
Meilenstein 3

Dokumentation

Migration Relational – NoSQL



Team 5

Nikola Babic, 1501205

Cordula Eggerth, 0750881

Denise Gall, 1347758

Gregor Langner, 1502605

BSc Wirtschaftsinformatik

VU Information Management & Systems Engineering

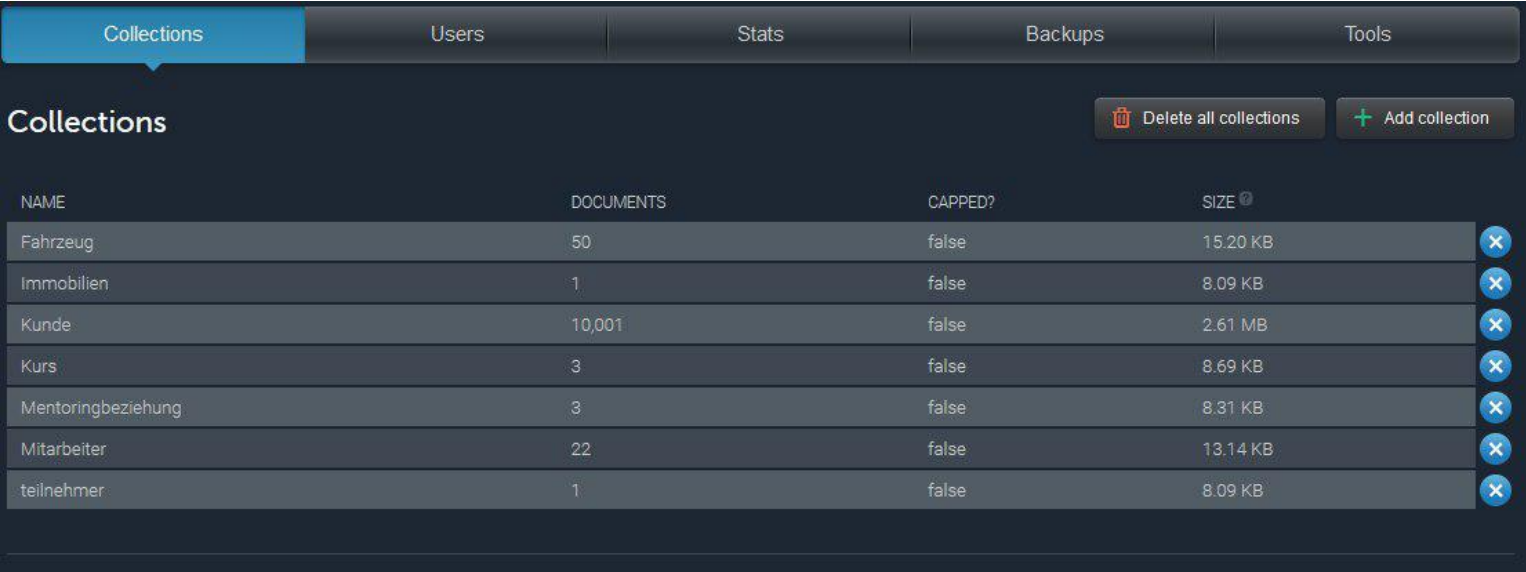
SoSe 2017

Inhaltsverzeichnis

1. NoSQL Datenbankdesign	3
2. Datenmigration	8
3. Erstellung des Websystems (basierend auf NoSQL - MongoDB).....	8
4. Anhang: Klassendiagramm	10
5. Anhang: Login-Hilfe	11

