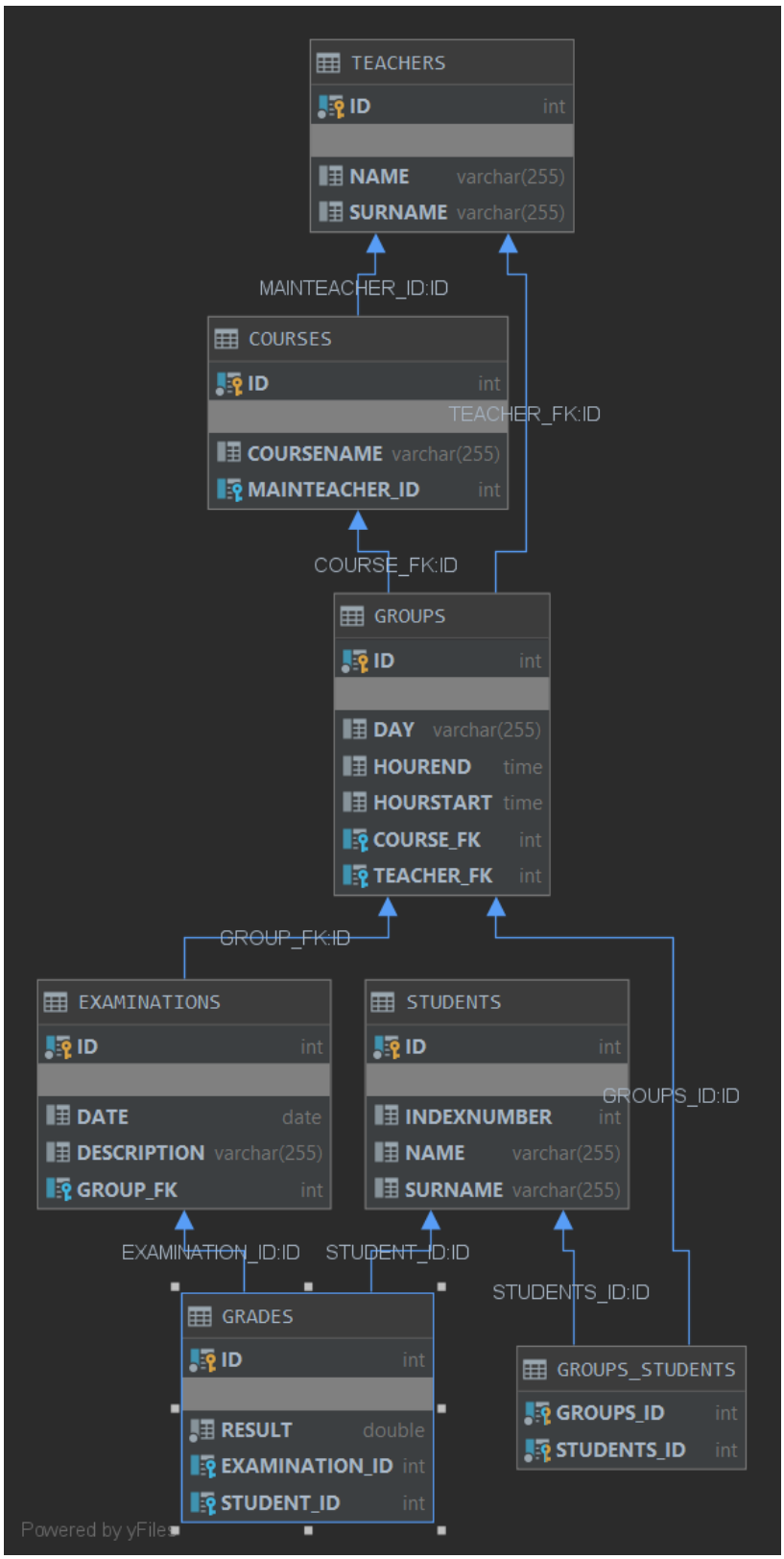
Dokumentacja

Diagram bazy danych



Opcje aplikacji

Aplikacja daje możliwość zalogowania się jako nauczyciel lub student za pomocą loginu. Dla uproszczenia loginem studenta jest jego indeks a loginem nauczyciela jest jego nazwisko.

