JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Protokoll

FUSSBALLVEREIN

INSY/SEW
4AHITM 2015/16

Matthias Mischek

Version 1.0

Note:

Betreuer:

Begonnen am 1. April 2016

Beendet am 18. April 2016

Inhaltsverzeichnis

1. Einführung

Diese Übung ist eine Möglichkeit, Kompetenzen aus INSY und SEW nachzubringen. Sollten nur entsprechende Kompetenzen unter Beweis gestellte werden, ist das Protokoll und die Abgabe auf diese zu beschränken. [BOR16]

1.1. Ziele

Ich habe folgende Kompetenzen nicht erfüllt:

Anwendungsentwicklung: können einfache Schnittstellen zur Kommunikation zwischen Anwendungen entwerfen und implementieren.

Ferner wäre es sinnvoll folgende Kompetenz ebenfalls nachzubringen: können umfangreiche Client-Server Anwendungen entwickeln.

[GOG01]

1.2. Voraussetzungen

- Postgresql Kenntnisse
- JavaFx Kenntnisse
- Java Kenntnisse
- Linux Kenntnisse

1.3. Aufgabenstellung

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.

[ELE01]

2. Vorbereitungen

2.1. PostgreSQL installieren

Zuerst musste ich PostgreSQL 9.5 installieren, da ich bis jetzt 9.4.2 benutzte.
Dafür habe ich zuerst mit folgendem Kommando PostgreSQL 9.4.2 gelöscht:

```
apt-get purge <postgres-packagename>
```

Als nächstes habe ich die Verison 9.5 installiert:
Install wget & ca-certificates and trust PostgreSQL's key:

```
apt-get update
apt-get -y install wget ca-certificates
wget --quiet -O - https://www.postgresql.org/media/
keysACCC4CF8.asc | apt-key add -
```

Add the PostgreSQL repository to your sources:

```
echo "deb http://apt.postgresql.org/pub/repos/apt/ $(lsb_release -
cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list
```

Install PostgreSQL 9.5:

```
apt-get update
apt-get -y install postgresql-9.5
```

[GAB01]

2.2. DB erstellen und Daten einfügen

Nun musste ich noch die Fussballverein DB erstellen:
Als superuser von postgres anmelden

```
su postgres
psql
```

Kommandos für die Erstellung eines neuen Users und einer neuen Datenbank:

```
CREATE USER fussballuser WITH LOGIN PASSWORD 'Fusspass';
CREATE DATABASE fussballverein OWNER fussballuser;
GRANT ALL PRIVILEGES ON DATABASE fussballverein TO fussballuser;
```

Verbindung zur Datenbank:

```
psql -h localhost -U fussballuser -W -d fussballverein
```

-h: Adresse oder localhost
-W: Passwort

Tabellen laden:

in das Directory mit den entpackten Dateien wechseln (cd)
von dort Postgres starten (siehe vorheriger Schritt)

```
\i datafiller_result.sql
```

(\i lädt ein externes File)

Konfigurationsänderungen in /etc/postgresql/<versionsnummer>/main:

postgres.conf: nach listen_addresses suchen, # davor weglöschen und den Wert in '' auf * setzen, damit der Zugriff von außen erlaubt ist.

pg_hba.conf (Einzelne spezielle Verbindung zu einer Datenbank erlauben):

#add
```
host <datenbankname> <datenbankuser> <Netzadresse/CIDR> md5
```

(md5 für die Authentifikationsmethode)

Neustarten von Postgres mit:

```
service postgresql restart
```

2.3. Designüberlegung

Ich habe folgende Klassenstruktur: Main.java, Controller.java und View.fxml
In Main.java befindet sich die Mainmethode und das Einlesen des View.fxml.
In Controller.java befinden sich alle wichtigen Methoden, welche von der GUI ausgeführt werden.
Die GUI Elemente werden in der View.fxml definiert.

