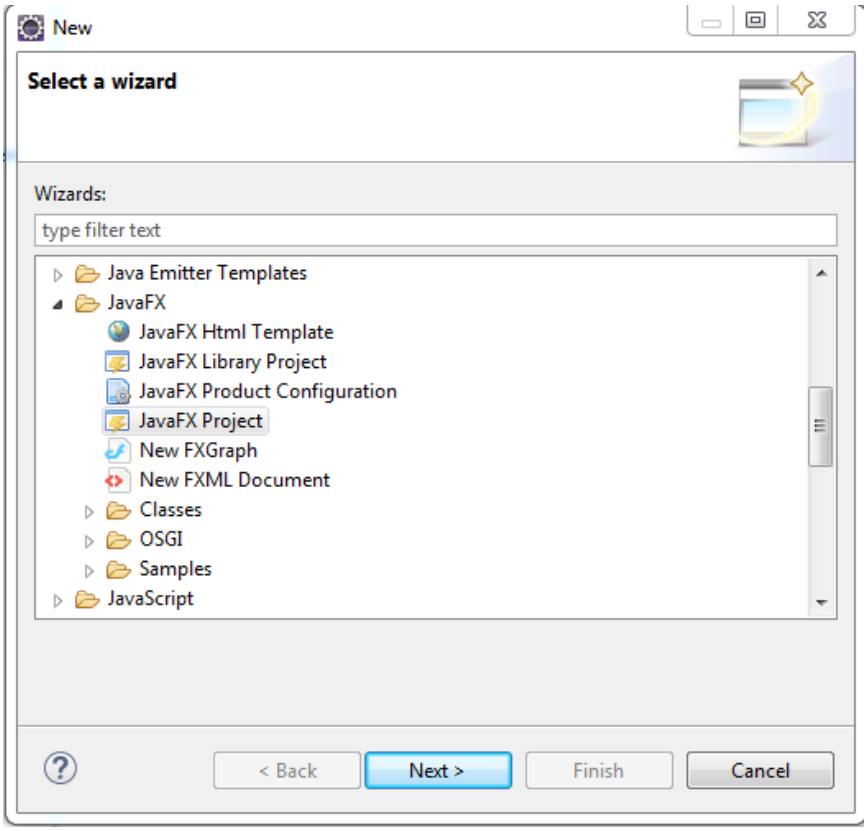
3.2 Installation der Entwicklungsoberfläche

1. Die letzte Version von Eclipse 4.5 SDK wurde unter http://download.eclipse.org/eclipse/downloads/ runtergeladen.
2. Eclipse wurde gestartet
3. Unter dem Menüpunkt „Help" wurde „Install New Software…" angeklickt.
4. „Add" wurde angelickt und unter Name „e(fx)clipse" und unter Location „http://download.eclipse.org/efxclipse/updates-released/2.3.0/site" angegeben.
5. e(fx)clipse – IDE wurde ausgewählt und installiert. („Contact all update sites during install to find required software" muss angeklickt sein")