Opcje nauczyciela:

1. Pokazanie listy prowadzonych kursów przez zalogowanego nauczyciela.
2. Pokazanie listy prowadzonych grup laboratoryjnych przez zalogowanego nauczyciela.
3. Pokazanie listy listy egzaminów z kursów prowadzonych przez zalogowanego nauczyciela.
4. Pokazanie listy studentów podanej grupy (należy podać id grupy).
5. Dodanie nowego egzaminu:
   Wymagane jest podanie id grupy dla której przeprowadzamy egzamin, daty przeprowadzenia egzaminu oraz opisu dotyczącego tego egzaminu.
6. Usunięcie wybranego egzaminu.(należy podać id egzaminu)
7. Dodanie nowej oceny z egzaminu:
   Wymagane jest podanie id egzaminu oraz indeksu ocenianego studenta.

Opcje studenta:

1. Pokazanie listy kursów, na które jest zapisany zalogowany student.
2. Pokazanie listy ocen jakie posiada zalogowany student.

Przykład użycia

Wstawienie nowego egzaminu i oceny z niego

Student - oceny na początku:

```
Bazy danych, Kolokwium z MongoDB 2020-04-20 - 5.0
Bazy danych, Kolokwium z Hibernate 2020-03-16 - 4.5
```

Nauczyciel – poprzednie kolokwia:

```
ExamId: 282, Bazy danych, Id grupy: 272, Kolokwium z Hibernate, 2020-03-16
ExamId: 358, Bazy danych, Id grupy: 272, Kolokwium na dzień dziecka, 2020-06-01
ExamId: 276, Bazy danych, Id grupy: 272, Kolokwium z MongoDB, 2020-04-20
```

Nauczyciel – wstawienie nowego kolokwium:

```
5
Enter group Id:
272
Enter date(rrrr-mm-dd):
2020-05-28
Description of exam:
Kolokwium z NoSql
```

Nauczyciel – lista kolokwiów:

```
3
ExamId: 458, Bazy danych, Id grupy: 272, Kolokwium z NoSql, 2020-05-28
ExamId: 282, Bazy danych, Id grupy: 272, Kolokwium z Hibernate, 2020-03-16
ExamId: 358, Bazy danych, Id grupy: 272, Kolokwium na dzień dziecka, 2020-06-01
ExamId: 276, Bazy danych, Id grupy: 272, Kolokwium z MongoDB, 2020-04-20
```

Nauczyciel – wstawienie oceny:

```
7
Enter Id of exam:
458
Enter Index number of Student:
296573
Enter result:
4.5
```

Student – lista ocen:

```
Bazy danych, Kolokwium z MongoDB 2020-04-20 - 5.0
Bazy danych, Kolokwium z Hibernate 2020-03-16 - 4.5
Bazy danych, Kolokwium z NoSql 2020-05-28 - 4.5
```

Struktura kodu

Wszystkie operacja wstawiania, aktualizowania itp. (operacje CRUD) są zrealizowane przy pomocy specjalnych klas Dao utworzonej do każdej encji – odpowiadają one za pośredniczenie między kodem samej aplikacji a dostępem do danych.

Nadklasa Dao:

```
public abstract class Dao {

   protected EntityManager em;
   protected EntityTransaction etx;

   public Dao(EntityManager em) {
      this.em = em;
      this.etx = em.getTransaction();
   }

   protected void beginTransaction() {
      try {
         etx.begin();
      } catch (IllegalStateException e) {
         rollbackTransaction();
      }
   }

   protected void commitTransaction() {
      try {
         etx.commit();
//          em.close();
      } catch (IllegalStateException e) {
         rollbackTransaction();
      }
   }

   protected void rollbackTransaction() {
      try {
         etx.rollback();
      } catch (IllegalStateException e) {
         e.printStackTrace();
      }
   }
}
```

Przykład podklasy – ExaminationDao:

```
public class ExaminationDao extends Dao {

   public ExaminationDao(EntityManager em) {
      super(em);
   }

   public Examination create(Group group, Date date, String description){
      Examination examination = new Examination(group, date, description);
      create(examination);
      return examination;
   }
```

```
    public void create(Examination examination){
        beginTransaction();
        em.persist(examination);
        commitTransaction();
    }

    public Examination find(int id) {
        return em.find(Examination.class, id);
    }

    public void addGrade(Examination examination, Student student, double result) {
        beginTransaction();
        examination.addGrade(student, result);
        commitTransaction();
    }

    public void addGradesOfExamination(Examination examination, Collection<Student> ratedStudents,
double result) {
        beginTransaction();
        for(Student s : ratedStudents) {
            examination.addGrade(s, result);
        }
        commitTransaction();
    }

    public void update(int id, Group group, Date date, String description) {
        beginTransaction();
        Examination examination = find(id);
        examination.setDate(date);
        examination.setGroup(group);
        examination.setDescription(description);
        em.merge(examination);
        commitTransaction();
    }

    public void delete(int id) {
        beginTransaction();
        Examination examination = find(id);
        if(examination != null) {
            em.remove(examination);
        }
        commitTransaction();
    }
}
```