3. Ergebnisse

3.1. DataFiller

Datafiller wird mit den Kommentaren in einem SQL-File bedient. (- -)

Es können Daten aus einer .list Datei ausgelesen werden und in eine Tabelle eingefügt werden:

```
-- df: name: word=names.list
```

Mit folgenden Kommando kann man definieren wie viele Datensätze generiert werden soll:

```
-- df: mult=1.0
```

(1.0 wird mal 100 gerechnet)

Um festzulegen dass nur ein Name aus der Liste genommen wird, verwendet man folgendes Kommando:

```
-- df: text=name length=1
```

Um zufällige Buchen einzufügen verwendet man folgende Syntax:

```
-- df: pattern=,(M|W|N)'
```

In diesem Fall wird zufällig zwischen M, W und N gewählt.

Um ein zufälliges Datum zu generieren, verwendet man folgendes Kommando:

```
— df: date
```

[DFl01]

3.2. JDBC

Die Datenbankverbindung konnte ich von einem INSY Beispiel verwenden. In meinem Beispiel ist es möglich die Daten für die Verbindung dynamisch über die GUI einzugeben.

```
public void connect(ActionEvent event) throws SQLException {

	data = FXCollections.observableArrayList();

	// Datenquelle erzeugen und konfigurieren
	ds = new PGSimpleDataSource();
	ds.setServerName(ip.getText());
	ds.setDatabaseName(dbName.getText());
	ds.setUser(user.getText());
	ds.setPassword(passwort.getText());
	// Verbindung herstellen
	try

	{
		this.con = ds.getConnection(); // Verbinden


		updatePane.setDisable(false);
		anzeigePane.setDisable(false);
		insertPane.setDisable(false);
		updateSpielT.setDisable(false);
		anzeigeSpielT.setDisable(false);
		status.setText("Erfolgreich Verbunden!");

		// Exception handling
	} catch (PSQLException e) {
		System.err.println("Error");
		status.setText("Bitter erneut versuchen.");

	} catch (Exception se) {
		con.rollback();
		System.err.println("Error");
		status.setText("Bitte erneut versuchen.");

	}

}
```

3.3. Anzeigen der DB

Zuerst habe ich ein Statement erstellt und die passende Query eingesetzt. Um die Tabellen anzuzeigen habe ich eine TableView verwendet. Damit die Tabellen nicht doppelt eingefügt werden, hab ich auch eine aktualisieren() Methode, welche die eingefügten Tabellen aus der

```
con = ds.getConnection();
// Abfrage vorbereiten und ausführen
Statement st = con.createStatement();
con.setAutoCommit(false);
ResultSet rs = st.executeQuery("select * from Spieler");


for (int i = 0; i < rs.getMetaData().getColumnCount(); i++) {
       final int j = i;
       TableColumn col = new TableColumn(rs.getMetaData().getColumnName(i + 1));
       col.setCellValueFactory(
       new Callback<CellDataFeatures<ObservableList, String>,
       ObservableValue<String>>() {
       public ObservableValue<String> call(CellDataFeatures<ObservableList,
       String> param) {
       return new SimpleStringProperty(param.getValue().get(j).toString());

              }
       });

       anzeige.getColumns().addAll(col);}

// Ergebnisse verarbeiten
while (rs.next()) { // Cursor bewegen
       ObservableList<String> row = FXCollections.observableArrayList();
       for (int i = 1; i <= rs.getMetaData().getColumnCount(); i++) {
              // Iterate Column
              row.add(rs.getString(i));  }

       data.add(row);
}
updateBox.setItems(data);
anzeige.setItems(data);
con.commit();
```

TableView löschen.
Für das Einfügen in die TableView habe ich mir Hilfe aus folgendem Forum geholt: http://stackoverflow.com/questions/36637532/javafx-using-dynamic-table-view-with-database-editable-cell-problems

3.4. UPDATE/INSERT

Die Methoden für Update und Insert habe ich eigentlich immer noch dem selben Prinzip geschrieben. Ich verwende die vorhandene Verbindung und erstelle ein neues Statement.

```
Connection con = ds.getConnection();
// Abfrage vorbereiten und ausführen
Statement st = con.createStatement();
con.setAutoCommit(false);
st.executeUpdate("UPDATE Spieler SET position=\'" + pos.getText()
             + "\', gehalt=\'" + cash.getText() + "\', von=\'" + von.getText()
+"\', bis=\'" + bis.getText() + "\' WHERE persnr=\'" + updatear[0] + "\'");
```