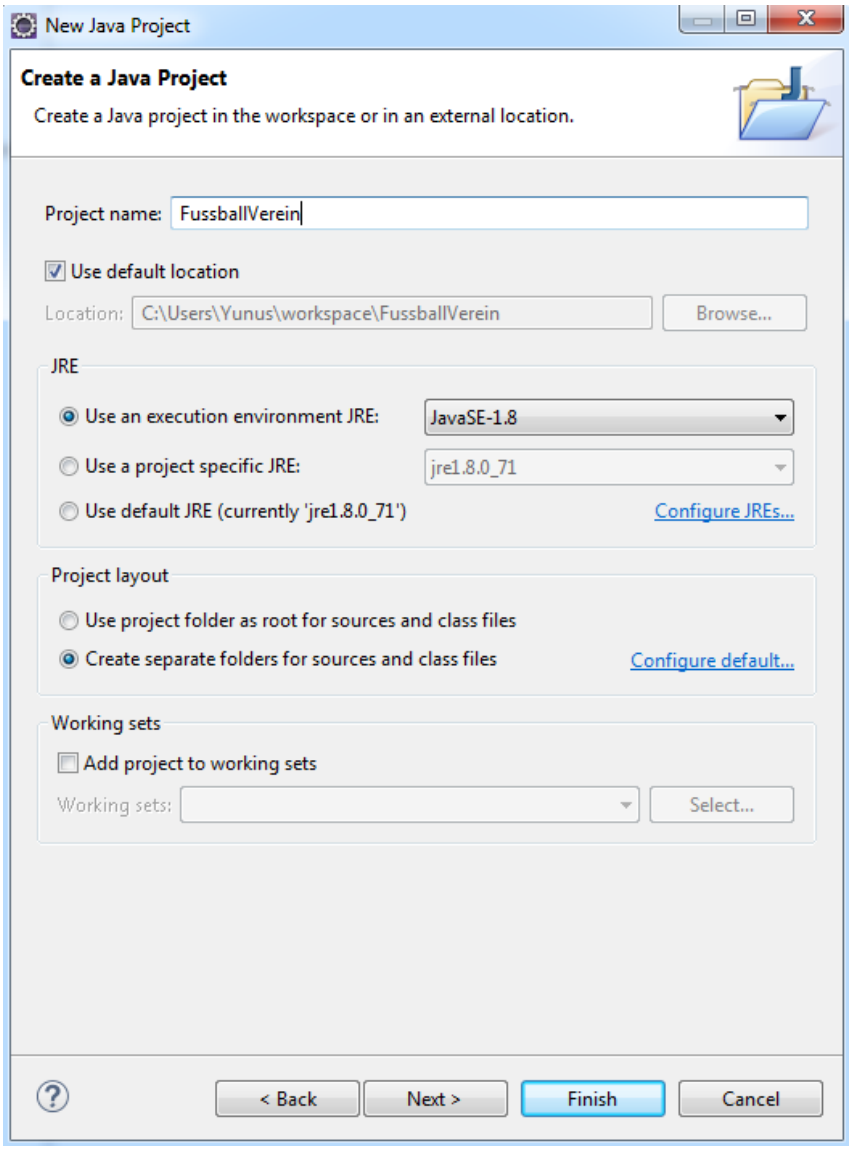
3.3 Erstellung des Projektes



Nach einem Rechtsklick auf dem Projektansichtsmenü erscheint ein Menü. Man geht auf New und klickt dann auf Other.



Man sucht nun den Ordner JavaFX und klickt auf den Punkt JavaFX und klickt dann auf Next.



Man wählt sich einen Namen aus und klickt, falls nicht ausgewählt auf „Use an execution environment JRE" und wählt JavaSE-1.8 aus.

4 Durchführung der Aufgabenstellung

4.1 Ablauf

Am Anfang wird eine Verbindung zur Datenbank erstellt.

Es werden 5 Buttons und 5 Labels erstellt, um die CRUD Funktionen durchzuführen, und, um die Funktionen der Buttons zu schildern.

Diese Buttons werden mit EventHandlern beschmückt, um deren Funktionalität zu programmieren.

Die CRUD-Befehle werden mit Prepared Statements ausgeführt, die von der Aufgabe „Prepared Statement" übernommen worden sind.

Vier von den Buttons werden normal ausgeführt und ein Insert wird in einem Thread ausgeführt. Wenn man diesen Button anklickt, kann man gleichzeitig andere Befehle, wie zum Beispiel einen Read-Befehl ausführen.