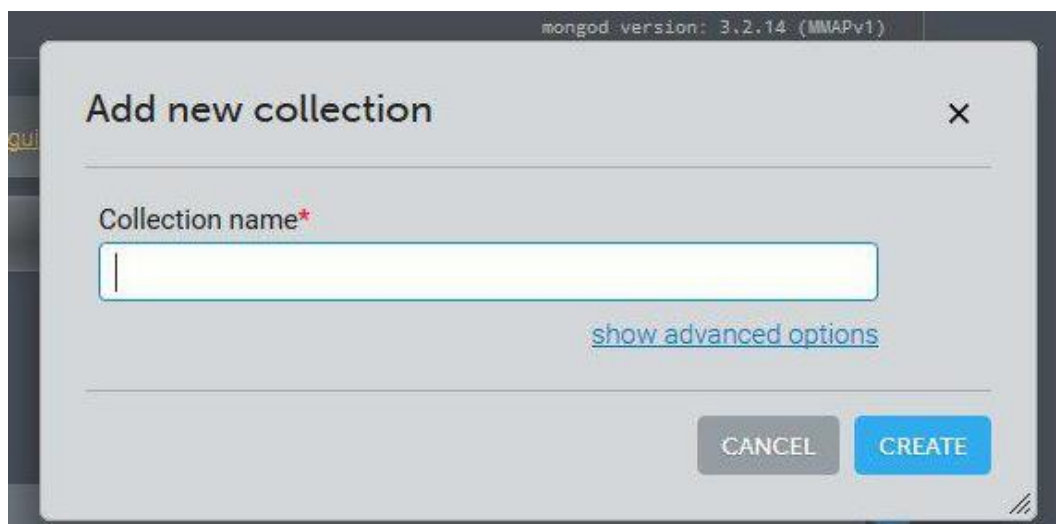
1. NoSQL Datenbankdesign

Die NoSQL Datenbank wurde auf mlab.com angelegt und es wurden Collections für die im Java-Project verwendeten Klassen erstellt. Der untenstehende Screenshot zeigt den Bereich der Collections, der online angelegt wurde, und zeigt eine Auflistung der Collections: Fahrzeug, Immobilien, Kurs, Kunde, Mentoringbeziehung, Mitarbeiter und teilnehmer. Darin ist auch ersichtlich, wie viele Documents (also Objekte) bereits von den jeweiligen Klassen in der MongoDB vorhanden sind und wie groß deren Speicherplatzverbrauch ist (siehe Tabellenspalte „size“). Die Collections können einfach über die Buttons „Delete“ oder „Add“ gelöscht oder neue hinzugefügt werden.



NAME	DOCUMENTS	CAPPED?	SIZE
Fahrzeug	50	false	15.20 KB
Immobilien	1	false	8.09 KB
Kunde	10,001	false	2.61 MB
Kurs	3	false	8.69 KB
Mentoringbeziehung	3	false	8.31 KB
Mitarbeiter	22	false	13.14 KB
teilnehmer	1	false	8.09 KB

Die folgende Abbildung zeigt die Ansicht, wenn eine neue Collection angelegt wird:



mongod version: 3.2.14 (MMAPv1)