Update:

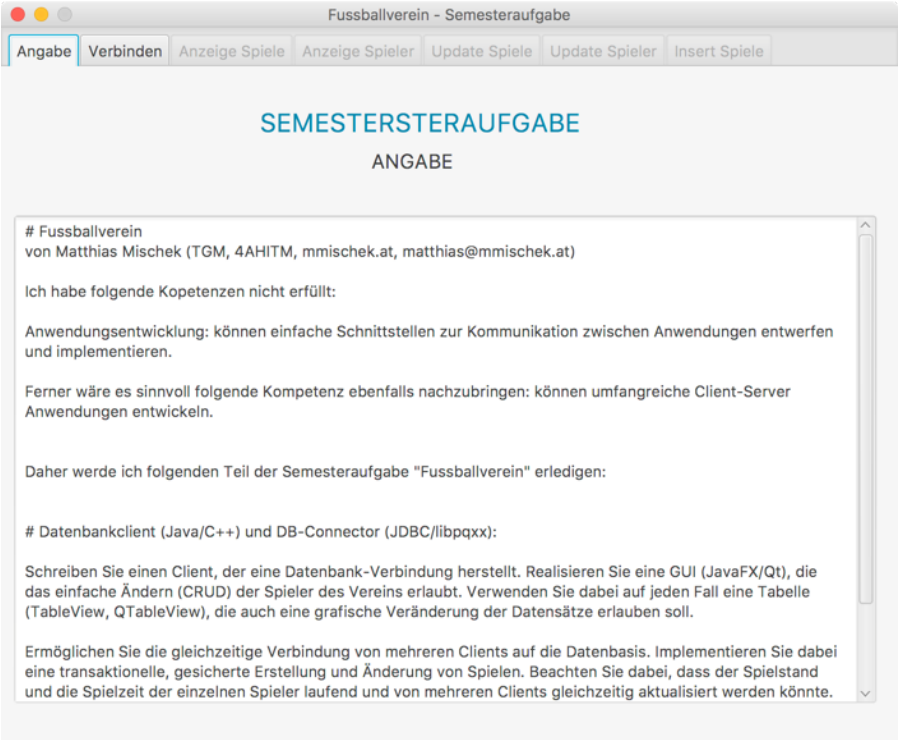
Um einen Datensatz auszuwählen und dann zu verändern, habe ich eine ComboBox verwendet. Die Daten bekommt die ComboBox über die Methode anzeige().
Wenn man einen Datensatz ausgewählt hat und die Methode auswahl() aufruft, bekommen die Textfelder den dazugehörigen String.
Sobald man dann auf update klickt wird die oben beschriebene Methode aufgerufen und das Statement ausgeführt. Hierbei habe ich auf Fehlerfälle geachtet und dementsprechend Exception gecatcht.