4.2 Klassen

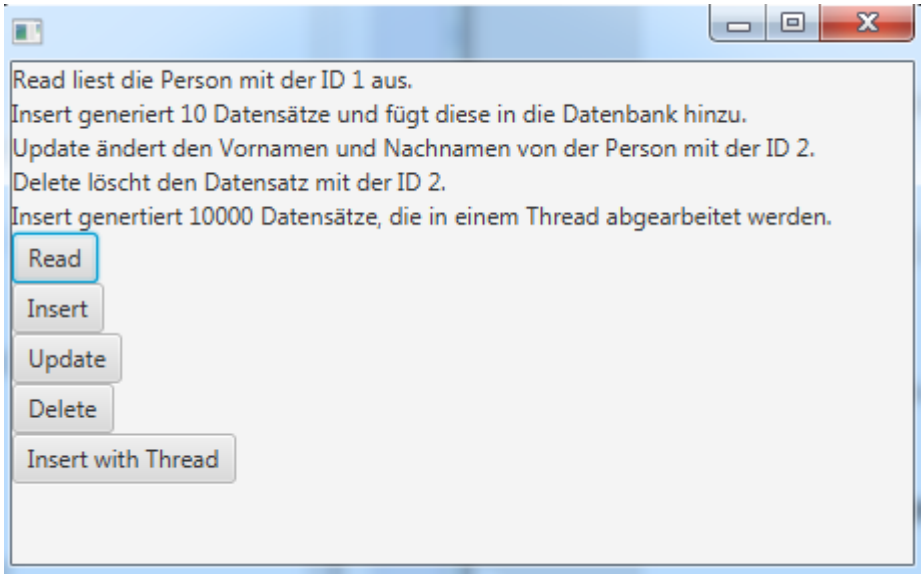
Klasse: Main
Inhalt: Erstellen und Ausführen aller Objekte, Labels, Buttons.

Klasse: CRUD
Inhalt: Alle CRUD Befehle mit der Benutzung von Prepared Statements

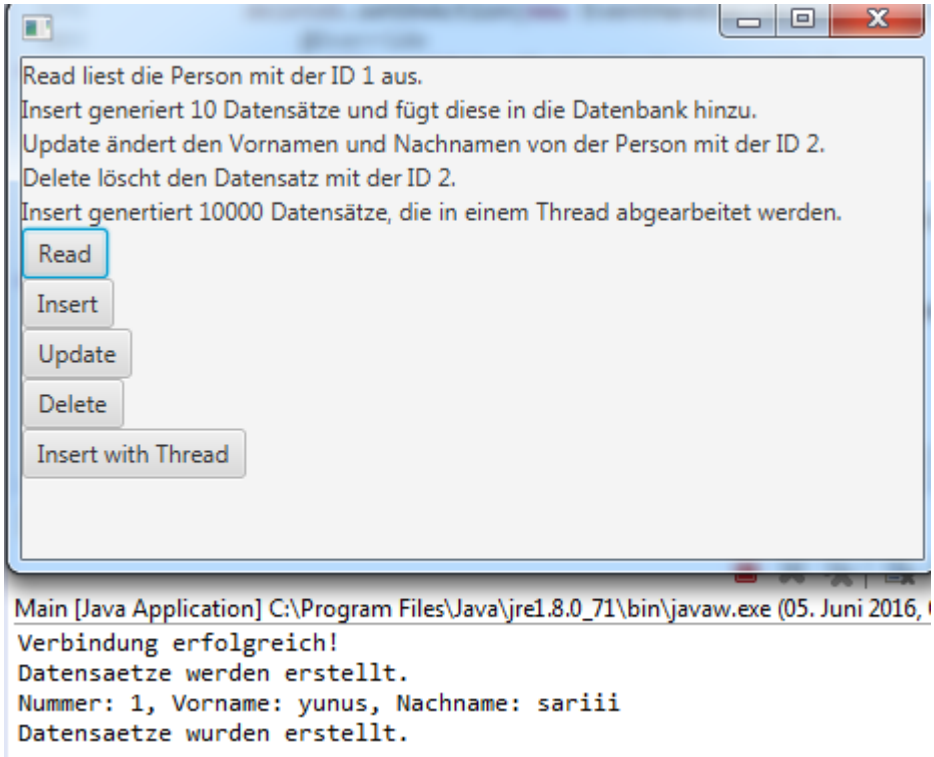
Klasse: DBConenctor
Inhalt: Verbindung mit der Datenbank.

Klasse: InsertThread
Inhalt: Ein Insert mit 10000 Datensätzen. Erbt Runnable für die Thread-Funktion.