Add new collection

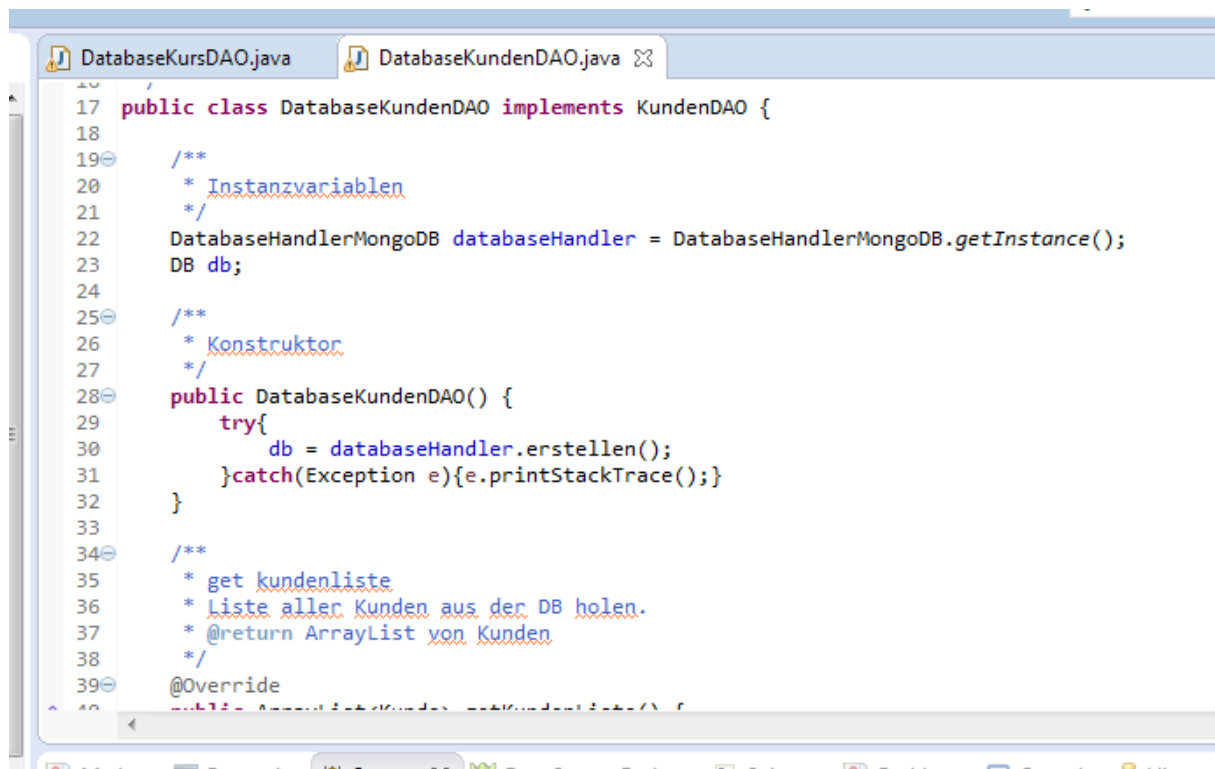
Collection name*

[show advanced options](#)

CANCEL CREATE

Die Schnittstelle zwischen Java-Project und MongoDB befindet sich im Package repository. Dort wurden der DatabaseHandler und die jeweiligen DatabaseDAO Klassen pro Objekt angepasst, sodass man Objekte mit dem entsprechenden Java-Code in die MongoDB hinzufügen, löschen, updaten oder abfragen kann, was im Folgenden anhand von Beispielen der Klasse DatabaseKundenDAO dargestellt wird, die Methoden, mit denen Aktionen in der Datenbank ausgeführt werden, beinhaltet.

Die folgende Abbildung zeigt den *DatabaseHandler*, der auf *die Verwendung der MongoDB* angepasst wurde:



```
17 public class DatabaseKundenDAO implements KundenDAO {
18
19     /**
20      * Instanzvariablen
21      */
22     DatabaseHandlerMongoDB databaseHandler = DatabaseHandlerMongoDB.getInstance();
23     DB db;
24
25     /**
26      * Konstruktor
27      */
28     public DatabaseKundenDAO() {
29         try{
30             db = databaseHandler.erstellen();
31         }catch(Exception e){e.printStackTrace();}
32     }
33
34     /**
35      * get kundenliste
36      * Liste aller Kunden aus der DB holen.
37      * @return ArrayList von Kunden
38      */
39     @Override
40     public ArrayList<Kunden> getKundenliste() {
```

Die folgende Abbildung zeigt die Methode *getKundenListe*, mit der eine Liste aller Kunden (Documents) aus der Collection Kunde der MongoDB geholt wird:

```
@Override
public ArrayList<Kunde> getKundenListe() {

    ArrayList<Kunde> kundenList = new ArrayList<Kunde>();
    DBCollection kundencoll = db.getCollection("Kunde");
    DBCursor cursor = kundencoll.find();

    while(cursor.hasNext()) {
        // System.out.println(cursor.next());

        BasicDBObject kundeObj = (BasicDBObject) cursor.next();

        boolean active = Boolean.parseBoolean(kundeObj.getString("active"));

        try{

            Kunde kunde=new Kunde(kundeObj.getInt("kundenid"),kundeObj.getString("vorname"),
                                   kundeObj.getString("nachname"),kundeObj.getInt("iban"),kundeObj.getString("bic"),
                                   kundeObj.getString("username"),kundeObj.getString("passw"),active);

            kundenList.add(kunde);
        }catch(Exception e){
            System.out.println(e.getMessage());
        }

        // CHECK
        for (Kunde kunde : kundenList) {
            System.out.println(kunde.getNachname());
        }

    }

}
```