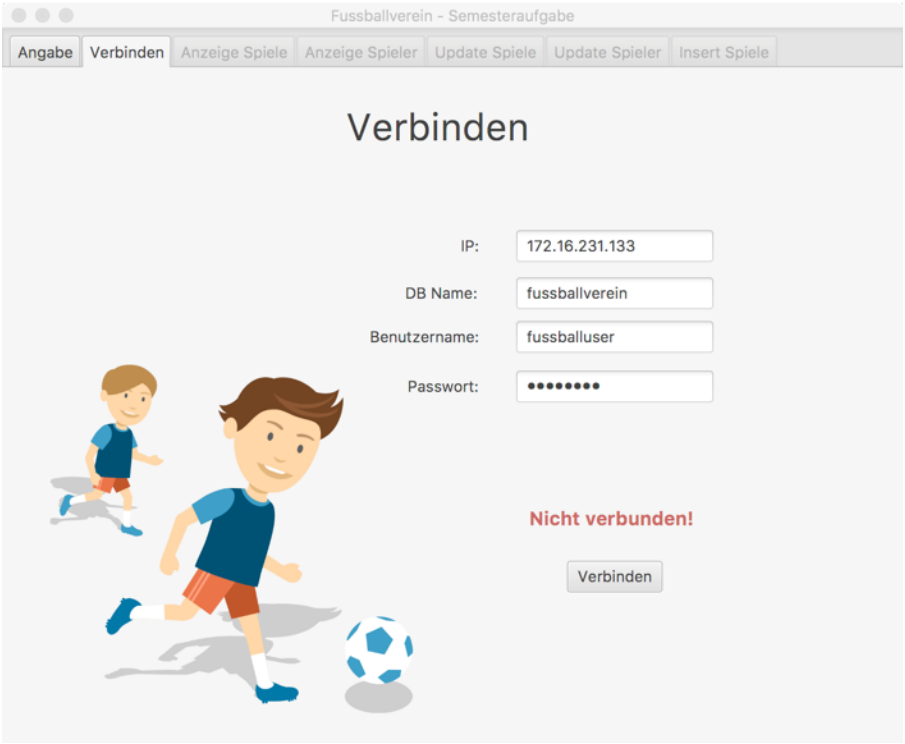
Insert:

Im Prinzip funktioniert die Insert Methode gleich, nur das Statement ist anders.

3.5. Grafische Oberfläche

Wenn das Java Programm gestartet wird, wird zuerst ein Tab mit der Aufgabenstellung angezeigt.

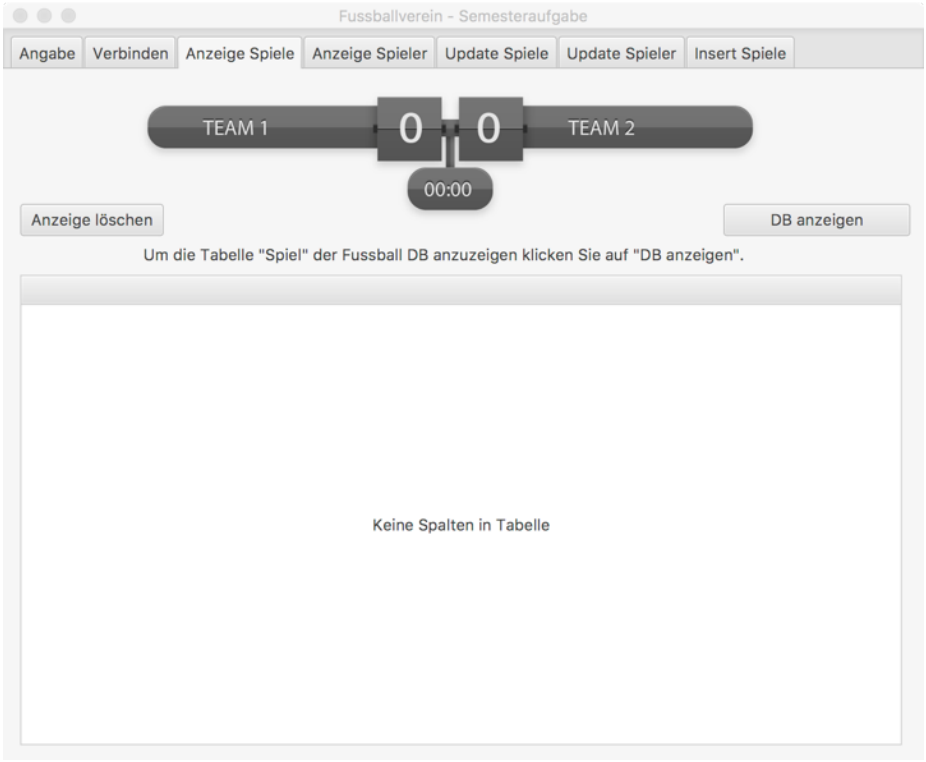




In dem Tab „Verbinden“ ist es möglich eine Verbindung zur Datenbank herzustellen. Es wird die IP, DP Name, Benutzername und das Passwort abgefragt.

Sobald man sich über den Button „Verbinden“ verbunden hat, ändert sich der rote Text auf „erfolgreich Verbunden“.