Przykład wstawiania przykładowych danych przy pomocy Daos:

```
public static void main(final String[] args) throws IOException {

    EntityManagerFactory emf = Persistence.createEntityManagerFactory("myConfig");
    EntityManager em = emf.createEntityManager();

    StudentDao studentDao = new StudentDao(em);
    CourseDao courseDao = new CourseDao(em);
    GroupDao groupDao = new GroupDao(em);
    ExaminationDao examinationDao = new ExaminationDao(em);
    TeacherDao teacherDao = new TeacherDao(em);
```

```
   // create students group1
   Student student1 = studentDao.create("Grzegorz", "Janosz", 296573);
   Student student2 = studentDao.create("Adam", "Bera", 296546);
   Student student3 = studentDao.create("Grzegorz", "Kowalski", 296333);
   Student student4 = studentDao.create("Andrzej", "Ruba", 296685);
   Student student5 = studentDao.create("Alicja", "Zielińska", 296499);

   // group2 students
   Student student6 = studentDao.create("Aleksandra", "Jarosz", 296572);
   Student student7 = studentDao.create("Sławomir", "Kubeczko", 296446);
   Student student8 = studentDao.create("Irena", "Steczkowska", 297343);
   Student student9 = studentDao.create("Mariusz", "Smiry", 300185);
   Student student10 = studentDao.create("Anna", "Wolna", 300122);
   //siema, mała zmiana

   // create new course (teacher can be create without dao, because he is saved with creating course if
doesn't exist)
   Teacher teacher1 = new Teacher("Robert", "Marcjan");
   Teacher teacher2 = new Teacher("Leszek", "Siwik");
   Teacher teacher3 = new Teacher("Jarosłow", "Koźlak");

   Course course = courseDao.create("Bazy danych", teacher1);
   Course course2 = courseDao.create("Systemy operacyjne", teacher3);

   // create new group
   GroupTime groupTime = new GroupTime(DayOfWeek.TUESDAY, Time.valueOf("12:50:00"),
Time.valueOf("14:20:00"));
   Group group = groupDao.create(course, teacher1, groupTime);
   groupDao.addStudentsToGroup(group, Arrays.asList(student1, student2, student3, student4,
student5));

   GroupTime groupTime2 = new GroupTime(DayOfWeek.THURSDAY, Time.valueOf("14:40:00"),
Time.valueOf("16:10:00"));
   Group group2 = groupDao.create(course, teacher2, groupTime2);
   groupDao.addStudentsToGroup(group2, Arrays.asList(student6, student7, student8, student9,
student10));

   GroupTime groupTime3 = new GroupTime(DayOfWeek.TUESDAY, Time.valueOf("14:40:00"),
Time.valueOf("16:10:00"));
   Group group3 = groupDao.create(course2, teacher3, groupTime3);
   groupDao.addStudentsToGroup(group3, Arrays.asList(student1, student2, student5, student9,
student10));

   // create examination
   Examination examination = examinationDao.create(group, Date.valueOf("2020-04-20"), "Kolokwium z
MongoDB");
   examinationDao.addGradesOfExamination(examination, Arrays.asList(student1, student2), 5);
   examinationDao.addGradesOfExamination(examination, Arrays.asList(student3, student4), 4.5);
   examinationDao.addGrade(examination, student5, 3);

   Examination examination2 = examinationDao.create(group, Date.valueOf("2020-03-16"), "Kolokwium z
Hibernate");
   examinationDao.addGradesOfExamination(examination2, Arrays.asList(student1, student2), 4.5);
   examinationDao.addGradesOfExamination(examination2, Arrays.asList(student3, student4), 4.0);
   examinationDao.addGrade(examination2, student5, 3.5);

   BufferedReader userInputReader =
         new BufferedReader(new InputStreamReader(System.in));
   new DataBaseProject(studentDao,courseDao,groupDao,examinationDao, teacherDao,
userInputReader).run();
}
```