Die folgende Abbildung zeigt die Methode *getKundeById(int id)*, mit der ein bestimmter Kunde (Documents) anhand der ID aus der Collection Kunde der MongoDB geholt wird:

```
@Override
public Kunde getKundeById(int id){

    Kunde suchKunde = null;

    DBCollection kundencoll = db.getCollection("Kunde");

    BasicDBObject idObject = new BasicDBObject("id", id);

    DBCursor cursor = kundencoll.find(idObject);
    for (int i=0; i<cursor.size(); i++) {
        BasicDBObject kundeObj = (BasicDBObject) cursor.next();
        if(id == kundeObj.getInt("kundenid")){
            try{

                boolean active = Boolean.parseBoolean(kundeObj.getString("active"));
                suchKunde=new Kunde(kundeObj.getInt("kundenid"),kundeObj.getString("vorname"),
                                   kundeObj.getString("nachname"),kundeObj.getInt("iban"),kundeObj.getString("bic"),
                                   kundeObj.getString("username"),kundeObj.getString("passw"),active);

            } catch(Exception e){
                System.out.println(e.getMessage());
            }
        }
    }

    return suchKunde;

}
```

Die folgende Abbildung zeigt die Methode ***deleteKunde(int id)***, mit der ein bestimmter Kunde (Documents) anhand der ID aus der Collection Kunde der MongoDB gelöscht wird:

```
@Override
public void deleteKunde(int id) {

    DBCollection kunde = db.getCollection("Kunde");
    BasicDBObject query = new BasicDBObject("kundenid", id);
    DBCursor cursor = kunde.find(query);
    for(int i=0;i<cursor.size();i++){
        BasicDBObject act = (BasicDBObject) cursor.next();
        if(id == act.getInt("kundenid")){
            kunde.remove(act);
        }
    }
}
```

Die folgende Abbildung zeigt die Methode ***addKunde(Kunde kunde)***, mit der ein bestimmter Kunde (Documents) als Objekt übergeben wird und in die Collection Kunde der MongoDB hinzugefügt wird:

```
@Override
public void addKunde(Kunde kunde) {

    DBCollection kundencoll = db.getCollection("Kunde");
    DBCursor cursor = kundencoll.find();

    // System.out.println(kundencoll.count());
    long next = kundencoll.count();
    next = next + 1;

    BasicDBObject doc = new BasicDBObject();
    doc.put("kundenid", next);
    doc.put("vorname", kunde.getVorname());
    doc.put("nachname", kunde.getNachname());
    doc.put("iban", kunde.getIban());
    doc.put("bic", kunde.getBic());
    doc.put("username", kunde.getUsername());
    doc.put("passw", kunde.getPassword());
    doc.put("active", 1);

    kundencoll.insert(doc);
}
```

Die folgende Abbildung zeigt die Methode *update(Kunde kunde)*, mit der ein bestimmter Kunde in der MongoDB aktualisiert wird:

```
@Override
public void updateKunde(Kunde kunde) {

    DBCollection kundencoll = db.getCollection("Kunde");
    BasicDBObject object = new BasicDBObject("kundenid", kunde.getId());
    DBCursor cursor = kundencoll.find(object);

    if(cursor.size()>0){
        for(int i=0; i<cursor.size(); i++){
            BasicDBObject next = (BasicDBObject) cursor.next();
            if(kunde.getId() == next.getInt("kundenid")){

                BasicDBObject update = new BasicDBObject();

                update.append("kundenid", kunde.getId());
                update.append("vorname", kunde.getVorname());
                update.append("nachname", kunde.getNachname());
                update.append("iban", kunde.getIban());
                update.append("bic", kunde.getBic());
                update.append("username", kunde.getUsername());
                update.append("bic", kunde.getBic());
                update.append("passw", kunde.getPassword());
                update.append("active", 1);

                kundencoll.update(new BasicDBObject().append("kundenid", kunde.getId()), update);
            }
        }
    }
}
```

2. Datenmigration

Die Collections wurde über mlab.com in der verwendeten MongoDB angelegt und die Documents (bzw. Objekte) werden über die Klassen DatabaseKundenDAO, DatabaseMitarbeiterDAO, DatabaseFahrzeugDAO usw. jeweils eingefügt in die Datenbank bzw Update oder Delete Operationen finden darüber statt.