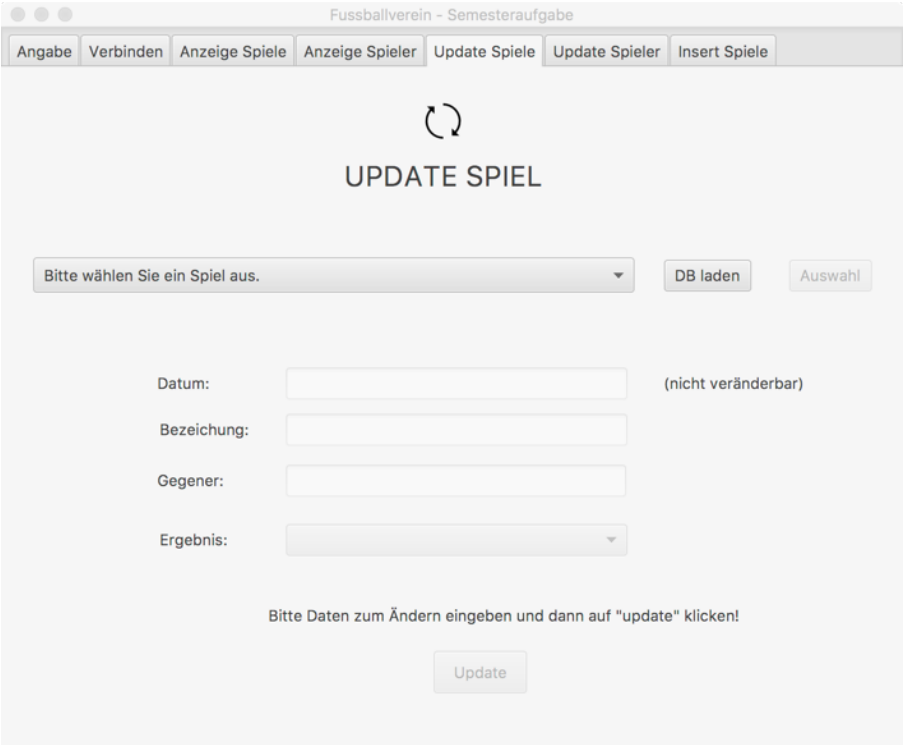
Wenn man verbunden ist, werden die weiteren Tabs aktiviert.



In dem Tab „Anzeige Spiele“ ist es möglich die Tabelle „Spiel“ in einer TableView anzeigen zu lassen.

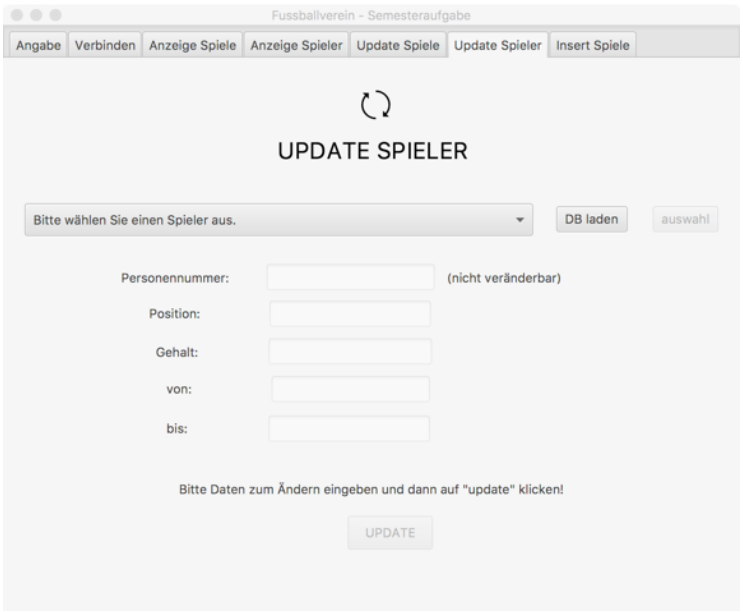
Damit man die Daten nicht doppelt in die TableView laden kann, wird der Button deaktiviert und erst nach dem Betätigen von „Anzeige löschen“ reaktiviert.