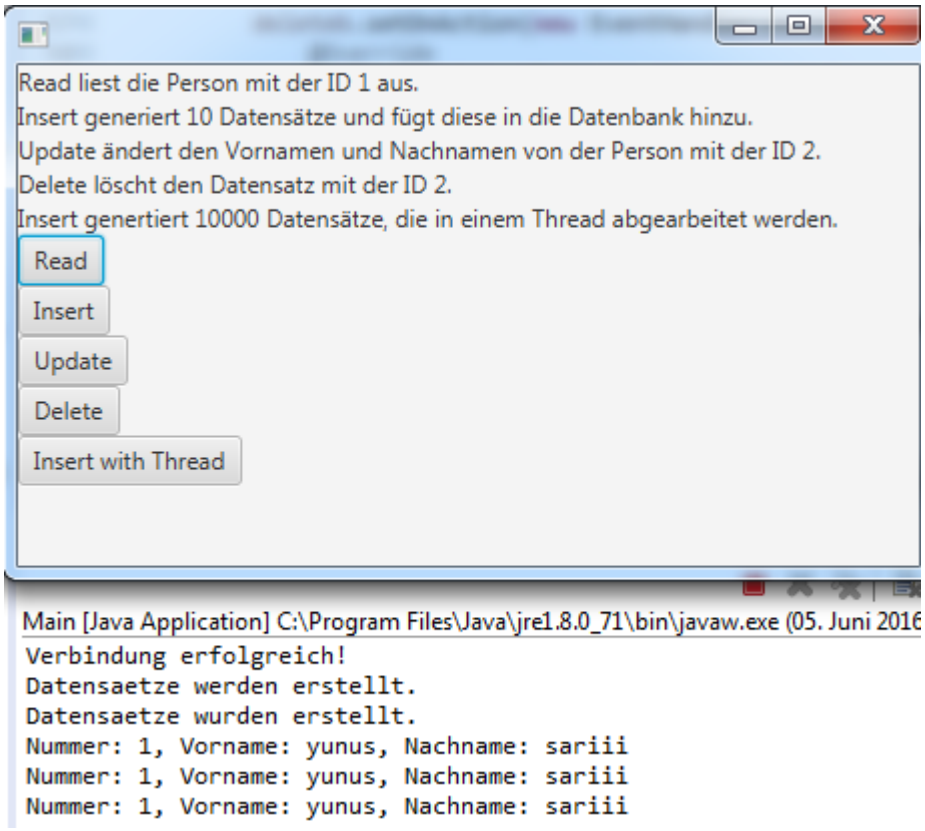
4.3 Demonstration



Die GUI hat 5 Buttons, die die CRUD Befehle ausführen.



Bei Verwendung von „Insert with Thread" möglich ist es möglich, einen Read-Befehl dazwischen zu machen. Jedoch ergibt das keinen Sinn, wenn die ganze Tabelle angezeigt werden soll, wenn diese gerade verändert wird.



Für Testzwecke wurde die Anzahl der generierten Datensätze bei „Insert" auf 1000 gesteigert. Während die Datensätze generiert worden sind, wurden die Klicke „ignoriert" und die Datensätze wurden erstellt. Erst nach der Fertigstellung wurden die Read-Befehle ausgeführt.

4.4 Verbesserungsvorschläge

- Dynamische Abfrage der Datenbankinformationen
- Dynamische Ausgabe der Tabellen
- Manuelles Eintragen von Inserts
- Manuelles Eintragen von Updates
- Löschen mit „Auswählfunktion" aus der Tabellensicht
- Ausgabe der Tabelle und „Statusinformationen" in der GUI
- Exception Handling
- JUnit Testing
- usw …

5 Zeitaufzeichnung

Arbeitspaket	Tatsächlicher Aufwand
Installation und Konfiguration der Umgebung	1 h
Erstellung der Grafischen Oberfläche	1 h 30 min
Erstellung der Funktionen	1 h 30 min
Protokoll	2 h 30 min
Summe	6 h 30 min

5.1 Versionisierung

https://github.com/ysari-tgm/fussballverein

6 Literaturverzeichnis

https://docs.oracle.com/javafx/2/api/

https://docs.oracle.com/javase/8/docs/api/

https://www.postgresql.org/docs/9.5/static/