Es wurden außerdem Testdaten generiert und in die MongoDB eingefügt, damit genug Daten vorhanden sind, um die Funktionalitäten der Website auszuprobieren.

3. Erstellung des Websystems (basierend auf NoSQL - MongoDB)

Das untenstehende Use-Case-Diagramm zeigt die realisierten Funktionalitäten des Informationssystems für die Fahrschule CarGo.¹ Die *Aufteilung der Use-Cases* unter den Team-Mitgliedern wurde mittels *Farbcodierung* gekennzeichnet, und in einer zusätzlichen Notiz im Diagramm wurden gemeinsame Bearbeitungen eines Use-Case sowie die Realisierung von Login und Logout vermerkt. Für *nähere Informationen* zu den *Aufgabenbereichen der einzelnen Teammitglieder* und wer, wann, woran gearbeitet hat, siehe *M3_Team5_TeamArbeitsprotokoll.pdf* in der Abgabe.

Die Datenbank ist nur mittels SSH-Verbindung zu erreichen und der Webserver wird mit selbst-signiertem Zertifikat abgesichert (mit https).

Das Projekt wurde auf dem Tomcat-Server deployed:

https://tomcat01lab.cs.univie.ac.at:31354/ise_Projekt/

Zur Farbcodierung des Use-Case-Diagramms auf der nächsten Seite:

- Nikola Babic: grün
- Cordula Eggerth: gelb
- Denise Gall: rot
- Gregor Langner: blau

¹ Die Funktionalitäten des Login und Logout wurden nicht im Diagramm inkludiert, da sie lt. SWE VO (Prof. Benkner) nicht als „Use-Cases im engeren Sinn“ zu zählen sind.