Genau das selbe Prinzip verwende ich in dem „Anzeige Spieler“ Tab. Der einzige Unterschied ist, dass eine andere Tabelle angezeigt wird.



In dem Tab „Update Spiele“ ist es möglich Datensätze aus der Tabelle „Spiel“ zu ändern. Um aus der ComboBox einen Datensatz auswählen zu können, muss man zuerst auf den Button „DB laden“ klicken. Falls davor die Datenbank bereits in dem Tab Anzeige geladen wurde, muss der Button nicht betätigt werden.

Wenn ein Datensatz ausgewählt wurde, kann man mit dem Button „Auswahl“ bestätigen und die Textfelder werden mit den jeweiligen Daten gefüllt. Jetzt ist es möglich über die Textfelder neue Werte einzugeben. Diese werden mit dem Button „update“ in die Datenbank angetragen.



Das gleiche Prinzip in dem Tab „Update Spieler“

In dem letzten Tab ist es möglich Spiele hinzuzufügen. Die Eingabe müssen einer vorgegebenen Form entsprechen. In welcher Form man die Daten eingeben soll, erkennt man an dem grauen Text.
Falls die Eingabe nicht dieser Form entspricht, wird ein Fehler ausgegeben.

Mit dem Button „hinzufügen“ wird der Datensatz in die Datenbank hinzugefügt.

4. Management

4.1. Zeitmanagement

Arbeitsbereich	Zeitlicher Aufwand (in h)
PostgreSQL Installation & Konfiguration	2
Designüberlegung	2
fxml Interface	11
JDBC	1,5
CRUD	21
Datafiller (Zusammenarbeit mit Benedikt Berger)	3
Gesamtanzahl	**40,5**

4.2. Versionierung

Ich habe github.com für die Versionierung meiner Aufgabe verwendet.

Link zu meinem Repository: https://github.com/mmischek/Fussballverein.git

Die Datenbank wurde in Zusammenarbeit mit Benedikt Berger erstellt.

5. Quellen

[GOG01] SEMESTERAUFGABE. GOOGLE DRIVE [ONLINE] AVAILABLE AT: HTTPS://DOCS.GOOGLE.COM/SPREADSHEETS/D/1T2--OPDFWGSXNTSKPMCOPZ4KD_99L6L4LGPRB7XHGGQ/EDIT?PREF=2&PLI=1#GID=1024423784 [ABGERUFEN AM 18. APRIL 2016]

[BOR16] MICHAEL BORKO (TGM MOODLE) [ONLINE] AVAILABLE AT: HTTPS://ELEARNING.TGM.AC.AT/MOD/ASSIGN/VIEW.PHP?ID=48203 [ABGERUFEN AM 18. APRIL 2016]

[ELE01] SEMESTERAUFGABE - "FUSSBALLVEREIN". TGM MOODLE [ONLINE] AVAILABLE AT: HTTPS://ELEARNING.TGM.AC.AT/MOD/ASSIGN/VIEW.PHP?ID=48203 [ABGERUFEN AM 18. APRIL 2016]

[DFI01] DATAFILLER [ONLINE] AVAILABLE AT: HTTPS://WWW.CRI.ENSMP.FR/PEOPLE/COELHO/DATAFILLER.HTML [ABGERUFEN AM 18. APRIL 2016]

[STA01] JAVAFX: USING DYNAMIC TABLE VIEW WITH DATABASE [ONLINE] AVAILABLE AT: HTTP://STACKOVERFLOW.COM/QUESTIONS/36637532/JAVAFX-USING-DYNAMIC-TABLE-VIEW-WITH-DATABASE-EDITABLE-CELL-PROBLEMS [ABGERUFEN AM 18. APRIL 2016]

[GAB01] POSTGRESQL 9.5 INSTALLATION INSTRUCTIONS [ONLINE] AVAILABLE AT: HTTP://WWW.GAB.LC/ARTICLES/INSTALL_POSTGRESQL_9-5_DEBIAN_UBUNTU [ABGERUFEN AM 18. APRIL 2016]