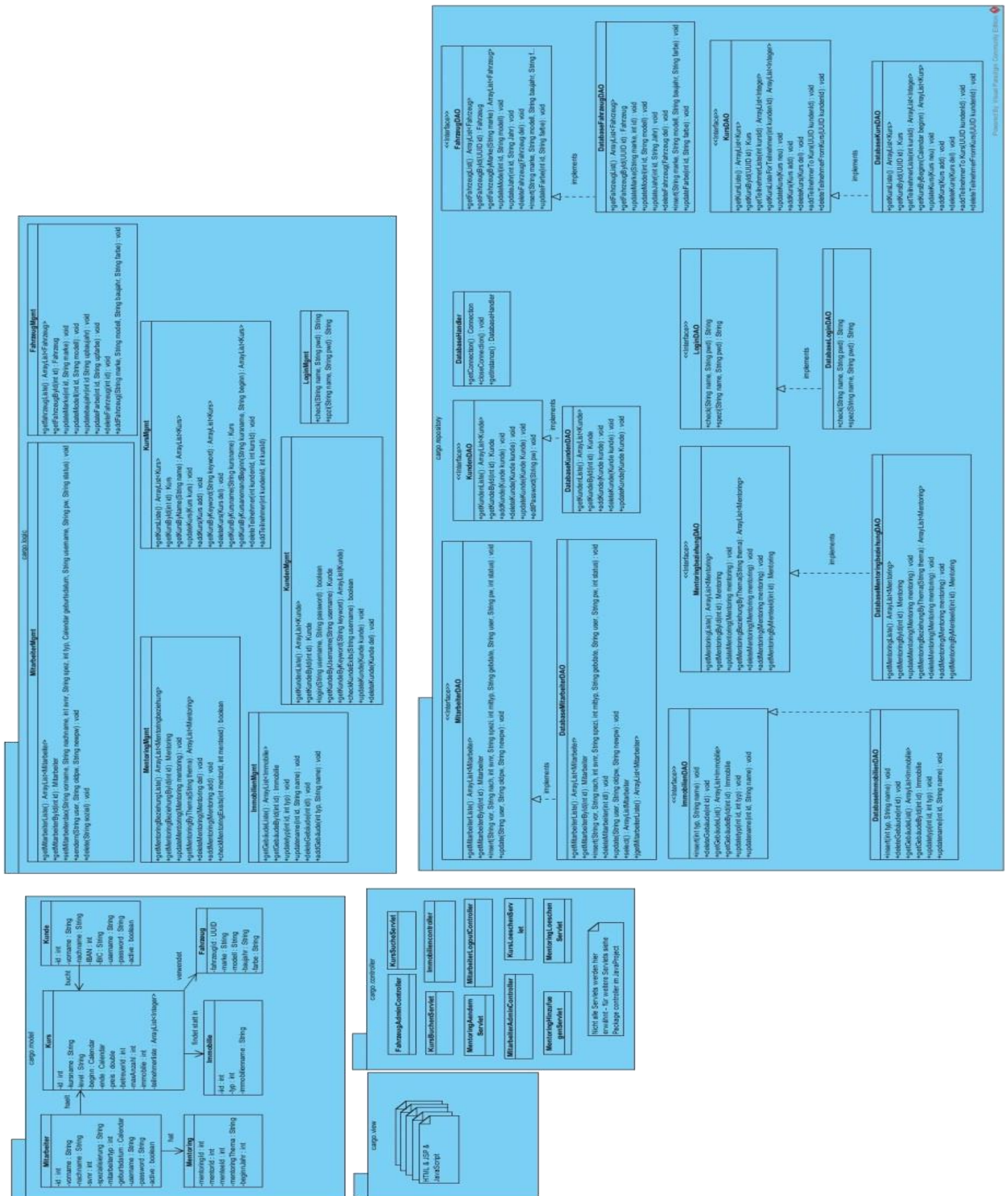
GRÜN: Nikola Babic (+
FinanzMgmt zusammen mit
Gregor Langner)

GELB: Cordula Eggerth
(+Index, Lageplan, Kunden-
Login, Logout)

BLAU: Gregor Langner
(+Mitarbeiter-Login,
FinanzMgmt zusammen mit
Nikola Babic)

ROT: Denise Gall


4. Anhang: Klassendiagramm



5. Anhang: Login-Hilfe

Login als Kunde: (implementiert von Cordula Eggerth)

Kunden-Account-Daten: Username: cargokunde; Password: cargokunde



HOME KURSKATALOG REGISTRIEREN LOGIN

LOGIN ALS KUNDE:

USERNAME:

PASSWORD:

Login

Hauptseite, wenn als Kunde eingeloggt:



HOME KURSKATALOG MEIN PROFIL ▾ LOGOUT

WILLKOMMEN BEI CARGO DRIVING SCHOOL

SIE SIND EINGELOGGT ALS: CARGOKUNDE

WIR FREUEN UNS UEBER IHREN BESUCH BEI DER CARGO FAHRSCHULE IN WIEN!
BEI UNS KOENNEN SIE SICH SCHNELL UND EINFACH UEBER DIE FAHRKURSE
IHRER WAHL INFORMIEREN, KURSE SUCHEN UND BUCHEN.
NATUERLICH KOENNEN SIE UNS AUCH VOR ORT BESUCHEN, WO WIR SIE PERSOENLICH
BERATEN.

KURSKATALOG

Login als Mitarbeiter: (implementiert von Gregor Langner)

Eingabe der Test-Mitarbeiterdaten:

Username: gregor

Passwort: db

Mittels dieser Login-Daten kommt man zur Hauptseite für die Mitarbeiter. Die Navigation erfolgt über die Navigationsleiste oben bzw. über die Buttons in den jeweiligen Seiten.

